Modeling Impact Damage in Laminated Composite Plates

by

Trevor Tippetts

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2003

Author ....
Department of Aeronautics and Astronautics
May 9, 2003

Certified by . .....
S. Mark Spearing
Associate Professor
Thesis Supervisor

Accepted by..... .....
Edward M. Greitzer
H.N. Slater Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

Modeling Impact Damage in Laminated Composite Plates

by

Trevor Tippetts

Submitted to the Department of Aeronautics and Astronautics
on May 9, 2003, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

The simulation of impact damage in laminated composite plates presents many challenges to modelers. Local failure takes the form of various fracture processes. For composites this is further complicated because the various modes can strongly interact with each other. Another modeling challenge particular to composite materials is that it is often necessary to simulate the structural behavior simultaneously on different length scales. This makes it difficult to create a model that is computationally efficient.

Cohesive zone models (CZMs) have been developed to model crack growth in a material or debonding between two different materials and have alleviated many of the numerical problems inherent in crack modeling. However, significant errors can emerge in finite element models if the element are much larger than the crack process zone. These errors are often large enough to cause the solver to fail to converge except for very small structures.

In this thesis, an improved numerical integration algorithm is presented that solves the interface convergence problem. A Multiple Length Scale Finite Element Method (MLS-FEM) is also presented as a means of modeling both large-scale structural behavior as well as small-scale damage progression. The application of these models is with impact on laminated composite plates. Both of these models, however, can be applied to a wide variety of problems in composite structural analysis.

Thesis Supervisor: S. Mark Spearing
Title: Associate Professor

Acknowledgments

To my wonderful fiancé, Emily, who has been so supportive and helpful and unendingly patient, and whose personal achievements in her own education have been an inspiration to me: She is a woman of many talents and refined virutes; and the very fact that she is willing to marry me is a great testament to her courage, longsuffering, and faith.

I owe many thanks to Professor S. Mark Spearing, who has been my advisor both for this thesis as well as during my undergraduate years.

There are many people at Los Alamos National Laboratories who have given me guidance and encouragement during my work there as both an undergraduate and graduate student intern. Irene Beyerlein has been my mentor there since 1999 and has helped me to take my first steps into the world of Research, for which I am very grateful. Todd Williams, Chuck Farrar and others in the Damage Prognosis group at Los Alamos have been a great help to me and I sincerely thank them.

I thank the developers of the CalculiX finite element program, not only for writing a general-purpose finite element code, but also for releasing it under an open source license. Without access to the source code, I could not have carried out this research.

A heartfelt thank-you goes to my family, whose advice, emotional support, and loans from the Bank of Mom and Dad have helped me through many long days and longer nights at MIT.

Most of all, I thank God, for all that I have and am.

Divina luz, con esplendor benigno, alúmbrame.

Oscuras son la noche y la senda; mi Guía sé.

Muy lejos de tu pabellón estoy...

Y al hogar de las alturas voy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The use of fiber composite materials continues to increase, particularly in primary structural components. In order to ensure reliable and safe use of these composites in critical structural components, better models are needed for structural analysis. This is perhaps especially true in for the prediction of the onset of damage and subsequent performance in the presence of damage. The development of purely empirical models from coupon and other sample testing is costly and slows the development and adoption of new material systems. Uncertainties that arise from applying the results of such tests often necessitate high factors of safety.

The simulation of impact damage in laminated composite plates presents many challenges to modelers. One of the greatest challenges is that local failure occurs in the form of various types of fracture. Delaminations, transverse matrix cracking, fiber fracture, and ply splitting, also referred to as interfiber fracture or matrix cracking, are all important composite failure modes that are essentially fracture processes. Fracture of itself is not a simple phenomenon to model and continues to be an area of active research. In the case of composites it is further complicated by the fact that these various modes often occur near each other and can strongly interact. A realistic model for composite failure, then, requires an accounting of the interaction between the individual modes.

Cohesive zone models (CZMs), or interface damage mechanics, have been developed

over the last decade as a very flexible method of modeling crack growth in a material or debonding between two different materials. These methods are relatively convenient to implement in finite element models and have alleviated many of the numerical problems inherent in crack modeling. However, researchers have also found that significant errors emerge in finite element simulations if the mesh is too coarse relative to the crack process zone size. Often these errors are large enough to cause the solver to fail to converge unless the structure size is on the order of tens of millimeters.

Another modeling challenge particular to composite materials is the role of length scales. Failure phenomena usually evolve in composite materials on a length scale that is significantly smaller than the length scale of structural detail. This difference in length scales makes a monolithic, finely discretized mesh computationally unpractical. However, in many cases the length scale of failure is not so small that its behavior can be well approximated as homogeneous material properties. A modeling approach that can simultaneously and efficiently model structural behavior on both length scales is needed.

1.1 Objectives

This thesis shows that the errors that plague CZMs are due to inaccurate integration of the interface properties. The use of an improved numerical integration algorithm on the cohesive zone model is shown to solve the interface convergence problem. It is shown that the improved integration algorithm can significantly decrease the total computation time of fracture simulation, so that much coarser meshes and larger structures can be simulated.

A Multiple Length Scale Finite Element Method (MLSFEM) is also presented as a means of modeling both large-scale structural behavior as well as small-scale damage progression. The MLSFEM uses a superposition of global and local displacement fields. After substituting the fields into the finite element governing equations, the degrees of freedom can be separated into two finite element models, each on a single length scale. The degrees of freedom of the two models are related to each other by a set of constraint equations that

can be implemented in existing finite element software as multiple point constraints. In this way, it is possible to take advantage of existing finite element software that was developed to simulate a structure on a single length scale.

In this context the application of these two modeling approaches is the simulation of impact damage in laminated composite plates. Both of these models, however, can be applied to a wide variety of problems in fracture simulation and composite structural analysis.

Chapter 2

Literature Review

2.1 Introduction

Fiber composite materials are often the materials of choice when high strength and stiffness are needed together with low weight. The enormous number of possibile combinations of constituent components, ply stacking sequence, and relative ply angles give structure designers great flexibility in specifying laminate elastic properties. However, with this great variability comes great complexity, and the prediction of damage due to impact loading on laminated composites presents unique challenges.

Transverse impact on a laminated plate can cause damage significant enough to make a component fail to meet its structural requirements. The damage done can be invisible to the naked eye and therefore difficult and expensive to detect. If a primary structural component such as a load-bearing wing panel fails, it could cause a catastrophic failure of an entire system. The risk caused by this uncertainty has slowed the transfer of new composite materials from the laboratory to the factory. In many cases it has also increased the cost because older parts with useful remaining life are retired, simply to avoid the risk that they might have been damaged. Clearly, improved methods for modeling impact damage in laminated composites could reduce costs and speed implementation of improved composite materials.

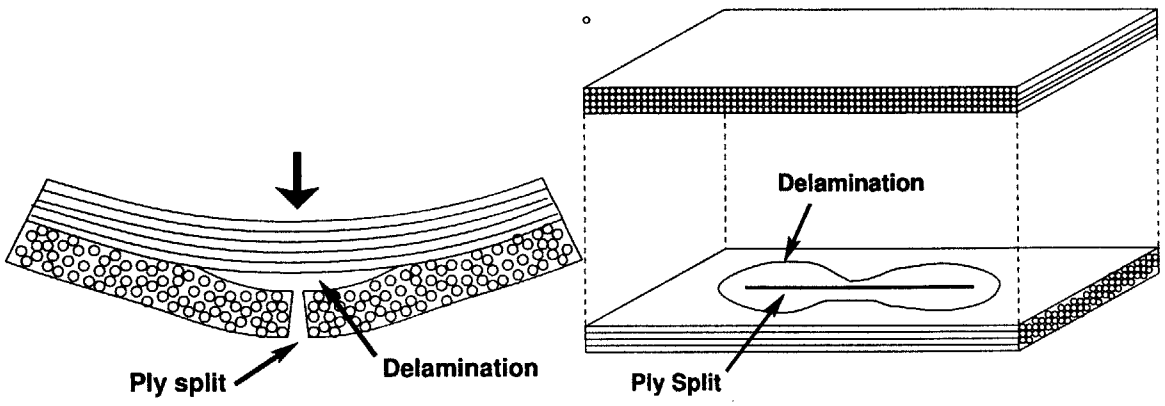


Figure 2-1: Schematic drawing of impact failure modes.

2.2 Failure modes for impact damage in laminated composite plates

Several micromechanical failure modes have been identified as critical in the failure and damage tolerance analysis of laminated composites. In impact damage, delamination is one of the most insidiously dangerous modes because the damage can be quite extensive before it is externally visible[8]. In the experimental work by Chang[8], damage was observed to initiate in the form of ply splitting in the plies opposite the impact location. When a ply split had propagated through the ply thickness, it initiated delamation, as shown in Figure 2-1. The delamination then propagated out from the ply split into a peanut-shaped void that was aligned parallel to the fibers in the lower ply.

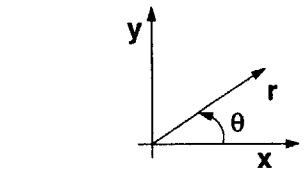
The ply split and the delamination are both fracture processes, and it is evident from experiment that the two modes are closely linked. A model for simulating impact damage behavior must account for this interaction.

2.3 Modeling approaches for fracture and interface failure

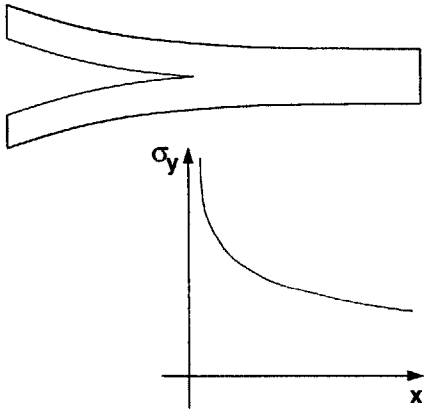
Many different approaches to modeling fracture have been taken. Perhaps the most classical example of fracture analysis is Linear Elastic Fracture Mechanics (LEFM). LEFM provides closed-form analytical expressions for stress, strain, and displacement fields around a crack tip[38]. However, these models are not amenable to implementation with a computer. Assuming linear elasticity leads to a singularity in stresses at the crack tip, $r = 0$, as shown in Figure 2-2(b). These infinite stresses could cause large roundoff errors if such a model were implemented by a computer with finite-precision arithmetic. Additionally, LEFM becomes very complicated in a system with anisotropic materials, mixed-mode fracture, or nonlinear material behavior[4].

Aside from the numerical problems inherent with LEFM, a linear constitutive model is often a poor model for a real material. In a real material, the fracture process involves not only the creation of new surfaces but also plastic yielding, microcracking, and other very localized, highly nonlinear material behavior in a small, highly stressed region near the crack tip. A hypothetical stress state along the crack plane due to such nonlinear processes is illustrated in Figure 2-2(c). Explicit modeling of this region of nonlinearity would require a very complex material constitutive model and a very fine discretization of the crack process zone. Additionally, a new model would need to be formulated for each new material system to be simulated.

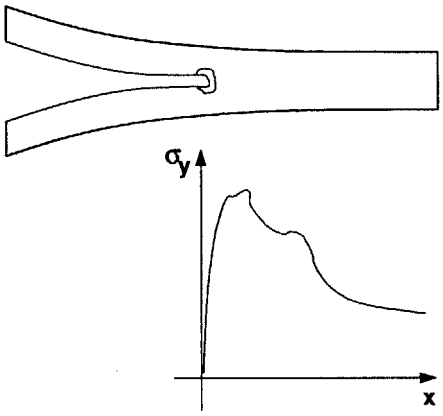
A number of analyses have been performed using strain energy release rates (SERR) or virtual crack extension techniques, which have been reviewed by Bolotin[5] and Storakers[32]. The models operate by testing whether hypothetical advances of a crack front are energetically favorable. SERR provides criteria for crack extension; however, this requires the assumed presence of a preexisting crack and direction of propagation. For many simulations, the location of crack initiation is one of the desired model outputs and therefore cannot be assumed *a priori*.



(a)



(b)



(c)

Figure 2-2: Modeling material behavior near a crack tip. (a) The coordinate system used has its origin centered on the crack tip and the x axis along the crack plane. (b) A linear elastic material has infinite stresses at the crack tip. (b) A more realistic material exhibits very complex nonlinear behavior in a small region near the crack tip.

2.3.1 Cohesive zone models

Cohesive zone models (CZMs) were developed to avoid the problems that plague linear elastic fracture mechanics. The first works on CZMs are attributed to Barenblatt[3] and Dugdale[13], who first put forth the concept of a finite region near a crack tip where the stresses were bounded by material strength. Barenblatt reasoned that the location of a crack tip is indeterminate if the stresses are singular. He argued that the stress near a crack tip must be finite because of the finite strength of the interatomic forces in any real material. Dugdale sought to model slits in a metal sheet under tension with a very simple CZM. He assumed that in the plane of the slits where the normal stress was constant, due to plastic yielding in the regions near the tips of the slits.

Since then an extensive body of work using CZMs has been developing. For example, in one of the earliest papers on cohesive zone models applied to a finite element simulation, Tvergaard[34] showed an application of CZMs to fiber debonding in a whisker-reinforced metal in a quasi-static model. Geubelle[17] used a dynamic bilinear CZM to simulate matrix cracking and delamination in a 2D composite laminate. Camacho[6] developed a CZM to model fracture and fragmentation in impact of brittle materials. Both Geubelle and Camacho incorporated a rate dependence to account for the change in the energy release rate with crack propagation velocity.

Following Tvergaard, Chaboche[7] used a CZM with a quadratic damage function to simulate a double cantilever beam and fiber debonding. He also added a viscous regularization to the CZM in order to reduce or eliminate erroneous "jumps" in the solution. The viscous regularization was shown to suppress the solution jumps at the expense of adding a non-physical time parameter and some deviation from conservation of energy. The same type of solution jumps are treated in this paper via adaptive integration. Corigliano[11] also demonstrated that an erratic error emerges for coarser meshes in a a double cantilever beam simulation, as well as other important numerical issues. Moura[22] applied a CZM to simulated compression of pre-impacted laminated composites in a quasi-static simulation. Dávila[12] also used a CZM for modeling composite failure by predicting the debonding

between composite skins and stiffeners. Samudrala[28] used a rate-dependent CZM to extend the model to intersonic mode II fracture.

2.4 Multi-length scale finite element modeling

In any structure, various physical phenomena take place on different length scales. Often it is most efficient to model a structure on a single length scale, *e.g.*, on the length scale with geometric detail of interest to the designer. All physical phenomena on length scales smaller than this are then homogenized into a material model. In many cases, however, an analyst is interested in structural behavior on more than one length scale simultaneously[31]. This is perhaps especially true in the case of composite materials, which by their very nature involve a heterogenous distribution of phases on at least one length scale between the structure and the length scales of the individual material phases.

One way to group the various models that have been developed to model structures on different length scales is into two categories[15]: homogenization and superposition.

2.4.1 Homogenization

In a homgenization model, a model is typically derived for a microstructure that is assumed to be periodic. A Representative Volume Element (RVE) is defined, in which the macroscale quantities are assumed to have a very simple variation, often constant, to facilitate a solution on this scale. The results from the RVE model are then averaged over the RVE region and fed back into the macroscale model.

While a vast body of literature exists on general multiscale structural analysis techniques, the most relevant efforts to the present work are those that can be or have been implemented with the finite element method. Feyel and Chaboche developed a model which they call the FE^2[14] model. Their approach was to use a finite element model with periodic boundary conditions of the microstructure, specifically, the fibers and matrix in a fiber composite, as a material model. A macroscale finite element model was employed

for the structural length scale and, at each integration point of the macroscale elements, the microscale finite element simulation is used to determine the material constitutive response.

Raghavan, Moorthy, Ghosh, and Pagano took a similar approach, using an adaptive multilevel model[24]. Computations with RVEs on smaller length scales are carried out in regions determined to have a greater error. This is continued recursively to the smallest material length scale, and the results at each scale are homogenized for inclusion in the next greater length scale.

With homogenization models, difficulties are encountered near boundaries or in any other case in which the assumption of periodicity is violated. This aspect of homogenization techniques complicates their adoption into modeling damage and failure. These phenomena are often very localized and aperiodic. In many cases, damage initiates on the microscale and eventually grows to become a macroscale feature. This transition between scales creates a modeling challenge that will likely continue for some time. The assumptions involved with homogenization that limit coupling between length scales hinder their ability to deal with these transitions.

The assumption that the macroscale fields have a very simple variation, or no variation at all, over the RVE is only appropriate when the two length scales are widely separated. In laminated composite materials, care must be taken with this assumption. Damage modes such as delamination can cover a region large enough to encounter significant variation that is very much dependent on the particular structural geometry and loading.

2.4.2 Superposition

The concept of superposing a small length scale variation on top of a large scale field is another means to model two (or more) length scales simultaneously. The superposition approach to multiple length scale modeling provides a means of considering local and global fields distinctly, as does homogenization. However, the local fields are not necessarily constrained by an assumption of periodicity, although often a homogenized model may be obtained as a special case of a superposition model by imposing a periodic constraint.

Superposition also allows for greater coupling between physical phenomena. It does so at the cost of a possibly increased model complexity and almost certainly more degrees of freedom, resulting in an increased computational cost.

A number of researchers have used extensions to the classical equivalent single layer theories[25, 37, 19, 20]. These include "zig-zag" or discrete layer plate theories[29, 30, 33, 18, 10, 9]. The various discrete layer plate theories center on a small-scale variation of the displacement and/or stress fields from a global field through the thickness of the plate. The in-plane variation of the fields is typically modeled on a single length scale. While the discrete layer plate theories allow local fields to be resolved through the thickness of the laminate, they do not provide a means for modeling damage in which the displacement could be discontinuous due to the presence of a crack. They have also been found to overpredict the stiffness for laminates with many lamina[2].

In order to address the specific problem of laminate plates, Reddy developed a variable kinematic theory that uses separate variations of the local fields for each ply through the thickness of the laminate[26, 27]. A similar method was used by Williams and Adessio in the Generalized Multilength Scale Plate Theory (GMLSPT)[40, 41, 39] with the added possibility of discontinuous displacements to allow for delamination.

In Fish's work[15], a method is presented for improving the local resolution on an existing mesh by superposing a displacement field within an element. The superposed displacement field is constrained to have a homogeneous Dirichlet boundary condition on its entire exterior so that the previously computed diplacement field is unchanged except within the superposed region. Although formulated differently, the result of this strategy for acheiving efficient local resolution is similar to the MLSFEM model presented in Chapter 4. One of the key differences between the two is that the MLSFEM does not constrain the local fields to have homogeneous boundary conditions.

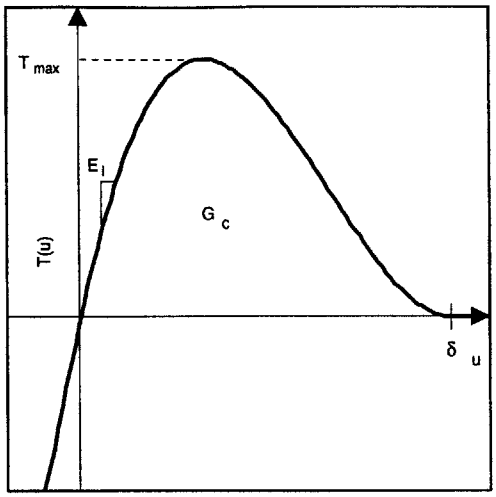
Chapter 3

Cohesive Zone Models

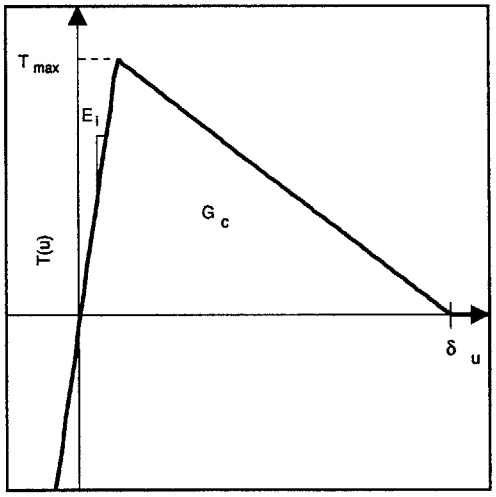
3.1 Cohesive Zone Model Formulation

In general, all cohesive zone model functions define a relationship between the displacement jump, u, across an interface and the interfacial tractions, $T(u)$[1, 11, 23, 21]. Like its applications, the functions used for the cohesive zone models can differ widely[6, 7, 17, 23, 34]. Two typical examples are shown in Figure 3-1. Generally a simple shape is chosen for the $T(u)$ curve, which is characterized by a small number of parameters. Four of the most used interface parameters are the critical displacement jump (δ), the maximum stress (T_{max}), the initial stiffness (E_i), and the fracture surface energy (*i.e.,* critical energy release rate, G_c). Usually only two or three are independent. With all else being equal, the crack process zone size, *i.e.* the crack region wherein the interface tractions are nonzero, increase with δ. In the case of mixed-mode fracture, T and u are vectors, and each interface parameter may have a component for each mode or relationships may exist between modal components to simulate coupling between modes[6, 17, 22, 12, 7]. This work will use δ, E_i, and G_c as interface parameters, only two of which will be independent.

One of the objectives of this work is to demonstrate how adaptive integration algorithms can improve accuracy and computational efficiency in cohesive zone fracture simulations using coarser meshes than considered to date. To demonstrate this, two Mode I fracture



(a)



(b)

Figure 3-1: Mode I interface models: Interfacial traction (T) is plotted as a function of the displacement jump (u.) G_c is the fracture surface energy, E_i is the initial stiffness, T_{max} is the maximum traction, and δ is the critical displacement jump. (a) cubic interface model. (b) Bilinear interface model.

problems will be analyzed, one quasi-static and one dynamic, as representative examples in Section 5.1. Many cohesive models coalesce when subject to only monotonic Mode I crack opening displacement. Therefore the two types of cohesive zone models shown in Figure 3-1 are considered sufficient to capture a broad range of cohesive zone models available.

3.1.1 Cohesive zone models

General Mixed-Mode CZM

Many CZMs in use today are specific cases of the following general form:

$$T(\mathbf{v}) = E_i \mathbf{v} F(\lambda) \tag{3.1}$$

For three dimensional fracture, $T(\mathbf{v})$ is a vector-valued function with three components, equal to the three components of interfacial traction. The variable $\mathbf{v}$ is a vector, also with three components, which represents each component of the displacement jump across the interface, normalized by δ. E_i is the initial stiffness of the interface. The parameters δ and E_i can each be specified uniquely for each mode, allowing for the possibility of fracture energies and maximum tractions that are dependent on the mode. In many of the CZMs in the literature, however, they are defined to be the same for all modes.

The function $F(\lambda)$ is a damage function that can take many forms, but typically has the following properties:

$$F(0) = 1 \text{ undamaged, pristine state}$$

$$F(\lambda \geq 1) = 0 \text{ complete separation}$$

$F(\lambda)$ is usually defined as a continuous function between 0 and 1.

In order for the fracture to be irreversible, *i.e.*, no crack healing, the damage parameter λ is defined such that it increases monotonically. This accounts for the fracture history in each interface element. If λ is increasing, it is often defined as the magnitude of the vector

ν.

$$\lambda = \max\left(|\nu|, \lambda_{previous}\right) \tag{3.2}$$

Numerically advantageous properties may be imparted to by the CZM by defining $F(\lambda)$ such that derivatives with respect to ν are continuous at $\lambda = 0$ and $\lambda = 1$, as explained in Section 3.2. The derivative of traction component T_j with respect to ν_k is

$$\frac{\partial T_j}{\partial \nu_k} = \delta_{jk} E_i F(\lambda) + T_{max} \nu_j \frac{\partial \lambda}{\partial \nu_k} \frac{\partial F(\lambda)}{\partial \lambda}, \tag{3.3}$$

where δ_{jk} is the Kronecker delta. If $|\nu|$ decreases, there will be a discontinuity in $\frac{\partial \lambda}{\partial \nu_k}$ (or any other derivative of λ with respect to ν) at the locus of points at which $\lambda = |\nu|$, due to the maximum operator. This cannot be avoided with a model of the form of Equations 3.1 and 3.2. However, in practice often the cracks of interest in a simulation are ones that are propagating, so that $|\nu| > \lambda_{previous}$. In these cases, derivatives of T will only be discontinuous if there is a discontinuity in a derivative of $F(\lambda)$.

Example CZM: Monotonic Mode I

The first CZM example used to validate the adaptive interface integration in Section 5.1 uses a cubic polynomial (see Equation 3.5 and Figure 3-1(a)) This can be obtained by constraining the Chaboche[7] or Tvergaard[34] model to monotonic mode I fracture.

$$\nu = \frac{u}{\delta} \tag{3.4}$$

$$T(u) = \begin{cases} E_i \nu & \nu \leq 0 \\ E_i \nu (1-\nu)^2 & 0 < \nu \leq 1 \\ 0 & \nu > 1 \end{cases} \tag{3.5}$$

The second example uses a bilinear model which is equivalent to the cohesive zone model used by Geubelle[17] in monotonic mode I fracture and similar to that of Camacho[6]. The bilinear model is defined in Equation 3.6 and shown in Figure 3-1(b).

$$T(u) = \begin{cases} E_i \nu & 0 < \nu \leq \frac{T_{max}}{E_i} \\ \frac{T_{max}}{1 - \frac{T_{max}}{E_i}} (1 - \nu) & \frac{T_{max}}{E_i} < \nu \leq 1 \\ 0 & \nu > 1 \end{cases} \tag{3.6}$$

These two cohesive zone models share a few key features. First if the jump in displacement across the interface is negative, the interface has a linear stiffness to penalize interpenetration in both models. Second as a positive displacement jump increases from zero, these interfaces are initially linear, representing an undamaged cohesive zone. Then the interfacial traction reaches a maximum and the interface softens, with a negative stiffness. If the displacement jump reaches the critical displacement jump, the interfacial traction is zero and the interface is completely separated. Therefore, the relative displacement between the two interface surfaces, u, within the cohesive zone is less than δ.

Like nearly all CZMs, both the cubic and the bilinear model are continuous functions of ν. However, because cohesive zone models are usually defined as piecewise functions, nearly all cohesive zone models have at least some derivatives with discontinuities. In this respect, the two models are different. The bilinear model is discontinuous in the first derivative at $\nu = \frac{T_{max}}{E_i}$ and $\nu = 1$. The cubic model is first-derivative continuous, but it is discontinuous in the second and third derivatives at $\nu = 0$ and $\nu = 1$, respectively. These discontinuities will be shown to play an important role in the accuracy of simulations involving cohesive zone models.

3.1.2 CZM as a regularization of LEFM

One way to view cohesive zone models and to relate material constants to CZM parameters is to regularize a linear elastic fracture model. The Williams series [38] gives the elasticity solution for a semi-infinite mode I stationary crack in a linear elastic isotropic 2D material in a polar coordinate system with coordinates r and θ. The singular term in the series for normal stress (T) and displacement jump across the interface (u) is

$$T = \sqrt{\frac{8G_cG}{\pi C_1 r}} \cos\frac{\theta}{2}\left(1+\sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right) \quad u = \sqrt{\frac{G_c r}{\pi C_1 G}}\left(C_2\sin\frac{\theta}{2}-\sin\frac{3\theta}{2}\right) \tag{3.7}$$

where the constants C_1 and C_2 are, in plane strain,

$$C_1 = 4(1-\nu) \quad C_2 = 7-8\nu \tag{3.8}$$

and, in plane stress,

$$C_1 = \frac{4}{1+\nu} \quad C_2 = \frac{7-\nu}{1+\nu}\ . \tag{3.9}$$

The symbols G and ν represent the shear modulus and Poisson's ratio, respectively.

In the plane of the crack ($\theta = 0$,) Equation 3.7 shows the classic $1/\sqrt{r}$ singularity that makes the Williams solution unsuitable for computational models. However, if Equation 3.7 is instead solved at a small distance y ($y = r\sin\theta$) above the crack plane, an expression without singularities is obtained:

$$\mathbf{u} = \frac{1}{\sqrt{\sin\theta}}\left(C_2\sin\frac{\theta}{2}-\sin\frac{3\theta}{2}\right) \quad \mathbf{T} = 2\sqrt{2\sin\theta}\ \cos\frac{\theta}{2}\left(1+\sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right). \tag{3.10}$$

In Equation 3.10, normalized quantities for stress $\mathbf{T}$ and displacement $\mathbf{u}$ have been introduced, which are

$$\mathbf{u} = \left(\sqrt{\frac{GC_1y\pi}{G_c}}\right)\frac{u}{y} \quad \mathbf{T} = \left(\sqrt{\frac{GC_1y\pi}{G_c}}\right)\frac{T}{G} \tag{3.11}$$

and which contain the length scale parameter y. Equation 3.10 is plotted parametrically in Figure 3-2, with the parametric variable θ varying from 0 to π.

A CZM may be regarded as simply an approximation to the constitutive model shown in Figure 3-2. Similar expressions may be derived to relate an analytical solution for mixed mode, bimaterial interfaces, or nonlinear materials to a corresponding CZM.

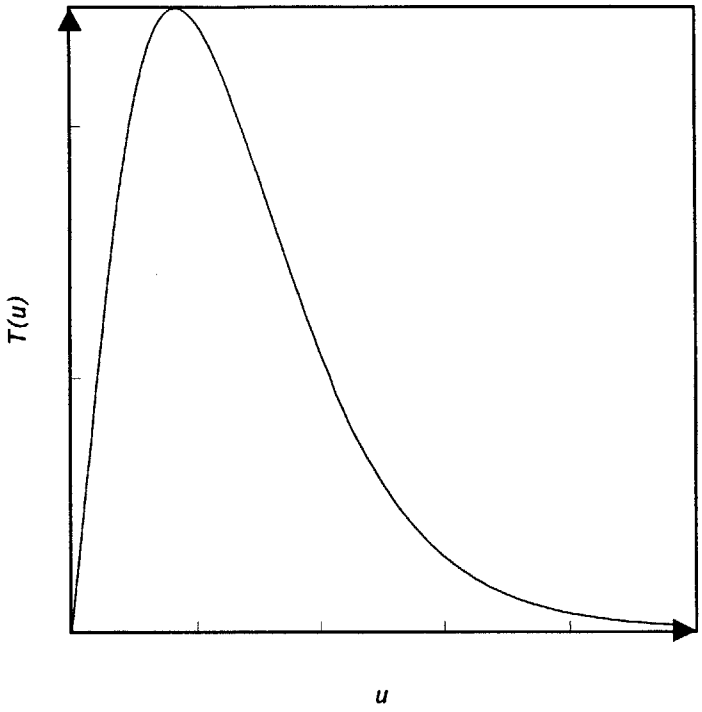


Figure 3-2: Parametric plot of Equation 3.10 for $\nu = 0.3$. The parametric variable spans the interval $0 < \theta < \pi$.

3.2 Error and integration order

3.2.1 The $\frac{\delta}{h}$ length scale parameter

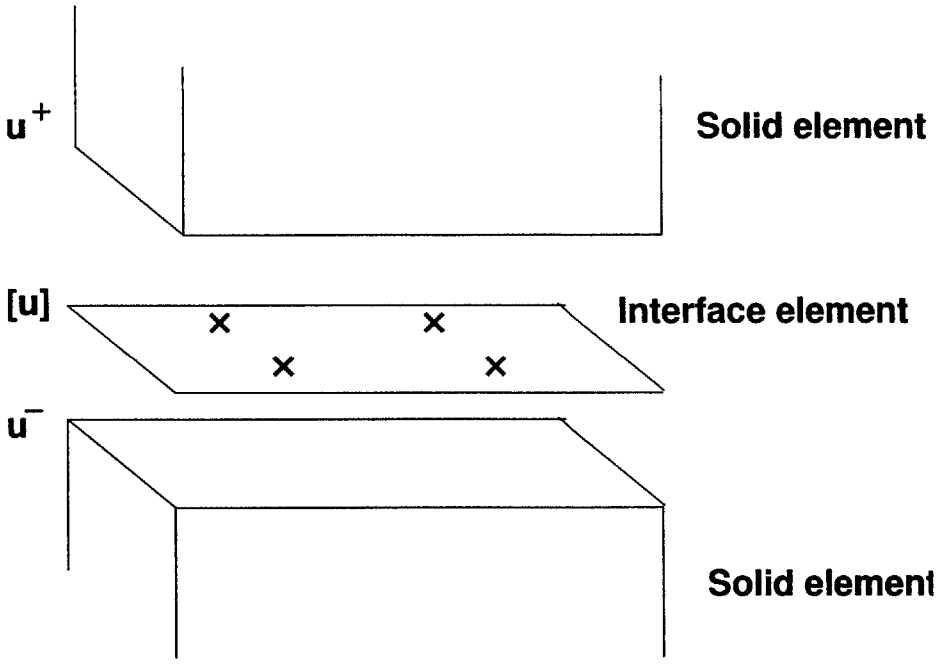


Figure 3-3: CZM in a finite element model.

In a finite element model, the CZM is implemented as an interface element that is between two solid elements, as shown in Figure 3-3. The displacement jump, $[u]$ in Figure 3-3, that correponds to the independent variable of the cohesive zone model is the difference between the displacements of the two opposing nodes, $u^+ - u^-$.

Every finite element/cohesive zone simulation has error in the discretized representation of both the interface and the solid regions of the structure. As the finite element mesh is refined, the contribution to the error in nodal displacements from the interface elements and from the solid elements will both decrease but generally at different rates. In many cases, the interface elements contribute much more to the total error than the solid elements. Therefore, separating the two sources of error and reducing the interface error independently from the solid element error can greatly increase the computational efficiency.

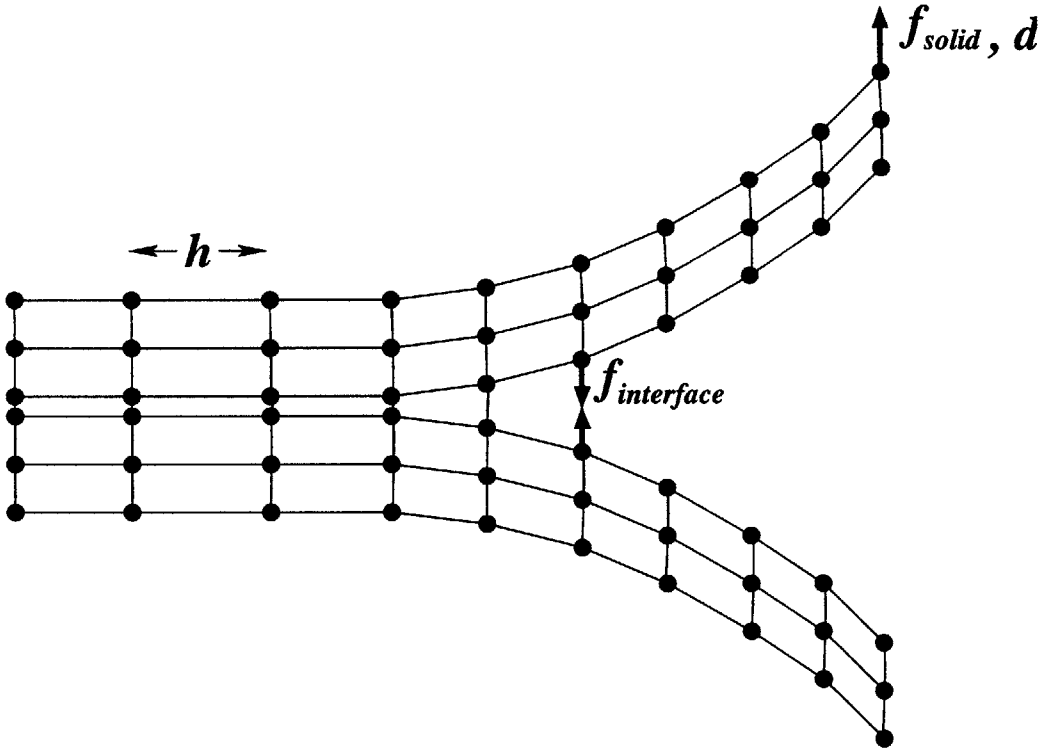


Figure 3-4: Example of a finite element model with cohesive zone model elements, to illustrate the importance of the $\frac{\delta}{h}$ length scale parameter.

A simple order of magnitude example of error in finite element analysis will serve to show the significance of the length scale ratio parameter $\frac{\delta}{h}$, where h is the characteristic element length. For this example, the output feature of interest is the reaction force at a node with a prescribed displacement, as shown in Figure 3-4. The measure of error is defined to be the absolute error in the nodal force near a crack tip divided by the reaction force output.

In a finite element model with small displacements, the magnitude of the nodal reaction force in a solid element, where the prescribed displacement is applied, is designated f_{solid}. This force is proportional to the Young's modulus (E), the characteristic element length (h), and the nodal displacements (d).

$$O(f_{solid}) = O(E)O(h)O(d) \tag{3.12}$$

The magnitude of the internal nodal forces in interface elements using a CZM ($f_{interface}$) is determined by the interface fracture energy (G_c), the interface critical displacement jump (δ), and the element area (h^2). The interface elements are not assumed necessarily to be near the node where f_{solid} is applied, but they are assumed to have dimensions of the same order of magnitude.

$$O\left(f_{interface}\right) = O(G_c)\,O\left(\frac{1}{\delta}\right)O\left(h^2\right) \tag{3.13}$$

The absolute integration error of the interface element traction is designated $\Delta f_{interface}$. The relative integration error is here defined as the ratio of the integration error to the magnitude of the interface nodal forces and define a parameter ε to be of the same order. The value of ε will depend on the integration algorithm, as described in Section 3.2.2.

$$O\left(\frac{\Delta f_{interface}}{f_{interface}}\right) = O(\varepsilon) \tag{3.14}$$

The contribution of the interface error to the solid simulation can be represented by the ratio of the interface nodal force integration error to the applied force in the solid.

Thus using Equation 3.12 through Equation 3.14 the relative interface error is related to the model parameters in the following manner:

$$O\left(\frac{\Delta f_{interface}}{f_{solid}}\right) = O(\varepsilon)O\left(\frac{G_c}{Ed}\right)O\left(\frac{h}{\delta}\right) \tag{3.15}$$

According to Equation 3.15, the error in simulating an interface may be reduced by either refining the mesh (as $\frac{\delta}{h}$ increases) or by improving accuracy of the integration algorithm (as ε decreases).

This convergence behavior in simulating crack propagation is important in the study of large structures with many cracks because the size of the solid elements must often be much greater than the size of the fracture process zone. In the literature, cohesive zone models have been typically used with $O\left(\frac{\delta}{h}\right) = 0.01$ to 0.1. Equation 3.15 explains why coarser meshes, which would lead to even smaller values than these, result in larger errors and hence convergence problems or erroneous crack velocities. Increasing $\frac{\delta}{h}$, however, in an attempt to lower Equation 3.15, is often not a viable option. For instance, if δ is on the order of $10\mu m$, a larger $\frac{\delta}{h}$ corresponds to element sizes smaller than 0.1 to $1mm$. These element sizes are prohibitively small for the simulation of many practical structures. Therefore, the capacity to simulate crack propagation for small $\frac{\delta}{h}$ ratios (when the mesh is coarse and δ is naturally small) is desirable. This requires an integration algorithm with a small relative error.

In summary, the challenge remains in keeping ε small, when at the same time $\frac{\delta}{h}$ is small, such that the computationally efficient coarse mesh fracture simulations do not produce erroneous crack velocities or even worse, fail to complete. This is the objective of this chapter; that is, to apply an adaptive integration algorithm in cohesive zone model fracture simulations to reduce ε when $\frac{\delta}{h}$ is smaller than 0.01. In Section 3.2.3, different integration schemes are described, beginning with the traditional fixed order algorithms and ending with a new application of a general adaptive integration scheme.

3.2.2 Integration error for a discontinuous function

It is necessary to analyze the integration error of a discontinuous function in order to determine the relationship between the relative integration error, ε in Equation 3.14, with other model parameters. If the function to be integrated, $f(x)$, is a polynomial with $k-1$ continuous derivatives and a discontinuity in the k^{th} derivative at x_d, $f(x)$ can be represented as a $k-1$ order polynomial plus a discontinuous function f_d.

$$f_d(x) = \begin{cases} f_{d1}(x) & x_d \leq x \\ f_{d2}(x) & x > x_d \end{cases} \tag{3.16}$$

If $f(x)$ is integrated with an algorithm of order greater than or equal to k, the $k-1$ order polynomial will be integrated exactly. Therefore, in considering the error of numerical integration of a function with a discontinuity in a derivative, it is only necessary to consider the error of integrating a piecewise function of the form

$$f_d(x) = \begin{cases} g(x) & x_d \leq x \\ 0 & x > x_d \end{cases} \tag{3.17}$$

The integral of f_d is approximated by a weighted sum of the integrand, sampled at n points, x_i.

$$\int_{x_1}^{x_2} f_d(x)dx = \int_{x_1}^{x_d} g(x)dx \cong \frac{x_2 - x_1}{2} \sum_{i=1}^{n} f_d(x) = \frac{x_2 - x_1}{2} \sum_{i=1}^{m} w_i g(x_i) \tag{3.18}$$

$$\begin{aligned} x_i, x_d &\in [x_1, x_2] & i &= 1, \ldots, n \\ x_i &\in [x_1, x_d] & i &= 1, \ldots, m \\ & & m &\leq n \end{aligned}$$

Only m points need to be considered in Equation 3.18 because $f_d(x_i) = 0$ for all $i > m$. The error of this integral is defined as

$$\Delta G = \int_{x_1}^{x_d} g(x)dx - \frac{x_2 - x_1}{2} \sum_{i=1}^{m} w_i g(x_i). \tag{3.19}$$

The mean value theorem gives

$$\int_{x_1}^{x_d} g(x)dx = (x_d - x_1)g(x_{mean}) \qquad x_{mean} \in [x_1, x_d] \ . \tag{3.20}$$

The integration error may be expressed in terms of the average g.

$$\Delta G = (x_d - x_1)g(x_{mean}) \left(1 - \frac{1}{2} \left(\frac{x_2 - x_1}{x_d - x_1} \right) \sum_{i=1}^{m} w_i \frac{g(x_i)}{g(x_{mean})} \right) \tag{3.21}$$

For a quadrature rule with positive weights,

$$O(w_i) = O\left(\frac{1}{n}\right), \tag{3.22}$$

where n is the number of integration points. Therefore, if p is the order of the integration algorithm, the order of the error is

$$O(\Delta G) = O(x_2 - x_1)\, O(g)\, O\left(\frac{1}{p+1}\right), \tag{3.23}$$

using the fact that $n = \frac{2}{p+1}$. Equation 3.23 is most obvious if, for example, $m = 0$ in Equation 3.21.

It is clear from two Taylor series, expanded from a small distance on either side of x_d, that

$$O(f_d) = O\left((x - x_d)^k\right) O\left(\Delta f^{(k)}\right), \tag{3.24}$$

where $\Delta f^{(k)}$ is the jump in the k^{th} derivative of f with respect to x at x_d. Therefore, the numerical integration error for the integral $\int_{x_1}^{x_2} f(x)dx$ is of order

$$O(\Delta F) = O\left((x_2 - x_1)^{k+1}\right) O\left(\Delta f^{(k)}\right) O\left(\frac{1}{p+1}\right). \tag{3.25}$$

If $[x_1, x_2]$ is a subdomain of length h_{sub} in a domain of integration of length h, then the order of the subdomain integration error relative to the integral over the entire domain is

$$O\left(\frac{\Delta F}{F}\right) = O\left(\frac{h_{sub}^{k+1}}{h}\right) O\left(\frac{\Delta f^{(k)}}{f}\right) O\left(\frac{1}{p+1}\right). \tag{3.26}$$

In two dimensions, an analogous development leads to the following expression:

$$O\left(\frac{\Delta F}{F}\right) = O\left(h_{sub}^{k}\left(\frac{h_{sub}}{h}\right)^{2}\right) O\left(\frac{\Delta f^{(k)}}{f}\right) O\left(\frac{1}{(p+1)^2}\right). \tag{3.27}$$

Integration error for CZMs

For a CZM used in a finite element model, the integrand of interest is either the interfacial traction (for the nodal force vector) or the derivative of the interfacial traction with respect to a degree of freedom (for the stiffness matrix.) For both of these integrands,

$$O\left(\frac{\Delta f^{(k)}}{f}\right) = \frac{1}{h^k}. \tag{3.28}$$

The relative error for a two dimensional interface element (in a three dimensional finite element model) is

$$O\left(\frac{\Delta F}{F}\right) = O\left(\left(\frac{h_{sub}}{h}\right)^{2+k}\right) O\left(\frac{1}{(p+1)^2}\right). \tag{3.29}$$

In Equation 3.29, note that the quantity $\frac{\Delta F}{F}$ is equivalent to ε in Equations 3.14 and 3.15.

When a fixed order integration algorithm is used, the domain of integration is never divided so that $h_{sub} = h$. A fixed order algorithm, as expected, gives a constant level of accuracy.

For adaptive integration, each subdomain is integrated with a fixed order quadrature. Regardless of how small the subdomains are divided, one or more will contain the discontinuity. However, with adaptive integration it is possible to maintain the relative error at an

optimal level independently from h because h is now independent from h_{sub}.

Equation 3.29 also indicates that CZMs with more continuous derivatives (*e.g.*, the first-derivative continuous cubic model) give rise to smaller integration errors due to the higher value of the exponent k.

3.2.3 Numerical integration algorithms

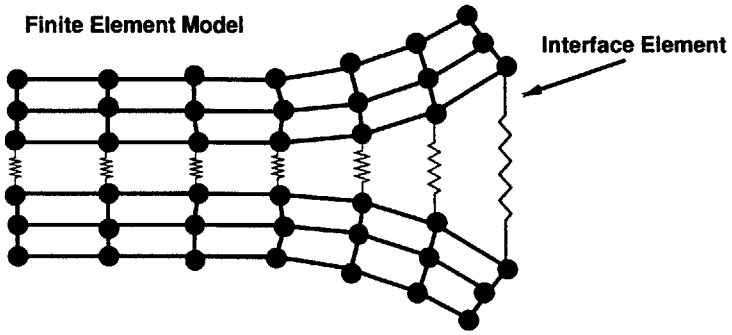
Numerical integration of a function $f(x,y)$ over a 2D finite domain is generally a weighted sum of samples of the function, as shown in Equation 3.30.

$$\int\int f(x,y)dxdy \cong \sum_{i=1}^{N} w_i f(x_i,y_i) \tag{3.30}$$

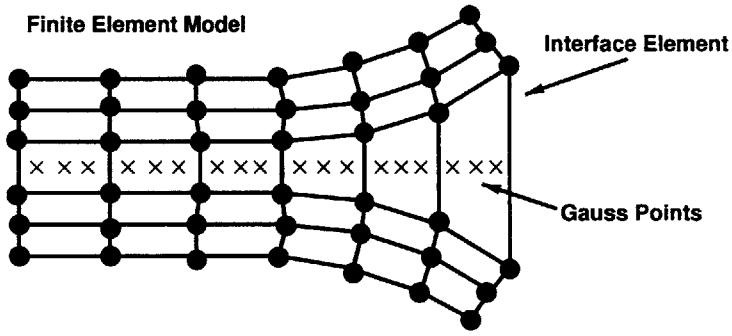
A set of integration points (x_i,y_i) and their corresponding weights (w_i) together form an integration rule or algorithm. The "order" of the integration algorithm is the highest-order polynomial function that can be exactly integrated by the algorithm.

Zero order integration

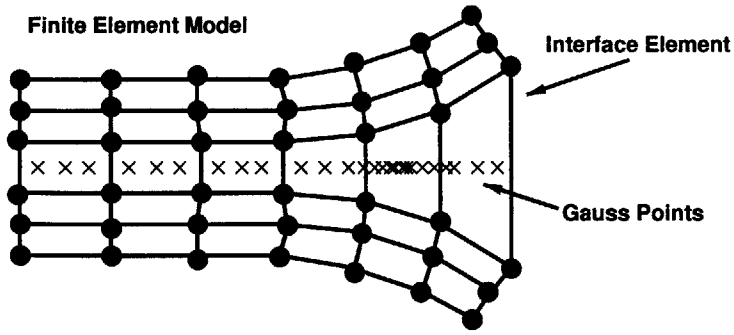
Some applications of cohesive elements use the difference in displacements only at nodal points, not along the element edges or faces. That is, the interface is represented as discrete node-to-node connections across the crack face only, as shown in Figure 3-5(a). This method of discretization is equivalent to a zero-order integration of a continuous cohesive element, with integration points at the nodes, because such an integration algorithm would exactly integrate a zero order polynomial. Therefore, the truncation error of the integration would be on the order of the interface length, $O(h)$. Thus the solution using a zero order integration algorithm converges as the mesh is refined, but the convergence is slow. Table 3.1 shows the positions of the four integration points and their weights.



(a) Zero order integration



(b) Fifth order integration with three Gauss points.



(c) Adaptive integration, with more integration points in regions with high derivatives.

Figure 3-5: Interface elements with various integration orders.

Table 3.1: Integration point locations and weights for zero order (nodal) integration. The domain of integration is $-1 \leq x,y \leq 1$.

x_i	y_i	w_i
± 1	± 1	1

Higher order integration

A simple and easy way to achieve higher accuracy in the interface integration is to use a higher-order numerical integration algorithm; e.g., Gaussian quadrature. Computationally, this amounts to the evaluation of interface properties inside the element at Gauss points (Figure 3-5(b).) Many researchers have used interface elements with three or four integration points in each direction of the crack surface[6, 17, 23]. As an example, Table 3.2 shows the points and weights for a fifth order integration algorithm. Figure 3-6(a) shows the locations of the integration points in the x,y plane.

Table 3.2: Integration point locations and weights for fifth order integration. The domain of integration is $-1 \leq x,y \leq 1$.

x_i	y_i	w_i
0	0	.790123
$\pm$.774597	0	.493827
0	$\pm$.774597	.493827
$\pm$.774597	$\pm$.774597	.308642

It is noteworthy that, with approximately the same number of points, a first order algorithm with $O\left(h^2\right)$ convergence may be implemented by evaluating the integrand in the center of the element rather than at the nodes. Therefore, there seems to be little or no reason to use a zero order algorithm for a cohesive zone model.

A polynomial of order $p = 2n-1$ over a one-dimensional domain can be exactly integrated with n Gauss points. Therefore, the truncation error of Gaussian quadrature for an arbitrary function is $O(h^{2n})$, where h is the length of the integration domain, but this

is *only if* the derivatives of that function are defined over the interval. Nearly all interface constitutive models, however, are defined piecewise. They therefore have derivatives with discontinuities; see, for example, Equations 3.5 and 3.6. These discontinuities tend to slow the convergence of higher order integration algorithms.

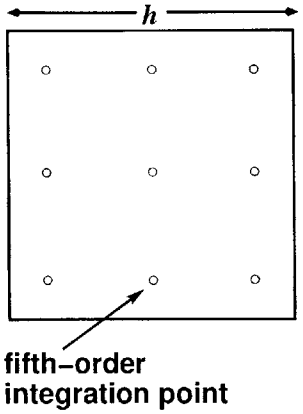
Adaptive integration

The employment of an adaptive algorithm is proposed in order to overcome the limitations of a fixed-order integration algorithm. Cohesive element constitutive models are typically piecewise low-order functions. Therefore, in most cases the best approach to integration would be a domain subdivision algorithm coupled with low-order numerical integration for each subdomain (Figure 3-5(c).) Because it is often possible to detect *a priori* whether or not a derivative discontinuity exists in a given interval, the domain subdivision algorithm could be applied only when there is a derivative discontinuity within the element to reduce computation costs.

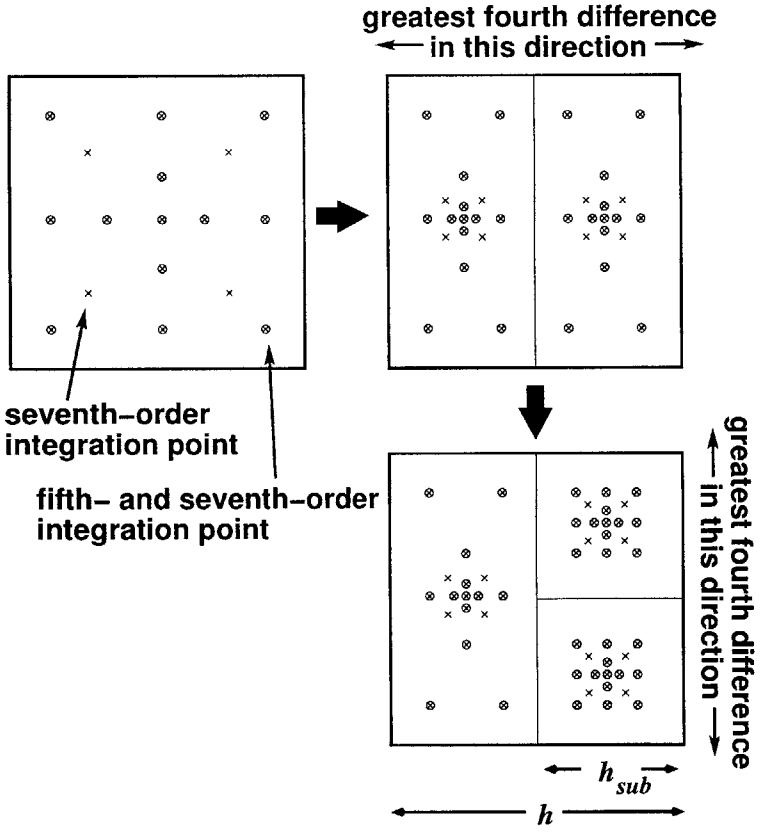
The adaptive integration algorithm used for the cohesive zone models and fracture examples in this work is that of Genz and Malik[16], which was modified from that of van Dooren and de Ridder[35]. The strategy used is to integrate the domain with two fixed-order numerical integration algorithms, a lower and higher order algorithm, which in this case will be fifth and seventh order respectively. The higher order algorithm is used to evaluate the integral, and the lower order algorithm is used to estimate the integration error. If the error is above a user-defined tolerance, the domain divides in half. The coordinate $j = x, y$ to be divided is the one with the largest absolute fourth difference, $D_j^4[f]$, of the integrand f in Equation 3.30. (See Figure 3-6(b).)

$$\begin{aligned} D_x^4[f] &= f(-x_2,0) - 2f(0,0) + f(x_2,0) - \frac{x_1^2}{x_2^2}\left(f(-x_1,0) - 2f(0,0) + f(x_1,0)\right) \\ D_y^4[f] &= f(0,-y_2) - 2f(0,0) + f(0,y_2) - \frac{y_1^2}{y_2^2}\left(f(0,-y_1) - 2f(0,0) + f(0,y_1)\right) \end{aligned} \tag{3.31}$$

The points used in Equation 3.31 coincide with integration points on each axis; that is,



(a)



(b)

Figure 3-6: (a) Fifth order Gauss integration. (b) Adaptive integration by recursive domain subdivision.

$$x_1, y_1 = \sqrt{\frac{9}{70}} \quad x_2, y_2 = \sqrt{\frac{9}{10}} \ . \tag{3.32}$$

The algorithm is then applied recursively to the subdivision with the greatest integration error until the total error is within the tolerance or until another stopping criterion, such as a maximum number of subdivisions, is reached. The lower order integration points and the points used to calculate the fourth difference ($D_j^4[f]$) coincide with a subset of the higher order integration points, as shown in Table 3.3, so that no additional integrand evaluations are needed. As with the examples in Section 3.1.1, it is possible to determine whether or not a derivative discontinuity exists within an interface element. This subdivision integration algorithm is applied to an interface element only if it is determined that a derivative discontinuity exists within the element. Otherwise, a fifth order Gauss integration is used.

The subdivision approach limits the integration error as a result of discontinuous derivatives by bounding them in a sufficiently small subdivision. In this way, the contribution of the interface integration error to the total error is maintained below a user-defined limit, regardless of the fineness of the mesh.

Choosing adaptive integration parameters

For the problems addressed in this thesis, the following set of integration rules was found suitable: the higher and lower order algorithms were set to seventh and fifth respectively, and the coordinate with the largest error defined by 3.31 was subdivided into halves. Because the adaptive integration algorithm used here was originally proposed as a very general algorithm for N-dimensional domains, it is likely that more fine-tuning of the integration rules used will result in higher efficiency. For example, it is possible to use a pair of integration algorithms other than the fifth and seventh order algorithms used here to estimate the integral and the error. The optimal set of integration rules would depend on the type of cohesive zone model used in the interface elements.

Because the adaptive algorithm checks its own error, it is necessary to specify an error tolerance. A smaller error tolerance will require more computer time to achieve, so the

Table 3.3: Integration points locations and weights for each subdomain in the adaptive order integration algorithm. The domain of integration is $-1 \leq x, y \leq 1$.
Fifth order integration rule:

x_i	y_i	w_i
0	0	$-\frac{3884}{729}$
$\pm\sqrt{\frac{9}{70}}$	0	$\frac{980}{486}$
0	$\pm\sqrt{\frac{9}{70}}$	$\frac{980}{486}$
$\pm\sqrt{\frac{9}{10}}$	0	$\frac{130}{729}$
0	$\pm\sqrt{\frac{9}{10}}$	$\frac{130}{729}$
$\pm\sqrt{\frac{9}{10}}$	$\pm\sqrt{\frac{9}{10}}$	$\frac{100}{729}$

Seventh order integration rule:

x_i	y_i	w_i
0	0	$-\frac{15264}{19683}$
$\pm\sqrt{\frac{9}{70}}$	0	$\frac{3920}{6561}$
0	$\pm\sqrt{\frac{9}{70}}$	$\frac{3920}{6561}$
$\pm\sqrt{\frac{9}{10}}$	0	$\frac{4080}{19683}$
0	$\pm\sqrt{\frac{9}{10}}$	$\frac{4080}{19683}$
$\pm\sqrt{\frac{9}{10}}$	$\pm\sqrt{\frac{9}{10}}$	$\frac{800}{19683}$
$\pm\sqrt{\frac{9}{19}}$	$\pm\sqrt{\frac{9}{19}}$	$\frac{6859}{19683}$

tolerance should be set as high as possible. In order for the solver to reach convergence, errors as high as 1 to 5 percent are often perfectly acceptable. Accuracy requirements in the results might make a lower tolerance necessary.

It is possible for the error estimator to overestimate the error, causing the adaptive algorithm to continue its recursive subdivision long after the error is sufficiently small. It is advisable to set a limit on the size of the smallest subdivision. Equation 3.29 shows how to set this limit. In Equation 3.29, the minimum subdomain size, h_{sub}, should be adjusted as h, δ, or k is changed so that the error remains below an acceptable level.

3.2.4 Separation of error factors

Adaptive integration will be advantageous over fixed order integration methods for more accurate and efficient fracture simulations employing cohesive zone models, primarily because it separates the two main factors that affect the error. In Section 3.2.1 two distinct factors of integration error in calculating the forces along the interface were delineated. One is the integration error generated by the integration algorithm and the other is generated by the ratio of the mesh size h to the process zone size, which is proportional to δ. Thus one can reduce the integration error by increasing the order of the integration algorithm or by increasing δ.

Combining Equations 3.15 and 3.29, the integration error relative to the output f_{solid} is

$$O\left(\frac{\Delta f_{interface}}{f_{solid}}\right) = O\left(\left(\frac{h_{sub}}{h}\right)^{2+k}\right) O\left(\frac{1}{(p+1)^2}\right) O\left(\frac{G_c}{Ed}\right) O\left(\frac{h}{\delta}\right). \tag{3.33}$$

For a non-adaptive algorithm, 3.33 reduces to

$$O\left(\frac{\Delta f_{interface}}{f_{solid}}\right) = O\left(\frac{1}{(p+1)^2}\right) O\left(\frac{G_c}{Ed}\right) O\left(\frac{h}{\delta}\right). \tag{3.34}$$

For fixed order integration, the integration error is not only dependent on the order of integration, but is also strongly coupled to the mesh size. One finds that for a fixed

integration order, the interpolation error decreases as the mesh is refined. The integration order needed highly depends on δ. As δ gets smaller with respect to h, a higher order integration (and adaptive integration as well), will be required as the traction distribution along the interface gets closer to an impulse. For any given δ, a high enough integration order will make the integration error arbitrarily small.

In adaptive integration, the domain subdivisions will ensure that the interpolation error remains fixed and will take place only in those regions where it is necessary. For instance, the expensive integration would only be required in an element that contains a derivative discontinuity, likely near the crack tip. In a three-dimensional model the increase in computation time would be proportional to the total length of all crack fronts rather than the total crack area as in a fixed order integration scheme. Likewise in a two-dimensional model, the increase in computation time would be proportional to the number of crack tips rather than the total crack length.

Thus the dominant error would be that associated with mesh refinement. Nonetheless, with adaptive integration, mesh refinement can be effectively separated from integration accuracy. Examples in Section 5.1 will show in what regimes and conditions adaptive integration provides a significant benefit over higher order integration.

3.3 Conclusion

Nearly all interface constitutive models have discontinuous higher derivatives, which lead to integration errors with algorithms that are based on polynomial approximations. These integration errors can lead to significant error in the overall simulation or, equivalently, require a very fine mesh to achieve the desired accuracy. A length scale parameter, $\frac{\delta}{h}$, was presented which characterizes the sensitivity of the model to the integration algorithm. Improving the integration accuracy through the use of an adaptive algorithm restores the finite element interpolation error as the dominant error source.

Chapter 4

Multiple Length Scale Finite Element Method

4.1 Introduction

The multiple length scale finite element method (MLSFEM) is a method for deriving a system of finite element equations with degrees of freedom and interpolation functions on more than one length scale. This multi-length scale representation of the fields can give great computational advantages. Of particular importance for parallel compution is the potential for greatly reduced system bandwidth, which decreases the need for interprocessor communication. This is possible because only large-scale field information is passed between processors. With MLSFEM, some accuracy is lost in comparison to a monolithic fine mesh. However, for many simulations this lost accuracy is minimal, and the improved computational efficiency more than makes up for this disadvantage.

4.2 MLSFEM Formulation

A typical displacement-based finite element formulation is derived from a potential functional, Π, which is a functional of the diplacement fields. The displacement fields, u, are

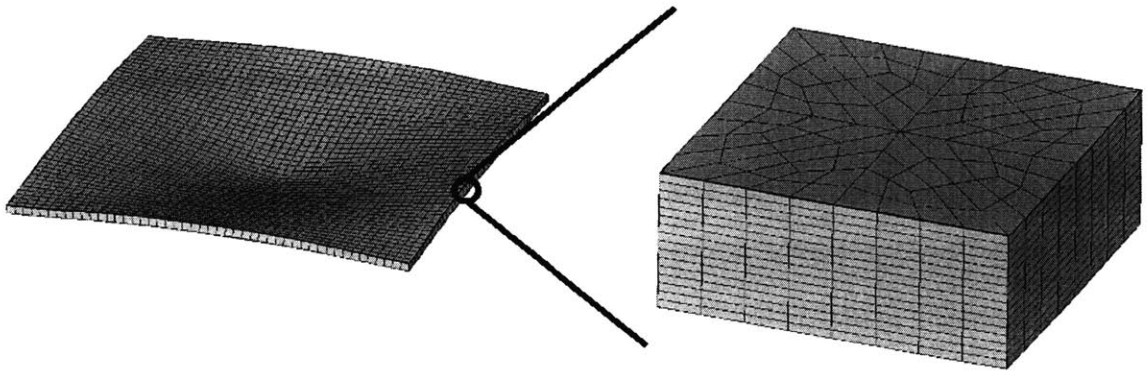


Figure 4-1: Multiple length scale finite element method schematic. An assembly of subelements is used to refine the global field of a superelement.

determined by the shape functions, n_i, and the degrees of freedom, q_i.

$$u = n_i q_i \tag{4.1}$$

This functional is minimized with respect to the degrees of freedom.

$$\delta\Pi = \frac{\partial \Pi}{\partial q_i}\delta q_i = 0 \tag{4.2}$$

Because the variations δq_i are arbitrary, a set of i equilibrium equations must be satisfied.

$$\frac{\partial \Pi}{\partial q_i} = 0 \tag{4.3}$$

In the two-scale MLSFEM formulation, the displacement fields equal to the sum of a local (u) and a global (U) displacement field. These two fields are determined by both global (Q_j) and local (q_i) degrees of freedom.

$$u_{total} = u + U = N_j Q_j + n_i q_i \tag{4.4}$$

The equilibrium equations are therefore

$$\frac{\partial \Pi}{\partial Q_j} = \frac{\partial \Pi}{\partial q_i} = 0. \tag{4.5}$$

4.2.1 Independence of the Fields

A constraint must be applied to make the fields for the two scales independent. Otherwise, the system matrices would be singular. Independence of the two fields is guaranteed if, and only if,

$$a_1 u + a_2 U \neq 0 \tag{4.6}$$

for any $a_k \neq 0$, over the entire domain Ω over which u and U are defined. In Equation 4.6, it is assumed that the individual shape functions within u and U are independent, as they would be in any ordinary finite element model.

Equation 4.6 is necessary and sufficient for independence of the degrees of freedom. However, it is much more convenient to enforce an equality than an inequality as a relationship between the degrees of freedom.

One way of meeting the condition in Equation 4.6 is to take inner products with each of the displacement fields over some part, Ω_m, of the domain Ω. This generates a system of two equations.

$$\begin{bmatrix} \int U^2 d\Omega_m & \int U u d\Omega_m \\ \int U u d\Omega_m & \int u^2 d\Omega_m \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} \neq \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \tag{4.7}$$

In order for Equation 4.7 to hold true, the determinant of the matrix must not be equal to 0. This means that

$$\int U u d\Omega_m \neq \sqrt{\int u^2 d\Omega_m \int U^2 d\Omega_m}. \tag{4.8}$$

Because u^2 and U^2 are real-valued functions,

$$\int u^2 d\Omega_m, \int U^2 d\Omega_m \neq 0 \tag{4.9}$$

unless $U = 0$ or $u = 0$ over the entire subdomain Ω_m. Therefore, if the two fields are required to be orthogonal over the domain Ω_m,

$$\int U u d\Omega_m = 0, \tag{4.10}$$

it is a sufficient, albeit not strictly necessary, condition for independence.

The domain of orthogonality, Ω_m, can be as small as two distinct points. However, in practical applications it is desirable that it be a larger fraction of the domain in order to ensure that 4.9 is satisfied. For the implementation used for this work, the fields are orthogonalized over the entire domain ($\Omega_m = \Omega$).

4.3 MLSFEM Implementation

The use of a displacement field that is the sum of a global and a local field is a common theme among superposition models. Equation 4.4 is typically implemented by defining superelements, whose interpolations form the global field, and subelements, whose interpolations form the local field. MLSFEM is unique in its implementation of the summed fields.

The implementation of this model in a finite element formulation is greatly simplified by defining the displacement fields within the subelements to be the sum of the two fields, rather than the local field only.

$$\tilde{u} = U + u = \tilde{n}_k \tilde{q}_k \tag{4.11}$$

The finite element equilibrium equations (4.5) become

$$\frac{\partial \Pi}{\partial q_i} = \frac{\partial \tilde{q}_k}{\partial q_i} \frac{\partial \Pi}{\partial \tilde{q}_k} = 0 \tag{4.12}$$

and

$$\frac{\partial \Pi}{\partial Q_j} = \frac{\partial \tilde{q}_k}{\partial Q_j}\frac{\partial \Pi}{\partial \tilde{q}_k} = 0. \tag{4.13}$$

These products involve a $i \times k$ matrix $\frac{\partial \tilde{q}_k}{\partial q_i}$ and a $j \times k$ matrix $\frac{\partial \tilde{q}_k}{\partial Q_j}$, which concatenated together form a square $(i+j) \times k$ Jacobian matrix ($k = i+j$). The Jacobian is nonsingular if there is a unique mapping between the two sets of degrees of freedom. The equilibrium equations then allow the problem to be solved as a single-scale finite element model in the new degrees of freedom.

$$\frac{\partial \Pi}{\partial \tilde{q}_k} = 0 \tag{4.14}$$

The local fields are coupled to each other through the orthogonalization constraint (Equation 4.10), which is expressed with the new degrees of freedom.

$$\int (N_j Q_j)\left((\tilde{n}_k \tilde{q}_k) - (N_j Q_j)\right) dV = 0 \tag{4.15}$$

In matrix notation, Equation 4.15 is

$$Q^T B\tilde{q} - Q^T A Q = 0, \tag{4.16}$$

in which

$$A_{mj} = \int N_m N_j dV \qquad B_{mk} = \int N_m \tilde{n}_k dV \tag{4.17}$$

It is convenient for both accuracy and computational efficiency to linearize Equation 4.16 with respect to Q_j. This yields a linear system of equations that can be implemented in finite element code as multiple point constraints (MPCs.)

$$A_{mj} Q_j = B_{mk} \tilde{q}_k \tag{4.18}$$

In this way, a model with two length scales can be solved as two single-scale models. The interaction between the two is accomplished with multiple point constraints. This al-

lows a MLSFEM developer to use existing finite element source code to speed the software development.

4.4 Computer Implementation

The formulation described by Equations 4.11 through 4.18 allows a relatively simple implementation with existing finite element software. The superelements and the subelements are each modeled with a serial, single length scale finite element simulation process. Constraints on the subelement degrees of freedom are applied with MPCs in the subelement simulation process. The superelement degrees of freedom needed for the MPCs are passed to the subelement process with the Message-Passing Interface (MPI) library. More than one of these patches of subelements may be simulated, each with a separate process. These processes may be run in parallel, on separate processors, or in turn for each time step on a single processor.

It should be noted that the constraint matrices of Equation 4.18 place a uniquely stringent demand on a finite element program's MPC implementation. It is the author's experience that some finite element code is written with the assumption that only a few MPCs will be used at a time. This can cause problems with memory use and computation time when a sizable patch of subelements is used. Many finite element codes might need to be improved to increase the efficiency of the MPC implementation. The user should be aware of this consideration and determine whether a particular finite element program can adequately handle the number of MPCs necessary to model a patch of the desired number of subelements.

Chapter 5

Results

5.1 CZM Model Validation

5.1.1 Double Cantilever Beam Simulation

In order to illustrate the effect of the interface integration algorithm under static conditions, a double cantilever beam (DCB) is used as a typical crack simulation model. This application is common in validation simulations as well as in experiments to determine material fracture properties. Two halves of a strip of material are pulled apart from one end, as shown in Figure 5-1. A crack is initiated on the plane of symmetry, and displacement-controlled loading is used to propagate a stable crack quasi-statically.

The cubic cohesive zone model (Figure 3-1(a) and Equation 3.5) is used for the DCB simulations, with the following parameters: E for the beam is 400 GPa, ν (Poisson's ratio) is .19, E_i for the interface is 2 GPa, and δ is 1 μm. The DCB has a 2 mm initial crack along the centerline. The finite element model is displacement-based, and it uses eight-node two dimensional plane strain elements for a quadratic displacement field within each element.

Figure 5-2 shows the reaction force at the tip of the DCB, plotted against the displacement at the tip.

For a Bernoulli-Euler beam, the analytical solution of the force applied at the beam tip

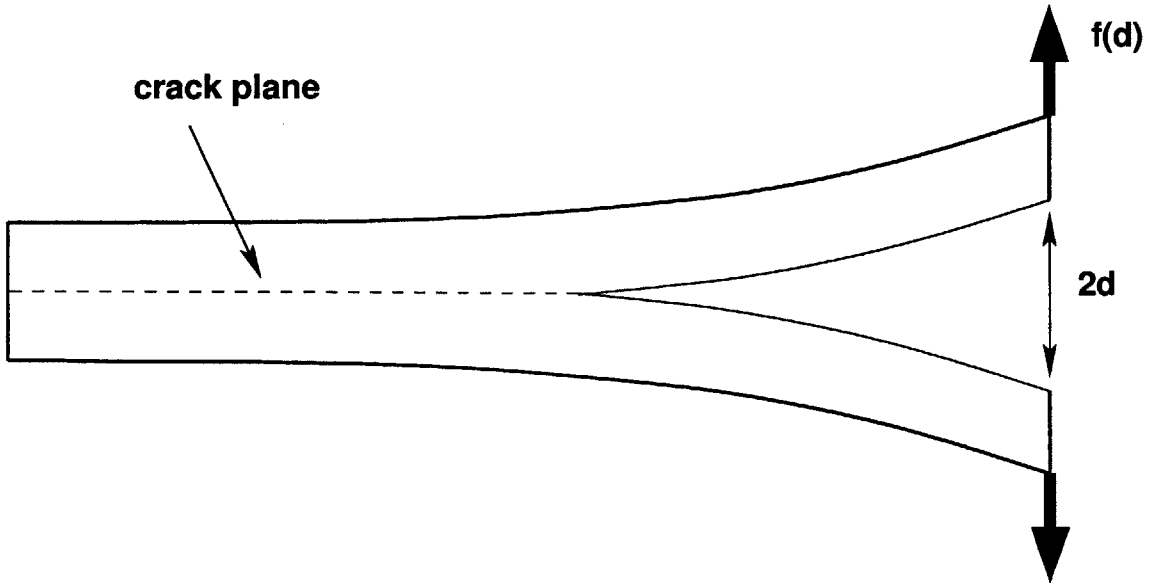


Figure 5-1: Schematic diagram of double cantilever beam

as a function of the beam tip displacement is

$$\frac{f(d)}{b} = 2^{-\frac{1}{2}} 3^{-\frac{3}{4}} (aG_c)^{\frac{3}{4}} E^{\frac{1}{4}} \frac{1}{\sqrt{d}}. \tag{5.1}$$

This solution is derived in Appendix A, but note that it may appear elsewhere. In Equation 5.1, b is the width, a is the height of each half of the split DCB, d is the displacement at the tip, and $f(d)$ is the tip reaction force. This analytical solution is plotted in Figure 5-2 for comparison to the numerical results.

Results in Figure 5-2 clearly show the difference in the performance of each integration algorithm. The consequence of error in the integration can be visualized in two ways in Figure 5-2: solution jumps and convergence failure. For the fixed order integration schemes, if the mesh is too coarse for the integration order, then erroneous solution "jumps" will occur in the DCB tip load $f(d)$ vs. displacement curve d [7], as shown by the dashed line in Figure 5-2) In the simulations performed for this paper, these jumps usually caused the solver to fail to converge. The greatest component of this error in the loading force is the error in the interface integration. For instance, for the fine mesh, $\frac{\delta}{h} = 0.005$, the zero

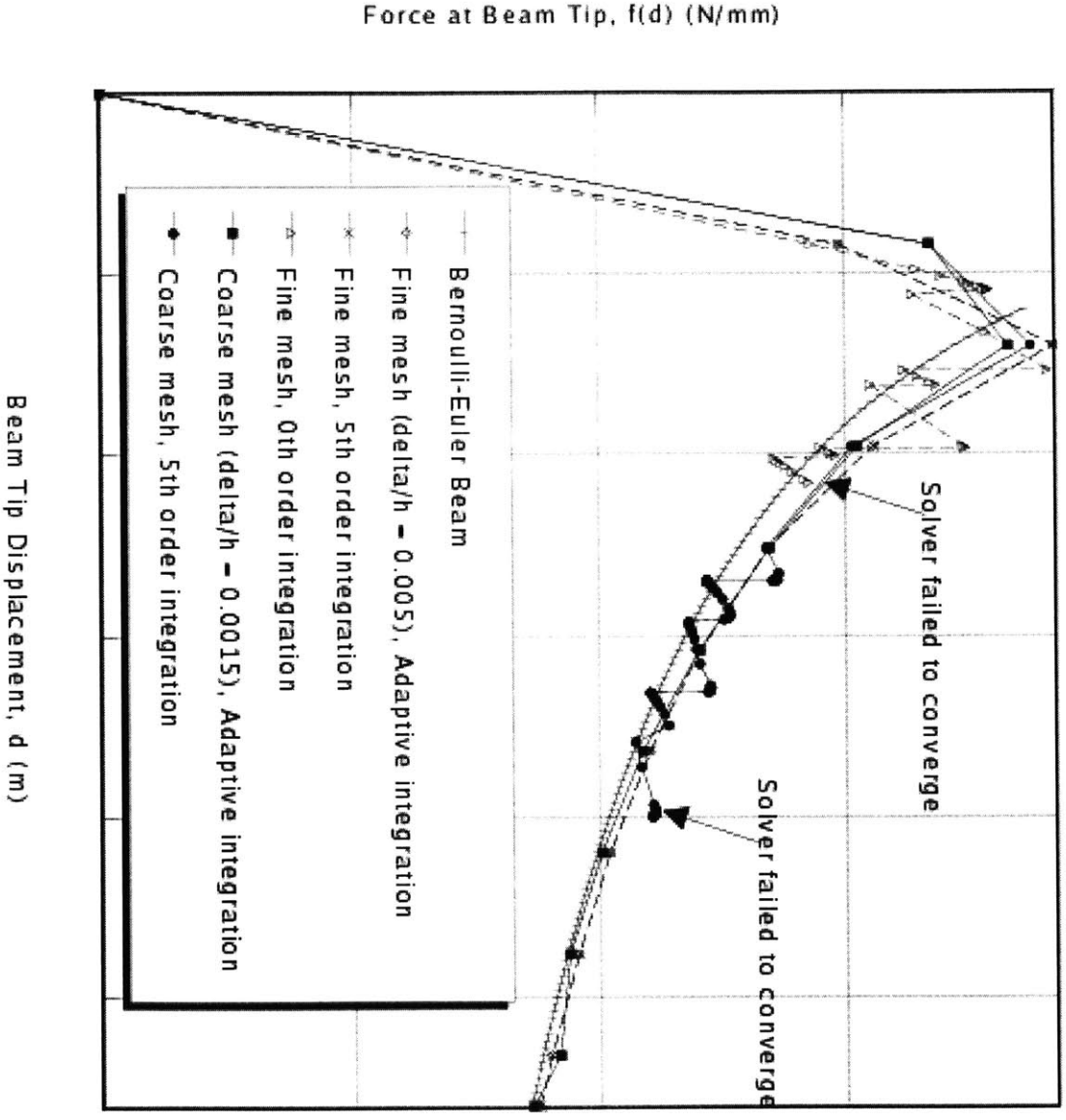


Figure 5-2: DCB with varied mesh and integration order

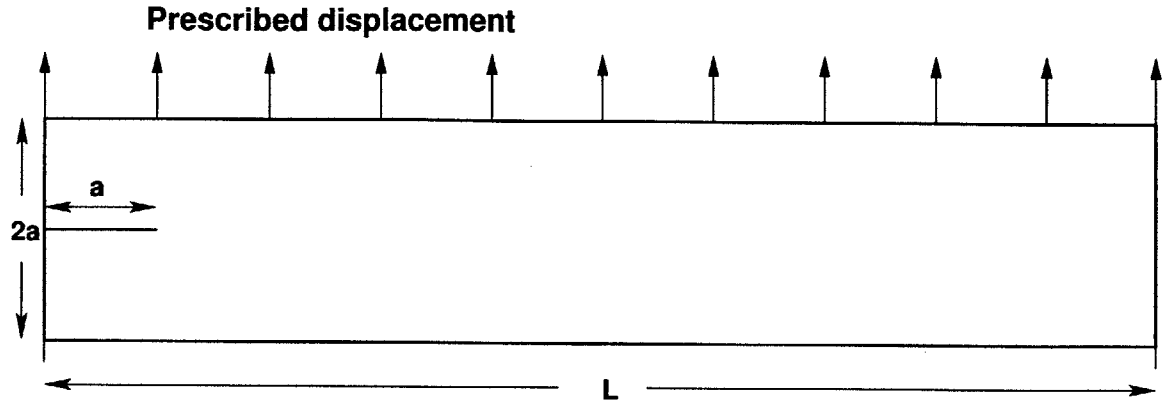


Figure 5-3: Schematic diagram of elastic strip

order integration scheme causes solution jumps and eventually convergence failure by d = 5.5 μm. Figure 5-2 shows that the use of a higher order integration algorithm, namely fifth order integration, eliminates the solution jumps for a fine mesh. However, if the mesh is made more coarse, $\frac{\delta}{h}$ = 0.0015, eventually the 5th order algorithm fails and the solution jumps return. With adaptive integration, however, the integration error is again reduced to give a smooth, physically correct solution for both fine and coarse meshes.

5.1.2 Dynamic elastic strip

Dynamic elastic strip simulations

In this section the relationship between the $\frac{\delta}{h}$ ratio and the integration algorithm is further explored, but under dynamic conditions. This case is demonstrated with an example of a crack propagating dynamically through an elastic strip, as shown in Figure 5-3.

An analysis of the dynamic elastic strip as used by Geubelle and Baylor [17] is performed. The properties of the isotropic elastic strip are E = 3.24 GPa, ν = .35, and ρ = 1130 kg/m^3. Following Geubelle and Baylor, a bilinear interface was employed with properties T_{max} = 324 MPa and G_c = 352.3 J/m^2. The strip has a total height of $2a$ = 0.2 mm and a length of L = 2 mm. An initial crack of length a = 0.1 mm is present at one end of the strip, along the center line. The strip is loaded with a prescribed displacement of 0.0032

mm. The analysis is a two dimensional plane strain simulation, explicitly integrated in time. The finite element model uses eight-node quadratic displacement elements.

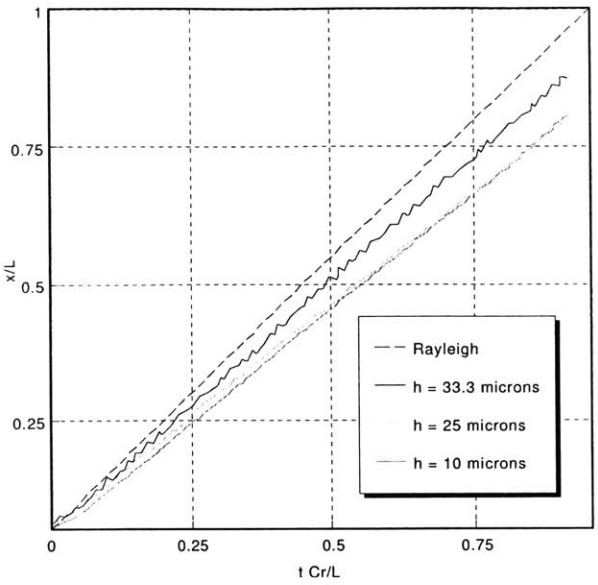
Figure 5-4 shows the position of the crack tip, normalized by L, as a function of normalized time for both large and small values of $\frac{\delta}{h}$. Also shown is the Rayleigh wave speed, $C_r = 940$ m/s, which is the theoretical limit for Mode I cracks[36]. All crack propagation velocities are under the Rayleigh wave speed. As in the work by Geubelle and Baylor [17], time is normalized by the ratio of the ratio of the Rayleigh wave speed to the strip length, $\frac{C_r}{L}$.

Because cohesive zone models prescribe a continuous progression of local interface failure from initiation to complete separation, the location of the crack tip must be defined. For the purpose of comparing integration algorithms, the crack tip position is defined as the distance from the end of the strip to the point on the crack line where the interface displacement jump is equal to the critical displacement jump ($\frac{u_{cracktip}}{\delta} = v_{cracktip} = 1.$)

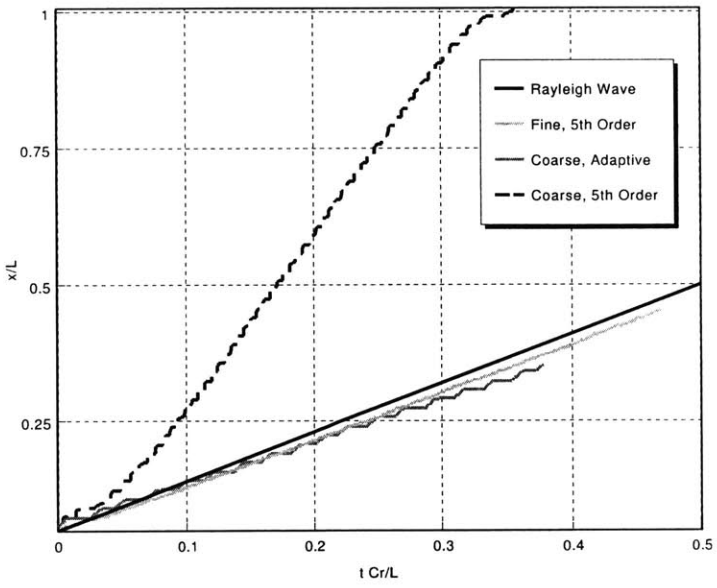
In Figure 5-4(a), δ is constant and $\frac{\delta}{h}$ ranges from 3.624×10^{-2} to 1.208×10^{-1} for the most coarse mesh to the most fine mesh as it is in the paper by Geubelle and Baylor [17]. As in the work of [17], all of the simulations in Figure 5-4(a) used a fifth order Gauss integration algorithm, which appears to be sufficient for carrying out the simulations to completion at this ratio of $\frac{\delta}{h}$. It can be seen that the crack propagation velocity seems to converge by $\frac{\delta}{h} = 4.832 \times 10^{-2}$. It also appears to converge from above; that is, the propagation velocity is faster with a coarser mesh.

Neither of these observations, that is convergence from above and convergence by $h = 25\,\mu$m, were present in Geubelle's simulations. Geubelle reported that the crack propagation velocity converged more slowly, and that more coarse meshes underestimated the velocity. The reasons for the difference are not yet clear. It is conjectured that the higher wave velocity observed here in the coarser meshes is due to the fact that displacement-based finite element models tend to overestimate structural stiffness.

Figure 5-4(b) shows a test of the capabilities of adaptive integration. In Figure 5-4(b), the same mesh is used but $\frac{\delta}{h}$ is 3.624×10^{-4}, corresponding to a δ two orders of magnitude



(a)



(b)

Figure 5-4: Crack tip location vs. time. The crack tip location (x) is normalized by the length of the strip (L), and the time (t) is normalized by the strip length and the Rayleigh wave speed (C_r).

smaller or a strip two orders of magnitude bigger.

In this case the fifth order integration gives a crack propagation velocity much larger than the Rayleigh wave speed, which is an unphysical result. This clearly erroneous solution is caused by the interaction of the inaccurate integration of the interface and of a low energy hourglass mode in the adjacent solid elements. This interaction results in the formation of secondary cracks ahead of the primary crack tip, allowing the crack to propagate supersonically. When the same simulation is carried out with adaptive interface integration, the crack propagates at a physically reasonable velocity. This propagation velocity is quite close to the velocity given by a fifth order integration algorithm used by a refined mesh, *i.e.*, a smaller h, for which $\frac{\delta}{h} = 1.208 \times 10^{-3}$. Again, the adaptive integration allows the use of a coarser mesh (smaller $\frac{\delta}{h}$) than is possible with a fixed order integration algorithm.

5.1.3 Effect on computation time

One important test of the usefulness of an algorithm is its relative cost in terms of computation time. Recall that there are two sources of error, both of which can be reduced by an increase in integration order and refining the mesh size, but at the expense of computation time. Any increase in computation time due to an increase in the integration order must be less than that of using a refined mesh for it to be useful. Figure 5-5 shows the total CPU time required to run the quasi-static DCB simulations of Section 5.1.1, with varying interface integration orders and mesh sizes. As in Section 5.1.1, the cubic cohesive zone model and associated parameters were used. The mesh size is indicated by the ratio $\frac{\delta}{h}$ on the abscissa, where h is the element length. Each point represents a simulation that was successfully completed (*i.e.*, without solution jumps) and in agreement with Equation 5.1. An Intel Pentium 1.8 GHz computer with a 133 MHz front side bus was used for all simulations.

Figure 5-5 shows that, for the fine meshes (large $\frac{\delta}{h}$), the total CPU time rises slightly with increasing integration order. Total CPU time also grows with $\frac{\delta}{h}$. Both of these effects are to be expected, as in either case the number of integration points is increased. For a

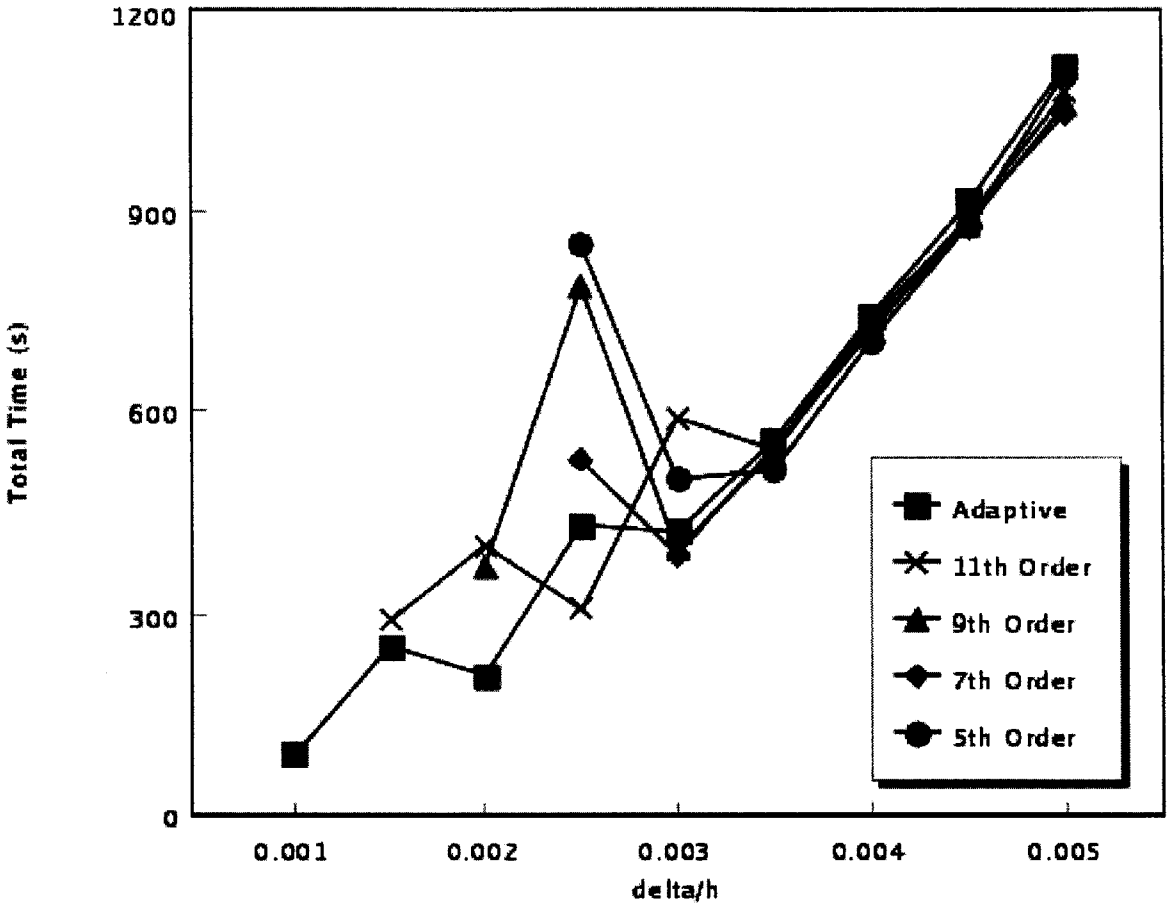


Figure 5-5: Total CPU time vs. mesh refinement, for various integration orders and adaptive integration

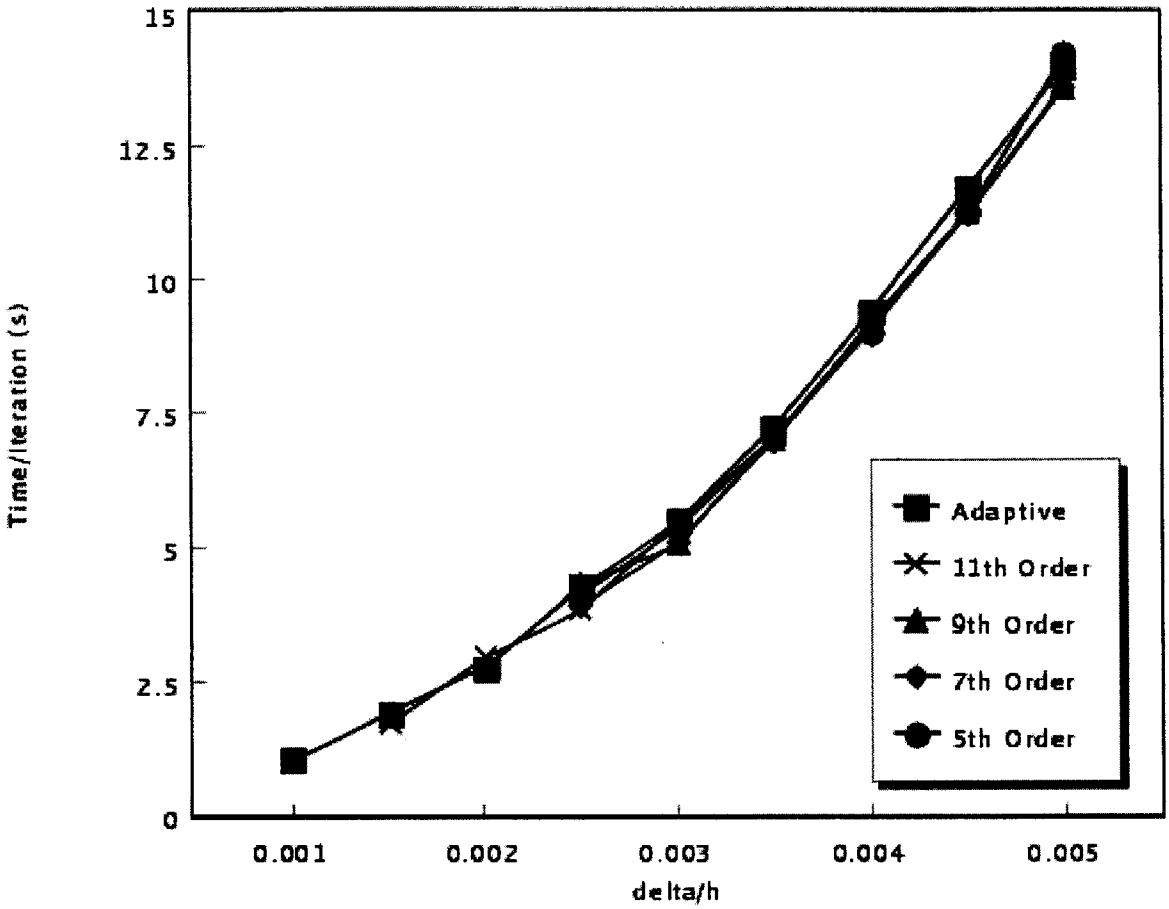


Figure 5-6: CPU time per iteration vs. mesh refinement

given integration order, as $\frac{\delta}{h}$ decreases a point is reached where the solver requires more iterations to reach convergence and the total time becomes erratic. In many cases the total time increases, sharply reversing its trend. As $\frac{\delta}{h}$ continues to decrease, the integration error becomes so large that the solver completely fails to converge. For higher integration orders, lower values of $\frac{\delta}{h}$ may be reached. However, for any fixed order algorithm, as $\frac{\delta}{h}$ is lowered the solver will eventually fail. In contrast, an adaptive integration algorithm will always be able to integrate to an accuracy sufficient for convergence, constrained only by the arithmetic precision of the computer.

In order to show the effect on computation time without including the effect of differing convergence efficiency, Figure 5-6 shows the average CPU time per iteration versus the

mesh refinement for the same set of simulations. The time per iteration shows the same trends as the total time. It is clear from Figure 5-6 that the main source of variability in the solution time is the increased number of iterations for the coarser meshes. The increased iterations are due to the greater integration error for coarser meshes with fixed order integration algorithms.

In summary, adaptive integration allows the use of a coarse mesh, for which fixed order integration will either fail to converge, give erroneous results if convergent, or be more expensive. While the cost of the interface integration will vary from one finite element package to another, for most simulations it is likely that the possibility of using a more coarse mesh will more than make up for the increased time required for adaptive integration near the crack tip.

5.2 MLSFEM Model Validation

In order to test the MLSFEM program, a simple example is used that demonstrates the ability of the subelements to satisfy the global fields in an average sense. At the same time, local refinements of the global field can also be seen.

In Figure 5-7, two subelements that compose a single superelement are shown. The superelement is an eight-node brick with a linear variation in the displacement fields. The two subelements are twenth-node quadratic bricks, each half the height of the superelement. A wireframe depicts the undeformed configuration of the two subelements. There are no boundary conditions applied directly to these subelements. Instead, a nodal force is applied to one of the superelement nodes, while the others have their displacements constrained by homogeneous Dirichlet boundary conditions. These boundary conditions are transferred to the subelements through the constraint equations.

It is evident that the resulting displacement fields in the subelements, as shown in Figure 5-7, approximates the linear global field. It can also be seen, however, that the displacement fields have a small-scale refinement of the linear global field. This is most easily

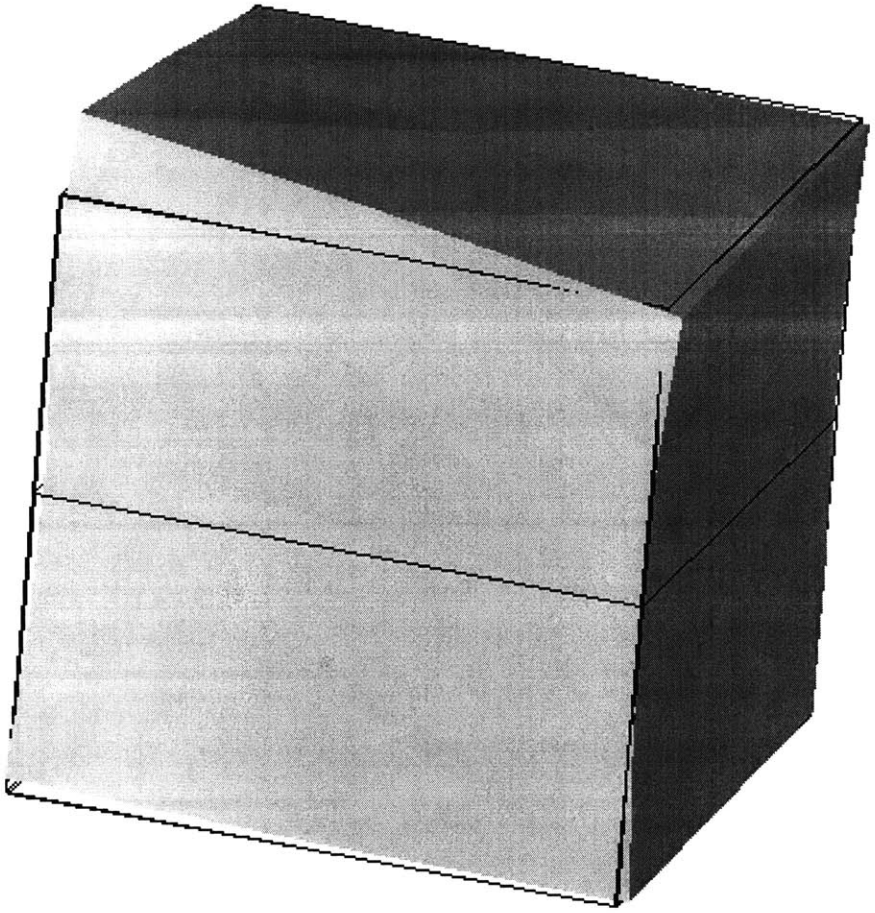


Figure 5-7: Example superelement. The two subelements that make up the superelement deform in response to a force applied to a supernode.

observed when the edges of the deformed subelements are compared to the straight lines of the wireframe. It is exactly this type of small-scale refinement that is needed order to allow for damage simulation on the local scale.

5.3 Laminate Impact Results

As a culmination of the models used in Sections 5.1 and 5.2, a laminated composite plate was simulated using cohesive zone elements among a group of subelements. The plate to be modeled was a two-ply, 0/90 graphite epoxy laminate, measuring twelve inches by twelve inches by 0.25 inches. Each ply was 0.125 inches thick. An example of the input file of the plate is listed in Appendix C.3.1. The material properties of the plies are listed in Appendix C.3.2.

For the CZM used in this laminate, the damage parameter, λ, is defined as

$$\lambda = \max\left(|v|, \lambda_{previous}\right). \tag{5.2}$$

The damage function, F, is defined as

$$F = \begin{cases} (1-\lambda)^2 & 0 \le \lambda \le 1 \\ 0 & \lambda > 1 \end{cases}. \tag{5.3}$$

Because λ is initialized to 0, F need not be defined for $\lambda < 0$.

The global length scale of the plate was modeled with linear, eight-node brick elements. Each global element was a cube with 0.25 inch edges. The plate was subjected to a transverse force at the center to simulate an impact event. No other boundary conditions are applied to the plate; that is, the plate is a free, unsupported structure. The impact force was applied as a triangular spike with a duration of 2×10^{-4} seconds, as indicated in the input file (Appendix C.3.1.) A one inch by two inch region at the center of the plate was modeled with a group of subelements, as shown in Figure 5-8.

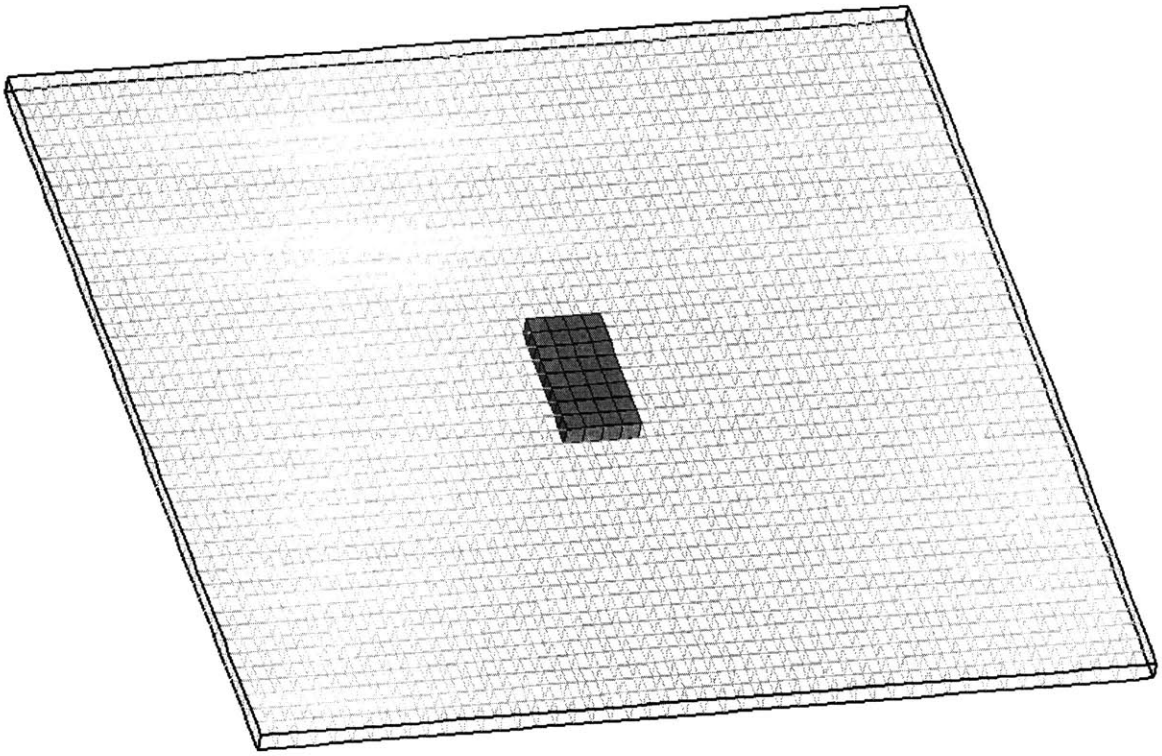
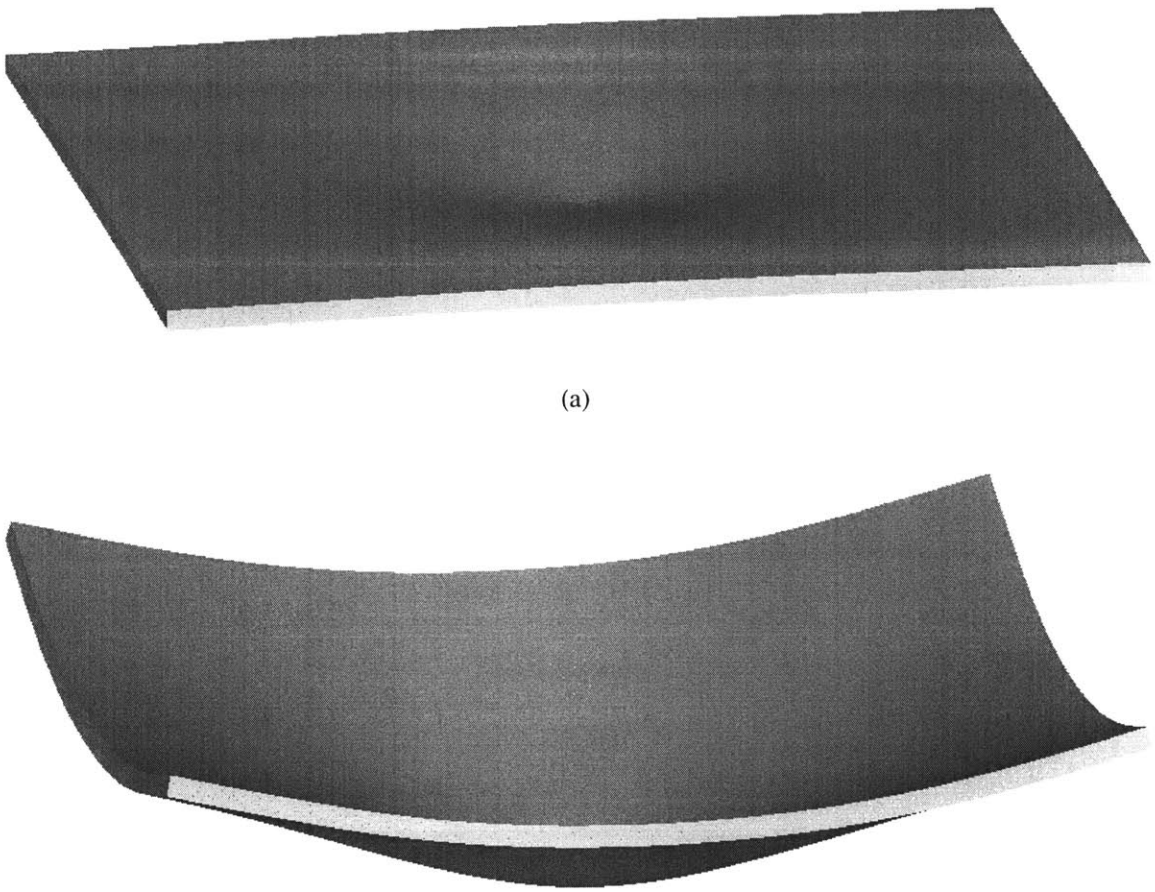


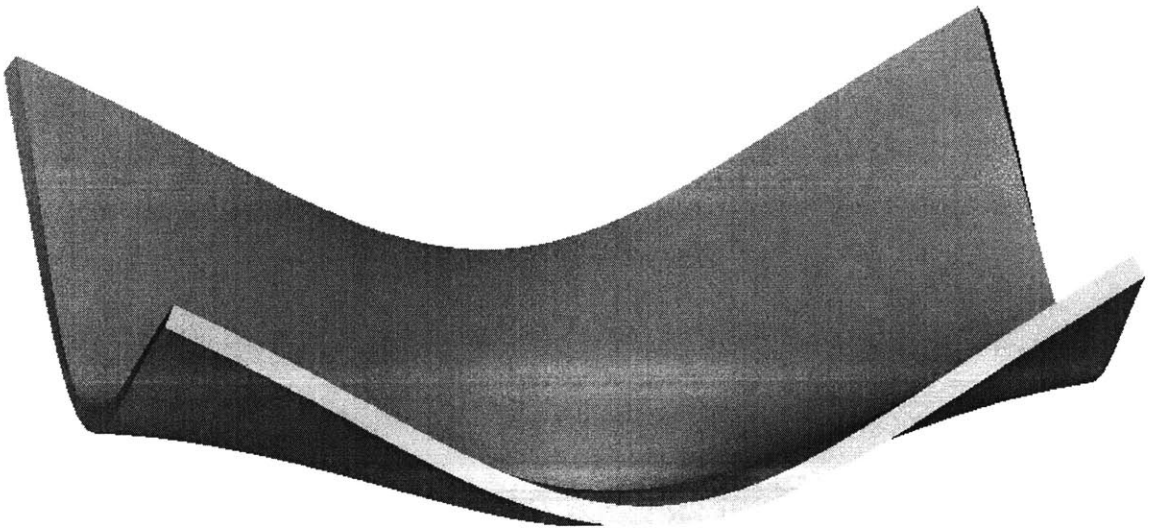
Figure 5-8: Superelements shown within the full plate mesh.



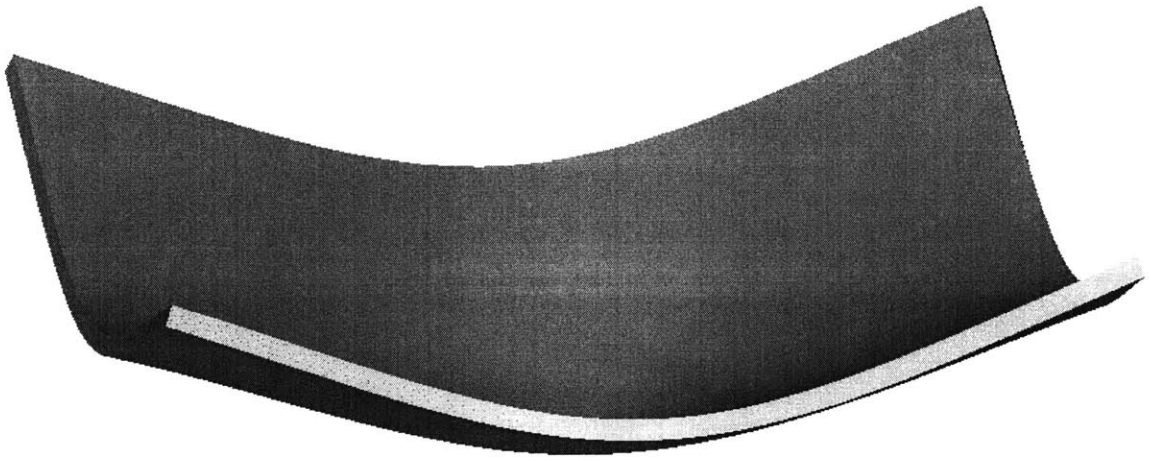
(a)

(b)

Figure 5-9: Plate displacement response. Only the global displacement field is shown. (a) time = 2.18×10^{-4} seconds. (b) time = 5.18×10^{-4} seconds.



(a)



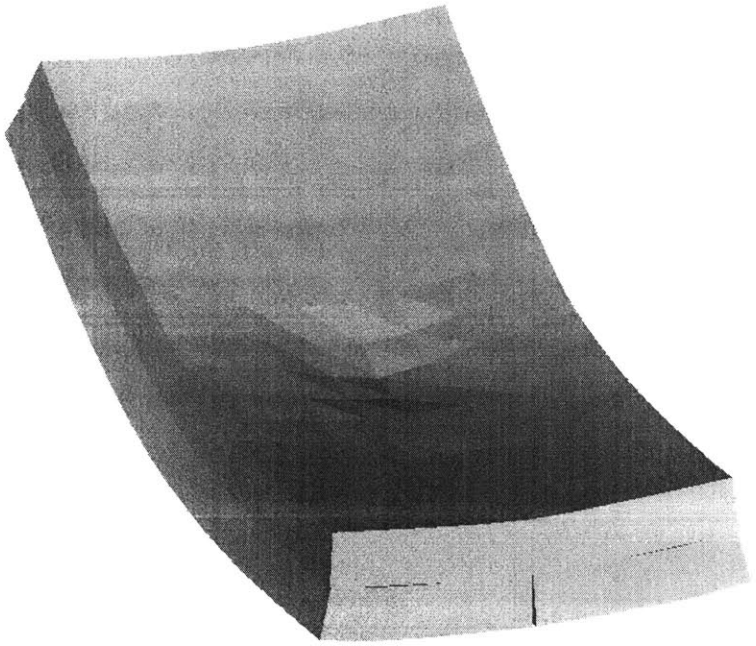
(b)

Figure 5-10: Plate displacement response (continued.) Only the global displacement field is shown. (a) time = 7.18×10^{-4} seconds. (b) time = 1×10^{-3} seconds.

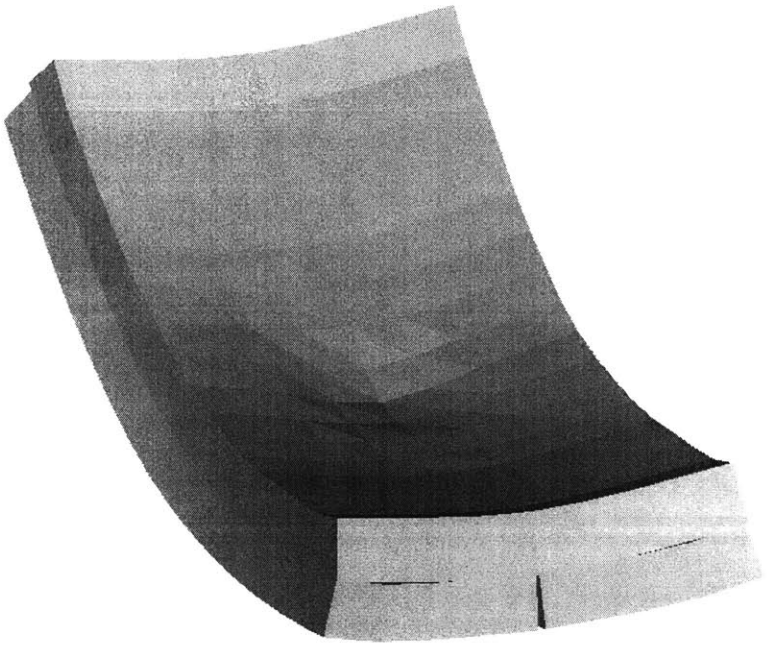
Figures 5-9 and 5-10 show the response of the global degrees of freedom of the plate, with the displacements applified by a factor of five. It can be seen that the wave propagation outward from the impact site and the subsequent bending oscillations of the plate are captured by the global length scale model of the plate.

Figures 5-11 and 5-12 show the subelements from above in the region surrounding the impact site, where damage is expected to initiate. The displacements are amplified by a factor of ten. The location of impact and the bending of the plate in response can be seen clearly. As with the example in Section 5.2, no boundary conditions are applied to the subelements; the subelements respond to the global degrees of freedom via the constraint equations.

Figures 5-13 and 5-14 show the same configurations of the subelements as in Figures 5-11 and 5-12, as viewed from below. As the bending deformation of the plate increases, a ply split in the bottom ply is observed to initiate at the center and propagate outward. A delamination, initiated by the ply split, is also observed between the two plies in the form of a discontinuity of in-plane displacement. This sequence of failure modes is the same as that observed in experimental studies. (See, for example, Chang and Choi[8].) Thus the model is shown to qualitatively predict the nature of the micromechanical damage and how the ply split interacts with the delamination.

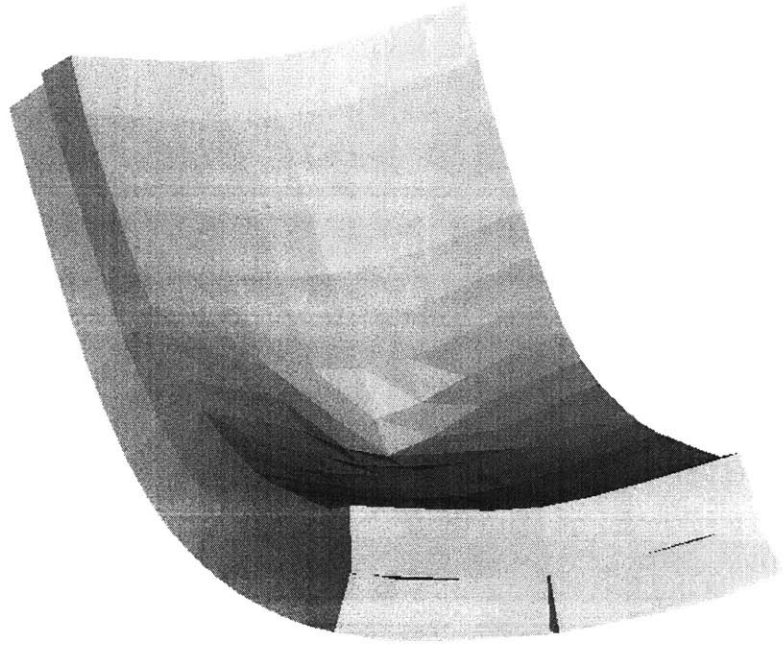


(a)

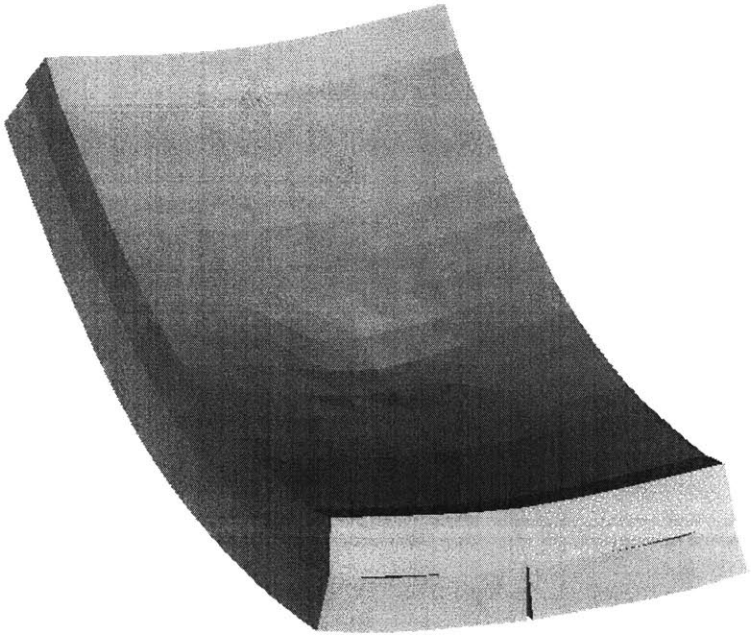


(b)

Figure 5-11: Superelements in damage zone. The view is from the impacted side of the plate. (a) time = 2.18×10^{-4} seconds. (b) time = 5.18×10^{-4} seconds.

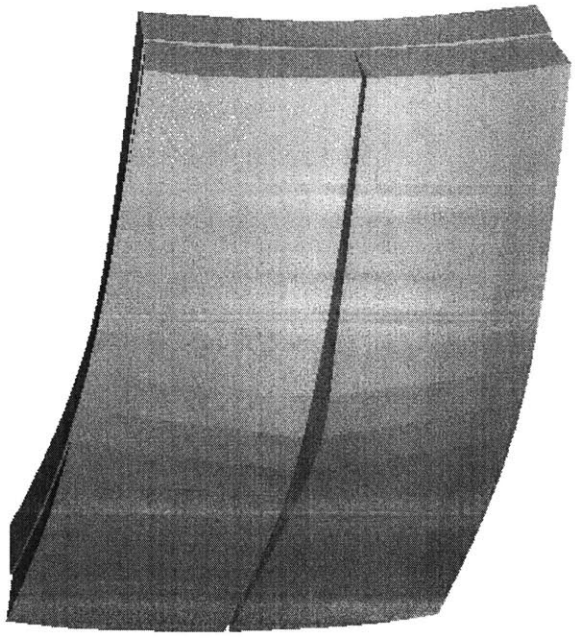


(a)



(b)

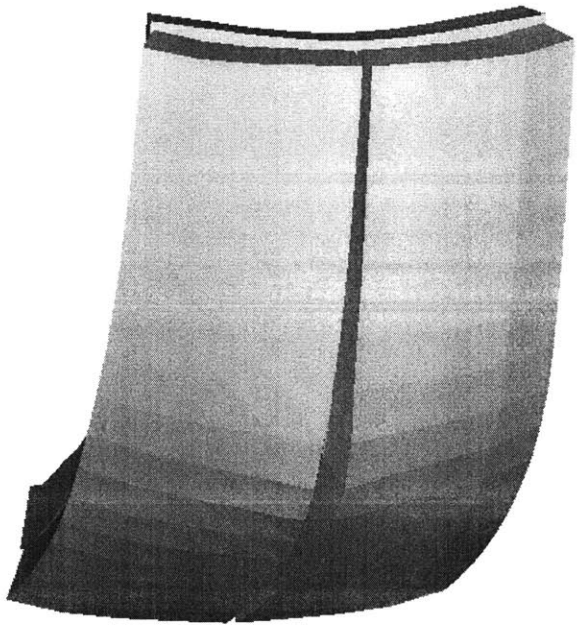
Figure 5-12: Superelements in damage zone (continued.) The view is from the impacted side of the plate. (a) time = 7.18×10^{-4} seconds. (b) time = 1×10^{-3} seconds.



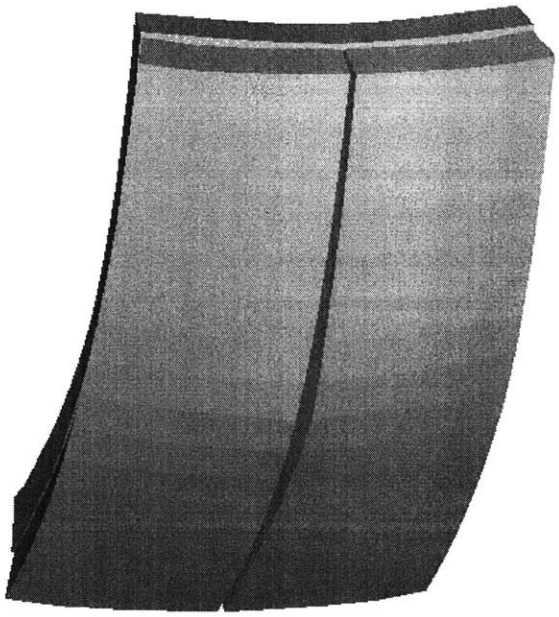
(a)

(b)

Figure 5-13: Superelements in damage zone. The view is from the side opposite the projectile. (a) time = 2.18×10^{-4} seconds. (b) time = 5.18×10^{-4} seconds.



(a)



(b)

Figure 5-14: Superelements in damage zone (continued.) The view is from the side opposite the projectile. (a) time = 7.18×10^{-4} seconds. (b) time = 1×10^{-3} seconds.

Chapter 6

Conclusions

6.1 Project Summary

In order to model impact damage in laminated composite plates, two key modeling approaches were illustrated. Cohesive Zone Models were identified as a way of simulating the initiation and propagation of the various forms of interacting cracks in the composite. A Multiple Length Scale Finite Element Method was proposed to meet the challenge of performing the simultaneous simulation of structural response on the structural and ply-thickness length scales.

While CZMs are a versatile way to model various types of fracture, numerical pitfalls have made them difficult to implement in finite element models. These numerical problems were shown to result from inaccurate integration of the CZM properties over the interface element. An adaptive integration technique was applied that alleviated the numerical problems and increased the computational efficiency of the CZMs. The improved convergence of the CZM with this adaptive integration algorithm was shown with both quasi-static and dynamic fracture examples.

The MLSFEM was shown to capture effectively the large scale structural behavior of the laminate with an acceptable degree of accuracy. The small scale fields were resolved in the region most likely to experience damage from the impact loading. The model simulated

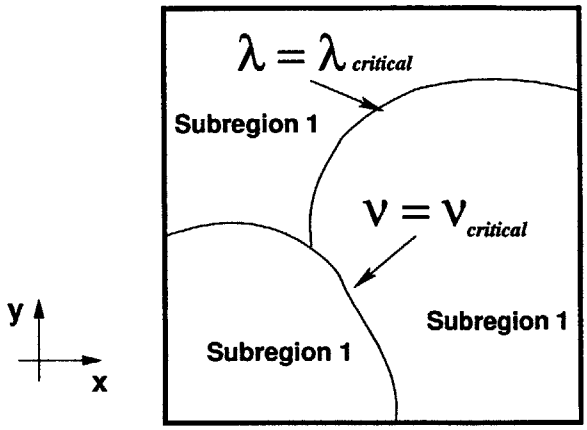


Figure 6-1: Subregions with continuous derivatives. The locus of points at which $\nu = \nu_{critical}$ or $\lambda = \lambda_{critical}$ are points at which there is a discontinuous derivative in the integrand.

the formation of ply splits, which in turn initiated delamination between the plies.

6.2 Recommendations for Future Study

6.2.1 Accurate CZM Integration

The integration algorithm described in Section 3.2.3 uses fifth- and seventh- order quadrature rules in each subdomain and a fourth difference to decide which direction to divide. These were the default given by Genz and Malik[16]. It is possible that experimentation with different combinations of quadrature rules and division criteria could lead to a more effective adaptive integration algorithm.

The algorithm presented for adaptive integration is a very general method for evaluating the nodal forces and tangent stiffness matrix methods. It can be implemented as a "black box" that can be used with any type or order of displacement interpolation or form of the CZM. This level of generallity is often desirable.

In many cases, however, a finite element code developer might find it advantageous to use an integration algorithm that is specialized to a particular CZM and displacement interpolation. Such an approach could have a higher computational efficiency than the "black

box" algorithm. For example, for a given CZM, it is possible to determine beforehand the values of ν and/or λ at which there are discontinuous derivatives. Then, using the known displacement interpolation for a given element type (e.g., linear, quadratic, etc.) it is possible to segment the domain of integration into subregions within which all derivatives are continuous, as shown in Figure . Because the integrand is of finite order and has continuous derivatives throughout each subregion, each subregion may be integrated to machine precision with a fixed order quadrature rule. The challenges of this approach to the integration problem are the difficulty of numerically integrating over oddly-shaped subregions and the fact that it would need to be reformulated for each possible combination of a CZM and displacement interpolation. Nevertheless, if an analyst has decided to make extensive use of a particular CZM and element displacemnet interpolation, such a method might be very worthwhile to develop.

6.2.2 Multiple Length Scale Finite Element Method

In the manufacture of laminated composites, there is great flexibility to vary the properties of the plies as well as the number, order, and angles of the plies in the layup. Much work could be done in further testing the application of MLSFEM to other composite systems and comparison with validation experiments. Additionally, the formulation of the model does not distinguish the thickness coordinate from the in-plane coordinates of the plate. MLSFEM is therefore not exclusively applicable to laminates and might be useful in the analysis of the fiber/matrix length scale, particulate or whisker composites, or many other composite systems.

Bibliography

[1] J. Aboudi. *Mechanics of Composite Materials: A Unified Micromechanical Approach*. Elsevier, New York, 1991.

[2] R.C. Averill and Y.C. Yip. Thick beam theory and finite element model with zig-zag sublaminate approximations. *AIAA Journal*, 34(8):1627–1632, 1996.

[3] G. Barenblatt. The mathematical theory of equilibrium crack in the brittle fracture. *Advances in Applied Mechanics*, 7:55–129, 1962.

[4] Beuth. Separation of crack extension modes in orthotropic delamination models. *International Journal of Fracture*, 77:305–321, 1996.

[5] V.V. Bolotin. Delaminations in composite structures; its origin, buckling, growth, and stability. *Composites Part B*, 27B:129–145, 1996.

[6] G.T. Camacho and M. Ortiz. Computational modelling of impact damage in brittle materials. *International Journal of Solids and Structures*, 33:2899–2938, 1996.

[7] J.L. Chaboche, F. Feyel, and Y. Monerie. Interface debonding models: a viscous regularization with a limited rate dependency. *International Journal of Solids and Structures*, 38:3127–3160, 2001.

[8] F.K. Chang and Hyung Yun Choi. A model for predicting damage in graphite/epoxy laminated composites resulting from low-velocity point impact. *Journal of Composite Materials*, 14:2134–2136, 1992.

[9] Z.Q. Cheng, D. Kennedy, and F.W. Williams. Effect of interfacial imperfection on buckling and beding behavior of composite laminates. *AIAA Journal*, 34(12):2590–2595, 1996.

[10] M. Cho and R.R. Parmerter. Efficient higher-order composite plate theory for general lamination configurations. *AIAA Journal*, 31(7):1299–1306, 1993.

[11] A. Corigliano. Formulation, identification, and use of interface models in the numerical analysis of composite delamination. *International Journal of Solids and Structures*, 30:2779, 1993.

[12] C.G. Dávila, P.P. Camanho, and M.F. de Moura. Mixed-mode decohesion elements for analyses of progressive delamination. In *42nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, pages AIAA–01–1486, 2001.

[13] D. Dugdale. Yielding of steel sheets containing slits. *Journal of the Mechanics and Physics of Solids*, 8:100–108, 1960.

[14] F. Feyel and J.L. Chaboche. FE^2 multiscale approach for modelling the elastoviscoplastic behaviour of long fibre SiC/Ti composite materials. *Computer Methods in Applied Mechanical Engineering*, 183:309–330, 2000.

[15] J. Fish and K. Shek. Multiscale analysis of composite materials and structures. *Composites Science and Technology*, 60:2547–2556, 2000.

[16] A.C. Genz and A.A. Malik. Remarks on algorithm 006: An adaptive algorithm for numerical integration over an n-dimensional rectangular region. *Journal of Computational and Applied Mathematics*, 6(4):295–299, 1980.

[17] P.H. Geubelle and J.S. Baylor. Impact-induced delamination of composites: A 2D simulation. *Composites Part B*, 29B:589–602, 1998.

[18] K.H. Lee, N.R. Senthilnathan, S.P. Lim, and S.T. Chow. An improved zig-zag model for the bending of laminated composite plates. *Composite Structures*, 15:137–148, 1990.

[19] K.H. Lo, R.M. Christensen, and E.M. Wu. A higher-order theory of plate deformation—Part 1: Homogeneous plates. *Journal of Applied Mechanics*, 44:663, 1977.

[20] K.H. Lo, R.M. Christensen, and E.M. Wu. A higher-order theory of plate deformation—Part 2: Laminated plates. *Journal of Applied Mechanics*, 44:669, 1977.

[21] J. McGee and C.T. Herakovic. Micromechanics of fiber/matrix debonding. Applied Mechanics Program Report AM-92-01, University of Virginia, 1992.

[22] M.F.S.F. Moura, J.P.M. Gonçalves, A.T. Marques, and P.M.S.T. de Castro. Prediction of compressive strength of carbon-epoxy laminates containing delamination by using a mixed-mode damage model. *Composite Structures*, 50:151–157, 2000.

[23] A. Needleman. An analysis of tensile decohesion along an interface. *Journal of the Mechanics and Physics of Solids*, 38(3):289–324, 1990.

[24] P. Raghavan, S. Moorthy, S. Ghosh, and N.J. Pagano. Revisiting the composite laminate problem with an adaptive multi-level computational model. *Composites Science and Technology*, 61:1017–1040, 2001.

[25] J.N. Reddy. A refined nonlinear theory of plates with transverse shear deformation. *International Journal of Solids and Structures*, 20:881–896, 1984.

[26] J.N. Reddy. *Mechanics of Laminated Composite Plates: Theory and Analysis.* CRC Press, Boca Raton, 1997.

[27] D.H. Robbins and J.N. Reddy. An efficient computational model for the stress analysis of smart plate structures. *Smart Materials and Structures*, 5:33–360, 1996.

[28] O. Samudrala, Y. Huang, and A.J. Rosakis. Subsonic and intersonic mode ii crack propagation with a rate-dependent cohesive zone. *Journal of the Mechanics and Physics of Solids*, 50:1231–1268, 2002.

[29] M. Di Sciuva. Development of an anisotropic, multilayered, shear-deformable rectangular plate element. *Computers and Structures*, 21(4):789–796, 1986.

[30] M. Di Sciuva. Multilayered anisotropic plate models with continuous interlaminar stresses. *Composite Structures*, 22:149–167, 1992.

[31] S.M. Spearing, P.A. Legace, and H.L.N. McManus. On the role of lengthscale in the prediction of failure of composite structures: Assessment and needs. *Applied Composite Materials*, 5:139–149, 1998.

[32] B. Storakers. *Nonlinear aspects of delamination in structural members.* 1989. In: Germain, P., Piau, M., and Caillere, D. (Eds.), Theoretical and Applied Mechanics, Elsevier Science.

[33] A. Toledano and H. Murakami. A high-order laminated plate theory with improved in-plane responses. *International Journal of Solids and Structures*, 23(1):111–131, 1987.

[34] V. Tvergaard. Effect of fibre debonding in a whisker-reinforced metal. *Materials Science and Engineering*, A125:203–213, 1990.

[35] P. Van Dooren and L. De Ridder. An adaptive algorithm for numerical integration over an n-dimensional cube. *Journal of Computational and Applied Mathematics*, 2(3):207–210, 1976.

[36] P.D. Washabaugh and W.G. Knauss. A reconciliation of dynamic crack velocity and rayleigh wave speed in isotropic brittle solids. *International Journal of Fracture*, 65:97–114, 1994.

[37] J. Whitney. *Structural Analysis of Laminated Anisotropic Plates*. Pergamon Press, Oxford, UK, 1997.

[38] M.L. Williams. On the stress distribution at the base of a stationary crack. *Journal of Applied Mechanics*, 24:109–114, 1957.

[39] T.O. Williams. A generalized multilength scale nonlinear composite plate theory with delamination. *International Journal of Solids and Structures*, 36:3015–3050, 1999.

[40] T.O. Williams and F.L. Addessio. A general theory of laminated plates with delaminations. *International Journal of Solids and Structures*, 34(16):2003–2024, 1997.

[41] T.O. Williams and F.L. Addessio. The dynamic behavior of laminated plates with delaminations. *International Journal of Solids and Structures*, 35(1-2):83–106, 1998.

Appendix A

Derivation of DCB

The energy, Π, input to the system is

$$\Pi = E_{interface} + U - W \tag{A.1}$$

where $E_{interface}$ is the energy consumed by the interface, U is the strain energy, and W is the work done on the system by the loading at the tips of the DCB. Π is not a potential because $E_{interface}$ cannot be recovered. However, it is equivalent to a potential if the tip displacement increases monotonically.

The components of A.1 are defined as

$$E_{interface} = G_c bL, \tag{A.2}$$

$$U = \frac{3EId^2}{L^3}, \tag{A.3}$$

$$W = 2\int_0^d f(d\prime)\mathrm{d}d\prime. \tag{A.4}$$

1. In Equations A.2 through A.4, b is the width of the beam and $f(d)$ and d are the reaction force and the displacement, respectively, at the beam tip.

The moment of inertia is

$$I = \frac{ba^3}{12}, \tag{A.5}$$

where a is the height of the beam.

In A.1 there are two degrees of freedom: d and L. Stationarity with respect to d gives

$$\frac{\partial \Pi}{\partial d} = \frac{6EId}{L^3} - 2f(d) = 0 \tag{A.6}$$

which relates beam tip displacement to reaction force for a cantilever beam. Stationarity with respect to L gives

$$\frac{\partial \Pi}{\partial L} = G_c b - \frac{9EId^2}{L^4} = 0. \tag{A.7}$$

Combining A.6 and A.7 gives the following expression:

$$\frac{f(d)}{b} = 2^{-\frac{1}{2}} 3^{-\frac{3}{4}} G_c^{\frac{3}{4}} (Ea)^{\frac{1}{4}} \frac{1}{\sqrt{d}} \tag{A.8}$$

Appendix B

Source Code Excerpts

The finite element computer program used for the simulations in this work is a modified version of an open source, general purpose finite element code named CalculiX. More information about this program, as well as the (unmodified) source code, is available at http://www.calculix.com/ and http://www.dhondt.de/. To include all of this author's modifications and additions to the finite element code would make an oppressively voluminous appendix. However, source code can be a precise and concise way to describe an algorithm. Therefore, the most critical excerpts from the source code are presented here.

B.1 Cohesive Zone Model

B.1.1 Function to Compute the Interfacial Traction, T

```
      subroutine interface_t(
     &     u,u_max,T,elcon,nmat,ncmat_,ntmat_)

      implicit none

      integer nmat,ncmat_,ntmat_
      real*8 u(3),u_max(3),T(3),elcon(0:ncmat_,ntmat_,*),
     &     E(3),delta(3),nu(3),nu_max(3)

      real*8 lambda,lambda_max,lambda_F,F
      integer m
```

```
                              ! Determine delta,E,nu,nu_max
      do m=1,3
         delta(m) = elcon(2,1,nmat)
         E(m) = elcon(1,1,nmat)
         nu(m) = u(m)/delta(m)
         nu_max(m) = u_max(m)/delta(m)
      enddo
                              ! Determine lambda_F
      call compute_lambda(nu,lambda)
      call compute_lambda(nu_max,lambda_max)
      if(lambda.ge.lambda_max) then
         lambda_F = lambda
      else
         lambda_F = lambda_max
      endif
                              ! Determine F(lambda_F)
      call compute_F_lambda(lambda_F,F)
                              ! T_m
      do m=1,3
         T(m) = E(m)*nu(m)*F
      enddo
                              ! Check for interpenetration
      if(nu(3).lt.0) then
         T(3) = E(3)*nu(3)
      endif

      return
      end
```

B.1.2 Function to Compute $\partial T/\partial u$

```
      subroutine interface_dtdu(
     &     u,u_max,dTdu,elcon,nmat,ncmat_,ntmat_)

      implicit none

      integer nmat,ncmat_,ntmat_
      real*8 u(3),u_max(3),dTdu(3,3),elcon(0:ncmat_,ntmat_,*),
     &     E(3),delta(3),nu(3),nu_max(3)
```

```
 real*8 nu_init,sigma_max,lambda,lambda_max,lambda_F,F,
&      dlambda_F_dnu(3),dFdlambda_F

 integer k,m,n

                              ! Determine delta,E,nu,nu_max
 do m=1,3
    delta(m) = elcon(2,1,nmat)
    E(m) = elcon(1,1,nmat)
    nu(m) = u(m)/delta(m)
    nu_max(m) = u_max(m)/delta(m)
 enddo
                              ! Determine lambda_F,dlambda_F_dnu
 call compute_lambda(nu,lambda)
 call compute_lambda(nu_max,lambda_max)
 if(lambda.ge.lambda_max) then
    lambda_F = lambda
    call compute_dlambda_F_dnu(nu,lambda,dlambda_F_dnu)
 else
    lambda_F = lambda_max
    do n=1,3
       dlambda_F_dnu(n) = 0
    enddo
 endif
                              !Determine F,dFdlambda_F
 call compute_F_lambda(lambda_F,F)
 call compute_dFdlambda_F(lambda_F,dFdlambda_F)
                              ! dTdu
 do m=1,3
    do n=1,3
       dTdu(m,n) = E(m)/delta(n)*nu(m)*dlambda_F_dnu(n)*dFdlambda_F
    enddo
 enddo


                              ! Add diagonal terms
 do m=1,3
    dTdu(m,m) = dTdu(m,m) + E(m)/delta(m)*F
 enddo
                              ! Check for interpenetration
 if(nu(3).lt.0) then
    dTdu(3,1) = 0
```

```
   dTdu(3,2) = 0
   dTdu(3,3) = E(3)/delta(3)
endif


return
end
```

B.1.3 Function to Compute the Damage Parameter, λ

```
subroutine compute_lambda(nu,lambda)

implicit none
real*8 nu(3),lambda

if(nu(3).gt.0) then
   lambda = sqrt(nu(1)**2 + nu(2)**2 + nu(3)**2)
else
   lambda = sqrt(nu(1)**2 + nu(2)**2)
endif

return
end
```

B.1.4 Function to Compute the Damage Function, F

```
subroutine compute_F_lambda(lambda,F)

implicit none
real*8 lambda,F

if(lambda.lt.1) then
   F = (1-lambda)**2
else
   F = 0
endif
if(lambda.lt.0) then
   write(*,*) "ERROR in compute_F_lambda:"
   write(*,*) "  lambda = ",lambda
   stop
endif
```

```
      return
      end
```

B.1.5 Function to Compute $\partial\lambda/\partial\nu$

```
      subroutine compute_dlambda_F_dnu(nu,lambda,dlambda_F_dnu)

      implicit none
      real*8 nu(3),lambda,dlambda_F_dnu(3)
      integer n

      dlambda_F_dnu(1) = nu(1)/lambda
      dlambda_F_dnu(2) = nu(2)/lambda
      if(nu(3).gt.0) then
         dlambda_F_dnu(3) = nu(3)/lambda
      else
         dlambda_F_dnu(3) = 0
      endif
c     get rid of division by zero problem, if necessary
      if(lambda.eq.0) then
         do n=1,3
            dlambda_F_dnu(n) = 0
         enddo
      endif

      return
      end
```

B.1.6 Function to Compute $\partial F/\partial\lambda$

```
      subroutine compute_dFdlambda_F(lambda_F,dFdlambda_F)

      implicit none
      real*8 lambda_F,dFdlambda_F

      if(lambda_F.lt.1) then
         dFdlambda_F = 2*(lambda_F-1)
      else
         dFdlambda_F = 0
      endif
      if(lambda_F.lt.0) then
```

```
         write(*,*) "ERROR in compute_dFdlambda_F:"
         write(*,*) "  lambda_F = ",lambda_F
         stop
      endif

      return
      end
```

B.2 Adaptive integration algorithm

```
c#define SAFE_ERROR 1
      subroutine dadmul(
     &     a,b_adapt,minpts,maxpts,eps,wk,iwk,result,
     &     relerr,nfnevl,ifail,nfuncs_,
     &     integrand_vars1,
     &     integrand_vars2)

      implicit none

      declare_integrand_vars

      logical ldv

      integer n,ifail,minpts,maxpts,iwk,nfnevl,nfuncs_
      real*8 a(*),b_adapt(*),wk(nfuncs_,*)
      real*8 ctr(15),wth(15),wthl(15),z(15)
      real*8 w(2:15,5),wp(2:15,3)

      real*8 f_return(nfuncs_)

      real*8 eps,relerr,relerr_safe,result(nfuncs_)
      integer ifncls,irgnst,irlcls,isbrgn,isbrgs,j_adapt,idvaxn,idvax0,
     &     j1,k_adapt,l,m_adapt,isbtmp,isbtpp,nfuncs_count,mef
      real*8 r1,hf,x12,x14,x15,w2,w4,wp2,wp4,abserr(nfuncs_),
     &     abserr_safe(nfuncs_),twondm,
     &     rgnvol,sum1(nfuncs_),sum2(nfuncs_),sum3(nfuncs_),
     &     sum4(nfuncs_),sum5(nfuncs_),dif,difmax,f2(nfuncs_),
     &     f3(nfuncs_),rgncmp(nfuncs_),rgnval(nfuncs_),rgnerr(nfuncs_)
```

```
      parameter (n = 2)
      parameter (r1 = 1, hf = r1/2)

      parameter (x12 =  0.35856 85828 00318 073d0)
      parameter (x14 =  0.94868 32980 50513 796d0)
      parameter (x15 =  0.68824 72016 11685 289d0)

      parameter (w2 =  980*r1/6561, w4 = 200*r1/19683)
      parameter (wp2 =  245*r1/486, wp4 = 25*r1/729)

      data (w(j_adapt,1),w(j_adapt,3),j_adapt=2,15)
     1/-0.193872885230909911d+00,  0.51821368693796768d-01,
     2 -0.555606360818980835d+00,  0.31499263323680330d-01,
     3 -0.876695625666819078d+00,  0.11177157953563989l1d-01,
     4 -0.115714067977442459d+01, -0.91449474165523547 3d-02,
     5 -0.139694152314179743d+01, -0.29467052786668698 6d-01,
     6 -0.159609815576893754d+01, -0.49789158156785042 4d-01,
     7 -0.175461057765584494d+01, -0.70111263526901376 8d-01,
     8 -0.187247878880251983d+01, -0.90433368897017724 1d-01,
     9 -0.194970278920896201d+01, -0.11075547426713407 1d+00,
     a -0.198628257887517146d+01, -0.13107757963725041 9d+00,
     b -0.198221815780114818d+01, -0.15139968500736675 2d+00,
     c -0.193750952598689219d+01, -0.17172179037748309 9d+00,
     d -0.185215668343240347d+01, -0.19204389574759944 7d+00,
     e -0.172615963013768225d+01, -0.21236600111771579 4d+00/

      data (w(j_adapt,5),w(j_adapt+1,5),j_adapt=2,14,2)
     1/ 0.871183254585174982d-01,  0.43559162729258750 8d-01,
     2  0.217795813646293754d-01,  0.10889790682314687 3d-01,
     3  0.544489534115734364d-02,  0.27224476705786719 3d-02,
     4  0.136122383528933596d-02,  0.68061191764466795 5d-03,
     5  0.340305958822333977d-03,  0.17015297941116699 5d-03,
     6  0.850764897055834977d-04,  0.42538244852791747 2d-04,
     7  0.212691224263958736d-04,  0.10634561213197937 2d-04/

      data (wp(j_adapt,1),wp(j_adapt,3),j_adapt=2,15)
     1/-0.133196159122085045d+01,  0.44581618655692729 2d-01,
     2 -0.229218106995884763d+01, -0.24005486968449930 9d-01,
     3 -0.311522633744855959d+01, -0.92592592592592587 5d-01,
     4 -0.380109739368998611d+01, -0.16117969821673525 1d+00,
     5 -0.434979423868312742d+01, -0.22976680384087791 5d+00,
```

```
     6 -0.476131687242798352d+01, -0.29835390946502056 4d+00,
     7 -0.503566529492455417d+01, -0.366941015089163228d+00,
     8 -0.517283950617283939d+01, -0.435528120713305891d+00,
     9 -0.517283950617283939d+01, -0.504115226337448555d+00,
     a -0.503566529492455417d+01, -0.572702331961591218d+00,
     b -0.476131687242798352d+01, -0.641289437585733882d+00,
     c -0.434979423868312742d+01, -0.709876543209876532d+00,
     d -0.380109739368998611d+01, -0.778463648834019195d+00,
     e -0.311522633744855959d+01, -0.847050754458161859d+00/

c      write(*,*) "in adapt_integrate.f"
c     Setup
      do nfuncs_count=1,nfuncs_
         result(nfuncs_count)=0
         abserr(nfuncs_count)=0
         abserr_safe(nfuncs_count)=0
      enddo
      ifail=3
      if(minpts .gt. maxpts) return

      mef = 1                    ! max error function; function with
!     the current maximum relative error (controls subdivision)
      ifncls=0                   !number of function evaluations
      ldv=.false.
      twondm=2**n                !constant = 4   subregion volume,
!     to be scaled later with wth
      irgnst=2*n+3               !constant = 7   abserr,result,idvax0,
!ctr,wth for each subregion
      irlcls=2**n+2*n*(n+1)+1    !constant = 17  points per subregion
      isbrgn=irgnst              !index in wk of current subregion ?
      isbrgs=irgnst              !last index in wk ?
      if(maxpts .lt. irlcls) return
      do j_adapt = 1,n
         ctr(j_adapt)=(b_adapt(j_adapt)+a(j_adapt))*hf !center of
!     subregion to integrate
         wth(j_adapt)=(b_adapt(j_adapt)-a(j_adapt))*hf !half width
!     of subregion to integrate
      enddo

c     Begin adaptive integration loop
   20 rgnvol=twondm
c      write(*,*) "mef = ",mef
```

```
      do j_adapt = 1,n
         rgnvol=rgnvol*wth(j_adapt)
         z(j_adapt)=ctr(j_adapt)
      enddo
c     < Sample Integrand >
      insert_integrand
      do nfuncs_count=1,nfuncs_
         sum1(nfuncs_count)=f_return(nfuncs_count)
      enddo

      difmax=0
      do nfuncs_count=1,nfuncs_
         sum2(nfuncs_count)=0
         sum3(nfuncs_count)=0
      enddo
      do j_adapt = 1,n
         z(j_adapt)=ctr(j_adapt)-xl2*wth(j_adapt)
         insert_integrand
         do nfuncs_count=1,nfuncs_
            f2(nfuncs_count)=f_return(nfuncs_count)
         enddo
         z(j_adapt)=ctr(j_adapt)+xl2*wth(j_adapt)
         insert_integrand
         do nfuncs_count=1,nfuncs_
            f2(nfuncs_count)=f2(nfuncs_count)+f_return(nfuncs_count)
         enddo
         wthl(j_adapt)=xl4*wth(j_adapt)
         z(j_adapt)=ctr(j_adapt)-wthl(j_adapt)
         insert_integrand
         do nfuncs_count=1,nfuncs_
            f3(nfuncs_count)=f_return(nfuncs_count)
         enddo
         z(j_adapt)=ctr(j_adapt)+wthl(j_adapt)
         insert_integrand
         do nfuncs_count=1,nfuncs_
            f3(nfuncs_count)=f3(nfuncs_count)+f_return(nfuncs_count)
         enddo
         do nfuncs_count=1,nfuncs_
            sum2(nfuncs_count)=sum2(nfuncs_count)+f2(nfuncs_count)
            sum3(nfuncs_count)=sum3(nfuncs_count)+f3(nfuncs_count)
         enddo
         dif=abs(7*f2(mef)-f3(mef)-12*sum1(mef))
```

```
      difmax=max(dif,difmax)
      if(difmax .eq. dif) then
         idvaxn=j_adapt
      endif
      z(j_adapt)=ctr(j_adapt)
   enddo

   do nfuncs_count=1,nfuncs_
      sum4(nfuncs_count)=0
   enddo
   do j_adapt = 2,n
      j1=j_adapt-1
      do k_adapt = j_adapt,n
         do l = 1,2
            wthl(j1)=-wthl(j1)
            z(j1)=ctr(j1)+wthl(j1)
            do m_adapt = 1,2
               wthl(k_adapt)=-wthl(k_adapt)
               z(k_adapt)=ctr(k_adapt)+wthl(k_adapt)
               insert_integrand
               do nfuncs_count=1,nfuncs_
                  sum4(nfuncs_count)=sum4(nfuncs_count)+
  &                     f_return(nfuncs_count)
               enddo
            enddo
         enddo
         z(k_adapt)=ctr(k_adapt)
      enddo
      z(j1)=ctr(j1)
   enddo

   do nfuncs_count=1,nfuncs_
      sum5(nfuncs_count)=0
   enddo
   do j_adapt = 1,n
      wthl(j_adapt)=-xl5*wth(j_adapt)
      z(j_adapt)=ctr(j_adapt)+wthl(j_adapt)
   enddo

90 continue
   insert_integrand
   do nfuncs_count=1,nfuncs_
```

```
         sum5(nfuncs_count)=sum5(nfuncs_count)+f_return(nfuncs_count)
      enddo
      do j_adapt = 1,n
         wthl(j_adapt)=-wthl(j_adapt)
         z(j_adapt)=ctr(j_adapt)+wthl(j_adapt)
         if(wthl(j_adapt) .gt. 0) then
            go to 90
         endif
c         continue
      enddo
c     </ Sample Integrand >

      do nfuncs_count=1,nfuncs_
         rgncmp(nfuncs_count)=rgnvol*(wp(n,1)*sum1(nfuncs_count)+
     &         wp2*sum2(nfuncs_count)+wp(n,3)*sum3(nfuncs_count)+
     &         wp4*sum4(nfuncs_count))
         rgnval(nfuncs_count)=w(n,1)*sum1(nfuncs_count)+
     &         w2*sum2(nfuncs_count)+w(n,3)*sum3(nfuncs_count)+
     &         w4*sum4(nfuncs_count)+w(n,5)*sum5(nfuncs_count)
         rgnval(nfuncs_count)=rgnvol*rgnval(nfuncs_count)
#ifdef SAFE_ERROR
         rgnerr(nfuncs_count)=abs(rgnval(nfuncs_count)-
     &         rgncmp(nfuncs_count))
#else
         rgnerr(nfuncs_count)=rgnval(nfuncs_count)-
     &         rgncmp(nfuncs_count)
#endif
         result(nfuncs_count)=result(nfuncs_count)+
     &         rgnval(nfuncs_count)
         abserr(nfuncs_count)=abserr(nfuncs_count)+rgnerr(nfuncs_count)
         abserr_safe(nfuncs_count)=abserr_safe(nfuncs_count)+
     &         abs(rgnval(nfuncs_count)-rgncmp(nfuncs_count))
      enddo
      ifncls=ifncls+irlcls

c     Divide?
      if(ldv) then              !integrate first half (or very first
!     subregion)
 110
     &         isbtmp=2*isbrgn
         if(isbtmp .gt. isbrgs) then !isbtmp is beyond previous max
!     index
```

```
            go to 160
          endif
          if(isbtmp .lt. isbrgs) then !isbtmp is within already used
!     memory
            isbtpp=isbtmp+irgnst
                                  !isbtmp set to index of greater
!     abserr of isbtmp and next abserr
#ifdef SAFE_ERROR
            if(wk(mef,isbtmp) .lt. wk(mef,isbtpp)) then !abserr of
!     isbtmp lt abserr of next abserr
#else
            if(abs(wk(mef,isbtmp)) .lt. abs(wk(mef,isbtpp))) then
!     abserr of isbtmp lt abserr of next abserr
#endif
               isbtmp=isbtpp
            endif
          endif
#ifdef SAFE_ERROR
          if(rgnerr(mef) .ge. wk(mef,isbtmp)) then !current subregion
!     error ge isbtmp error
#else
          if(abs(rgnerr(mef)) .ge. abs(wk(mef,isbtmp))) then
#endif
            go to 160
          endif
!     copy isbtmp to isbrgn
          do k_adapt = 0,irgnst-1
            do nfuncs_count=1,nfuncs_
              wk(nfuncs_count,isbrgn-k_adapt)=
     &             wk(nfuncs_count,isbtmp-k_adapt)
            enddo
          enddo
          isbrgn=isbtmp
          go to 110
        else                        !already integrated first half
  140     isbtmp=(isbrgn/(2*irgnst))*irgnst
          ! while...
          if(isbtmp .ge. irgnst .and.
#ifdef SAFE_ERROR
     &        rgnerr(mef) .gt. wk(mef,isbtmp)) then
#else
     &        abs(rgnerr(mef)) .gt. abs(wk(mef,isbtmp))) then
```

```
#endif
            do k_adapt = 0,irgnst-1
               do nfuncs_count=1,nfuncs_
                  wk(nfuncs_count,isbrgn-k_adapt)=
     &                 wk(nfuncs_count,isbtmp-k_adapt)
               enddo
            enddo
            isbrgn=isbtmp
            go to 140
         endif
       endif

  160 do nfuncs_count=1,nfuncs_
#ifdef SAFE_ERROR
         wk(nfuncs_count,isbrgn)=rgnerr(nfuncs_count)
#else
         wk(nfuncs_count,isbrgn)=rgnerr(nfuncs_count)
#endif
         wk(nfuncs_count,isbrgn-1)=rgnval(nfuncs_count)
         wk(nfuncs_count,isbrgn-2)=idvaxn
       enddo
       do j_adapt = 1,n
          isbtmp=isbrgn-2*j_adapt-2
          do nfuncs_count=1,nfuncs_
             wk(nfuncs_count,isbtmp+1)=ctr(j_adapt)
             wk(nfuncs_count,isbtmp)=wth(j_adapt)
          enddo
       enddo
       if(ldv) then              !integrate other half of newly
!      divided subregion
          ldv=.false.
          ctr(idvax0)=ctr(idvax0)+2*wth(idvax0)
          isbrgs=isbrgs+irgnst
          isbrgn=isbrgs
          go to 20
       endif
c      pick the max relative error of all the integrated functions
       relerr = 0
       do nfuncs_count=1,nfuncs_
          if(abs(abserr(nfuncs_count)/result(nfuncs_count)).gt.
     &         abs(relerr))then
             relerr = abs(abserr(nfuncs_count)/result(nfuncs_count))
```

```
            mef = nfuncs_count
         endif
         if(abs(abserr_safe(nfuncs_count)/result(nfuncs_count)).gt.
     &        abs(relerr_safe))then
            relerr_safe = abs(abserr_safe(nfuncs_count)/
     &           result(nfuncs_count))
         endif
       enddo
c      Check for termination conditions

c      IWK is too small for the specified number MAXPTS?
       if(isbrgs+irgnst .gt. iwk) then
          ifail=2
       endif
c      MAXPTS is too small for the specified accuracy EPS?
       if(ifncls+2*irlcls .gt. maxpts) then
          ifail=1
       endif
c      Normal exit, relerr < eps?
       if(((abs(relerr) .lt. eps).or.(abs(relerr_safe).lt.eps)) .and.
     &     ifncls .ge. minpts) then
          ifail=0
       endif
       if(ifail .eq. 3) then

c      No termination; Prep next cycle
!
          ldv=.true.
          isbrgn=irgnst
          do nfuncs_count=1,nfuncs_
             abserr(nfuncs_count)=abserr(nfuncs_count)-
     &           wk(nfuncs_count,isbrgn)
             abserr_safe(nfuncs_count)=abserr_safe(nfuncs_count)-
     &           abs(wk(nfuncs_count,isbrgn))
             result(nfuncs_count)=result(nfuncs_count)-
     &           wk(nfuncs_count,isbrgn-1)
          enddo
          idvax0=wk(mef,isbrgn-2)
          do j_adapt = 1,n
             isbtmp=isbrgn-2*j_adapt-2
             ctr(j_adapt)=wk(mef,isbtmp+1)
             wth(j_adapt)=wk(mef,isbtmp)
```

```
            enddo
            wth(idvax0)=hf*wth(idvax0)
            ctr(idvax0)=ctr(idvax0)-wth(idvax0)
            go to 20
         endif
c        Terminate

         write(*,*) "func evals   ",ifncls
         nfnevl=ifncls
         return
         end
```

Appendix C

Example Finite Element Input Files

In the following input files, only a few of the entries for the node and element definitions and multiple point constraints are shown. This is done to give the reader an understanding by example without too much repetition.

C.1 Double Cantilever Beam (DCB)

```
*NODE, NSET=Nbeam
       1,  1.00000e-02,  1.81608e-11,  2.00000e-03
       2,  9.33333e-03,  1.81608e-11,  2.00000e-03
       3,  9.33333e-03,  1.00000e-03,  2.00000e-03
       4,  1.00000e-02,  1.00000e-03,  2.00000e-03
       5,  1.00000e-02,  1.81608e-11,  1.33333e-03
** More nodes...
*ELEMENT, TYPE=C3D20, ELSET=Ebeam
     1,     1,     2,     3,     4,     5,     6,     7,     8,     9,    10,
          11,    12,    17,    18,    19,    20,    13,    14,    15,    16
     2,     2,    21,    22,     3,     6,    23,    24,     7,    25,    26,
          27,    10,    30,    31,    32,    18,    14,    28,    29,    15
     3,    21,    33,    34,    22,    23,    35,    36,    24,    37,    38,
          39,    26,    42,    43,    44,    31,    28,    40,    41,    29
     4,    33,    45,    46,    34,    35,    47,    48,    36,    49,    50,
          51,    38,    54,    55,    56,    43,    40,    52,    53,    41
     5,    45,    57,    58,    46,    47,    59,    60,    48,    61,    62,
** More elements...
*NODE, NSET=Ninterface
```

```
     299,  1.00000e-02,  1.81608e-11, -1.05426e-11
     300,  9.33333e-03,  1.81608e-11, -1.05426e-11
     301,  9.33333e-03,  1.00000e-03, -1.05426e-11
     302,  1.00000e-02,  1.00000e-03, -1.05426e-11
     307,  9.66667e-03,  1.81608e-11, -1.05426e-11
** More nodes...
*ELEMENT, TYPE=INT1, ELSET=Einterface
    46,   299,   300,   301,   302,   409,   410,   411,   412,   307,   308,
         309,   310,   417,   418,   419,   420,   413,   414,   415,   416
    47,   300,   311,   312,   301,   410,   421,   422,   411,   315,   316,
         317,   308,   425,   426,   427,   418,   414,   423,   424,   415
    48,   311,   318,   319,   312,   421,   428,   429,   422,   322,   323,
         324,   316,   432,   433,   434,   426,   423,   430,   431,   424
    49,   318,   325,   326,   319,   428,   435,   436,   429,   329,   330,
         331,   323,   439,   440,   441,   433,   430,   437,   438,   431
    50,   325,   332,   333,   326,   435,   442,   443,   436,   336,   337,
         338,   330,   446,   447,   448,   440,   437,   444,   445,   438
** More elements...
*NSET,NSET=Nfixxz
512,
409,
513,
412,
1007,
** More nodes...
*MATERIAL,NAME=MATERIAL1
*ELASTIC
400e9,.19
*MATERIAL,NAME=MATERIAL2
*USER MATERIAL, CONSTANTS=4
1.2e9,5e-7,5,2
*DENSITY
0
*DEPVAR
90
*SOLID SECTION,MATERIAL=MATERIAL1,ELSET=Ebeam
*SOLID SECTION,MATERIAL=MATERIAL2,ELSET=Einterface
*BOUNDARY
**Mode I
Nbeam,2
Ninterface,2
Nfixxz,2,3
```

```
Nfixx,1,2
*STEP,NLGEOM,INC=1000
*STATIC,SOLVER=SPOOLES
.15,1,1e-6,.1
*BOUNDARY
**Mode I
Nload,3,3,2e-5
*NODE PRINT,NSET=Nload
U,RF
*NODE FILE
U
*EL FILE
PE
*END STEP
```

C.2 Dynamic elastic strip

```
*NODE, NSET=Nbeam
       1,  1.00000e+00,  1.86265e-09,  1.00000e-01
       2,  9.66667e-01,  1.86265e-09,  1.00000e-01
       3,  9.66667e-01,  1.00000e-01,  1.00000e-01
       4,  1.00000e+00,  1.00000e-01,  1.00000e-01
       5,  1.00000e+00,  1.86265e-09,  6.66667e-02
** More nodes...
*ELEMENT, TYPE=C3D20, ELSET=Ebeam
     1,     1,     2,     3,     4,     5,     6,     7,     8,     9,    10,
          11,    12,    17,    18,    19,    20,    13,    14,    15,    16
     2,     2,    21,    22,     3,     6,    23,    24,     7,    25,    26,
          27,    10,    30,    31,    32,    18,    14,    28,    29,    15
     3,    21,    33,    34,    22,    23,    35,    36,    24,    37,    38,
          39,    26,    42,    43,    44,    31,    28,    40,    41,    29
     4,    33,    45,    46,    34,    35,    47,    48,    36,    49,    50,
          51,    38,    54,    55,    56,    43,    40,    52,    53,    41
     5,    45,    57,    58,    46,    47,    59,    60,    48,    61,    62,
          63,    50,    66,    67,    68,    55,    52,    64,    65,    53
** More elements...
*NODE, NSET=Ninterface
     584,  1.00000e+00,  1.86265e-09,  0.00000e+00
     585,  9.66667e-01,  1.86265e-09,  0.00000e+00
     586,  9.66667e-01,  1.00000e-01,  0.00000e+00
     587,  1.00000e+00,  1.00000e-01,  0.00000e+00
```

```
     592,  9.83333e-01,  1.86265e-09,  0.00000e+00
** More nodes...
*ELEMENT, TYPE=INT1, ELSET=Einterface
    91,   584,   585,   586,   587,   799,   800,   801,   802,   592,   593,
         594,   595,   807,   808,   809,   810,   803,   804,   805,   806
    92,   585,   596,   597,   586,   800,   811,   812,   801,   600,   601,
         602,   593,   815,   816,   817,   808,   804,   813,   814,   805
    93,   596,   603,   604,   597,   811,   818,   819,   812,   607,   608,
         609,   601,   822,   823,   824,   816,   813,   820,   821,   814
    94,   603,   610,   611,   604,   818,   825,   826,   819,   614,   615,
         616,   608,   829,   830,   831,   823,   820,   827,   828,   821
    95,   610,   617,   618,   611,   825,   832,   833,   826,   621,   622,
         623,   615,   836,   837,   838,   830,   827,   834,   835,   828
** More elements...
*NSET,NSET=Nfixxz
1008,
802,
1007,
799,
1997,
** More nodes...
*MATERIAL,NAME=EL1
*ELASTIC
3.24e9,.35
*DENSITY
1.13e-3
*MATERIAL,NAME=EL2
*USER MATERIAL, CONSTANTS=4
3.24e8,.0012082,5,2
**3.24e8,.0012082,0
*DENSITY
0
*DEPVAR
90
*SOLID SECTION,MATERIAL=EL1,ELSET=Ebeam
*SOLID SECTION,MATERIAL=EL2,ELSET=Einterface
*AMPLITUDE,NAME=BIQUADRATIC
0,0.00000,1e-7,0.02000,2e-7,0.08000,3e-7,0.18000,
4e-7,0.32000,5e-7,0.50000,6e-7,0.68000,7e-7,0.82000,
8e-7,0.92000,9e-7,0.98000,1e-6,1.00000

*STEP,NLGEOM,INC=10000
```

```
*DYNAMIC, EXPLICIT
**DYNAMIC, EXPLICIT, DIRECT
**DYNAMIC
1e-9,1.75e-6,1e-10
*BOUNDARY, OP=NEW
Nbeam,2
Ninterface,2
Nfixxz,1,3
*BOUNDARY, AMPLITUDE=BIQUADRATIC
**Mode I
Ndispz,3,3,.005
Ndispz,1,2,0
*NODE PRINT,NSET=Ncrack
U
*EL PRINT,ELSET=Ebeam
STEN
*NODE FILE
U
*EL FILE
PE
*END STEP
```

C.3 Multi-length scale plate

C.3.1 Global scale finite element model: Plate

```
*NODE, NSET=Nsuperelements_0001
    2005,
    2006,
    2007,
    2008,
    2009,
** More nodes...
*NODE, NSET=Nall
       1, -6.00000e+00,  6.00000e+00,  1.25000e-01
       2, -5.75000e+00,  6.00000e+00,  1.25000e-01
       3, -5.75000e+00,  5.75000e+00,  1.25000e-01
       4, -6.00000e+00,  5.75000e+00,  1.25000e-01
       5, -6.00000e+00,  6.00000e+00, -1.25000e-01
** More nodes...
*ELEMENT, TYPE=C3D8, ELSET=Eall
     1,     1,     2,     3,     4,     5,     6,     7,     8
```

```
     2,     2,     9,    10,     3,     6,    11,    12,     7
     3,     9,    13,    14,    10,    11,    15,    16,    12
     4,    13,    17,    18,    14,    15,    19,    20,    16
     5,    17,    21,    22,    18,    19,    23,    24,    20
** More elements...
*NODE, NSET=Ncenter
2401,
2402,
*NODE, NSET=Ncenterpoint
2401,
*NODE, NSET=Nyedgepoint
97,
*MATERIAL,NAME=EL1
*ELASTIC
1.925e9,.39
*DENSITY
.03032
*SOLID SECTION,MATERIAL=EL1,ELSET=Eall
*AMPLITUDE,NAME=SPIKE
0,0,1e-4,1,2e-4,0

*STEP,NLGEOM,INC=1000
*DYNAMIC,SOLVER=ITERATIVE CHOLESKY
1e-5,1e-3,1e-5,1e-4
*BOUNDARY, OP=NEW
Ncenter,1,2
Ncenterpoint,1,3
Nyedgepoint,1
*DLOAD,AMPLITUDE=SPIKE
Eall,GRAV,10000000,0,0,1
*NODE PRINT,NSET=Nsuperelement_0001
*NODE FILE
U
*EL FILE
U
*END STEP
```

C.3.2 Local scale finite element model: Subelements

```
*NODE, NSET=Nsuperelements_0001
    2005,
```

```
    2006,
    2007,
    2008,
    2009,
** More nodes...
*NODE, NSET=Nsubelements_0001
    3001,
    3002,
    3003,
    3004,
    3005,
** More nodes...
*NODE, NSET=Nlamina1
    3001, -5.00000e-01, 1.00000e+00, 1.25000e-01
    3002, -2.50500e-01, 1.00000e+00, 1.25000e-01
    3003, -2.50500e-01, 7.50000e-01, 1.25000e-01
    3004, -5.00000e-01, 7.50000e-01, 1.25000e-01
    3005, -5.00000e-01, 1.00000e+00, 1.00000e-03
** More nodes...
*ELEMENT, TYPE=C3D20, ELSET=Elamina1
    1501,3001,3002,3003,3004,3005,3006,3007,3008,3009,3010,
    3011,3012,3017,3018,3019,3020,3013,3014,3015,3016
    1502,3002,3021,3022,3003,3006,3023,3024,3007,3025,3026,
    3027,3010,3030,3031,3032,3018,3014,3028,3029,3015
    1503,3004,3003,3033,3034,3008,3007,3035,3036,3011,3037,
    3038,3039,3019,3042,3043,3044,3016,3015,3040,3041
    1504,3003,3022,3045,3033,3007,3024,3046,3035,3027,3047,
    3048,3037,3032,3050,3051,3042,3015,3029,3049,3040
    1505,3034,3033,3052,3053,3036,3035,3054,3055,3038,3056,
    3057,3058,3043,3061,3062,3063,3041,3040,3059,3060
** More elements...
*NODE, NSET=Nlamina2
    3166, -5.00000e-01, 1.00000e+00, -1.00000e-03
    3167, -2.50500e-01, 1.00000e+00, -1.00000e-03
    3168, -2.50500e-01, 7.50000e-01, -1.00000e-03
    3169, -5.00000e-01, 7.50000e-01, -1.00000e-03
    3174, -3.75250e-01, 1.00000e+00, -1.00000e-03
** More nodes...
*ELEMENT, TYPE=C3D20, ELSET=Elamina2
    1533,3166,3167,3168,3169,3262,3263,3264,3265,3174,3175,
    3176,3177,3270,3271,3272,3273,3266,3267,3268,3269
    1534,3167,3178,3179,3168,3263,3274,3275,3264,3182,3183,
```

```
    3184,3175,3278,3279,3280,3271,3267,3276,3277,3268
    1535,3169,3168,3185,3186,3265,3264,3281,3282,3176,3189,
    3190,3191,3272,3285,3286,3287,3269,3268,3283,3284
    1536,3168,3179,3192,3185,3264,3275,3288,3281,3184,3194,
    3195,3189,3280,3290,3291,3285,3268,3277,3289,3283
    1537,3186,3185,3196,3197,3282,3281,3292,3293,3190,3200,
    3201,3202,3286,3296,3297,3298,3284,3283,3294,3295
** More elements...
*NODE, NSET=Ninterface1
    3005, -5.00000e-01, 1.00000e+00, 1.00000e-03
    3006, -2.50500e-01, 1.00000e+00, 1.00000e-03
    3007, -2.50500e-01, 7.50000e-01, 1.00000e-03
    3008, -5.00000e-01, 7.50000e-01, 1.00000e-03
    3017, -3.75250e-01, 1.00000e+00, 1.00000e-03
** More nodes...
*ELEMENT, TYPE=INT1, ELSET=Einterface1
    1517,3005,3006,3007,3008,3166,3167,3168,3169,3017,3018,
    3019,3020,3174,3175,3176,3177,3170,3171,3172,3173
    1518,3006,3023,3024,3007,3167,3178,3179,3168,3030,3031,
    3032,3018,3182,3183,3184,3175,3171,3180,3181,3172
    1519,3008,3007,3035,3036,3169,3168,3185,3186,3019,3042,
    3043,3044,3176,3189,3190,3191,3173,3172,3187,3188
    1520,3007,3024,3046,3035,3168,3179,3192,3185,3032,3050,
    3051,3042,3184,3194,3195,3189,3172,3181,3193,3187
    1521,3036,3035,3054,3055,3186,3185,3196,3197,3043,3061,
    3062,3063,3190,3200,3201,3202,3188,3187,3198,3199
** More elements...
*NODE, NSET=Nplysplit2
    3178, -1.00000e-03, 1.00000e+00, -1.00000e-03
    3179, -1.00000e-03, 7.50000e-01, -1.00000e-03
    3183, -1.00000e-03, 8.75000e-01, -1.00000e-03
    3192, -1.00000e-03, 5.00000e-01, -1.00000e-03
    3194, -1.00000e-03, 6.25000e-01, -1.00000e-03
** More nodes...
*ELEMENT, TYPE=INT1, ELSET=Eplysplit2
    1557,3419,3274,3275,3420,3421,3178,3179,3422,3423,3279,
    3424,3425,3428,3183,3429,3430,3426,3276,3277,3427
    1558,3420,3275,3288,3431,3422,3179,3192,3432,3424,3290,
    3433,3434,3429,3194,3436,3437,3427,3277,3289,3435
    1559,3431,3288,3299,3438,3432,3192,3203,3439,3433,3301,
    3440,3441,3436,3205,3443,3444,3435,3289,3300,3442
    1560,3438,3299,3310,3445,3439,3203,3214,3446,3440,3312,
```

```
    3447,3448,3443,3216,3450,3451,3442,3300,3311,3449
    1561,3445,3310,3321,3452,3446,3214,3225,3453,3447,3323,
    3454,3455,3450,3227,3457,3458,3449,3311,3322,3456
** More elements...

*MATERIAL,NAME=EL1
*ELASTIC
1.925e9,.39
*DENSITY
.03032
*MATERIAL,NAME=EL2
*USER MATERIAL, CONSTANTS=4
2e5,1e-3,5,2
*DENSITY
0
*DEPVAR
90
*MATERIAL,NAME=EL3
*ELASTIC, TYPE=ORTHO
3.6879e+09,1.61028e+08,3.51237e+08,1.61028e+08,1.73283e+08,3.51237e+08,1.45415e+08,1
7.8791e+07,0
*DENSITY
.03032
*MATERIAL,NAME=EL4
*ELASTIC, TYPE=ORTHO
3.51237e+08,1.61028e+08,3.6879e+09,1.73283e+08,1.61028e+08,3.51237e+08,1.45415e+08,7
1.45415e+08,0
*DENSITY
.03032
*SOLID SECTION,MATERIAL=EL3,ELSET=Elamina1
*SOLID SECTION,MATERIAL=EL4,ELSET=Elamina2
*SOLID SECTION,MATERIAL=EL2,ELSET=Einterface1
*SOLID SECTION,MATERIAL=EL2,ELSET=Eplysplit2
*EQUATION
    48
 3013,1,  -0.7452365240, 2005,1,   1.0000000000, 2006,1,
   0.5000000000, 2007,1,   0.5000000000,
 2008,1,   0.2500000000, 2103,1,   0.5000000000, 2104,1,
   0.2500000000, 2105,1,   0.2500000000,
 2106,1,   0.1250000000, 3001,1,   0.3729281370, 3002,1,
   0.3418050040, 3003,1,   0.2486811290,
 3004,1,   0.3415565100, 3005,1,   0.3728667560, 3006,1,
```

```
  0.3111145080, 3007,1,   0.2179292520,
3008,1,   0.3108046330, 3009,1,  -0.6209895160, 3010,1,
 -0.3111145080, 3011,1,  -0.3104947580,
3012,1,  -0.6203697660, 3014,1,  -0.3737350001, 3015,1,
 -0.1868675000, 3016,1,  -0.3726182620,
3017,1,  -0.4979820081, 3018,1,  -0.2494879921, 3019,1,
 -0.2489910040, 3020,1,  -0.4974850200,
3166,1,   0.1230075080, 3167,1,   0.1230075080, 3168,1,
  0.0923170120, 3169,1,   0.1229461270,
3174,1,  -0.2460150159, 3175,1,  -0.1232530319, 3176,1,
 -0.1230075080, 3177,1,  -0.2457694920,
3262,1,   0.1229461270, 3263,1,   0.0923170120, 3264,1,
  0.0615651350, 3265,1,   0.0921942500,
3266,1,  -0.2457694920, 3267,1,  -0.1232530319, 3268,1,
 -0.0616265160, 3269,1,  -0.1228847460,
3270,1,  -0.1230075080, 3271,1,  -0.0616265160, 3272,1,
 -0.0615037540, 3273,1,  -0.1228847460,
   84
3014,1,  -0.7452365240, 2005,1,   0.2500000000, 2006,1,
  0.1250000000, 2007,1,   1.0000000000,
2008,1,   0.5000000000, 2009,1,   0.2500000000, 2010,1,
  0.1250000000, 2103,1,   0.1250000000,
2104,1,   0.0625000000, 2105,1,   0.5000000000, 2106,1,
  0.2500000000, 2107,1,   0.1250000000,
2108,1,   0.0625000000, 3001,1,   0.1701891955, 3002,1,
  0.3729281370, 3003,1,   0.3415565100,
3004,1,   0.1237515050, 3005,1,   0.1548746380, 3006,1,
  0.3728667560, 3007,1,   0.3108046330,
3008,1,   0.1084369475, 3009,1,  -0.3092552580, 3010,1,
 -0.6203697660, 3011,1,  -0.1546276290,
3012,1,  -0.1546276290, 3013,1,  -0.1857507620, 3015,1,
 -0.3726182620, 3016,1,  -0.0928753810,
3017,1,  -0.2479970280, 3018,1,  -0.4974850200, 3019,1,
 -0.1239985140, 3020,1,  -0.1239985140,
3021,1,   0.1716163085, 3022,1,   0.1249311160, 3023,1,
  0.1562408620, 3024,1,   0.1094937965,
3025,1,  -0.3117342580, 3026,1,  -0.1564918470, 3027,1,
 -0.1558671290, 3028,1,  -0.1879902060,
3029,1,  -0.0939951030, 3030,1,  -0.2499849801, 3031,1,
 -0.1254934620, 3032,1,  -0.1249924900,
3166,1,   0.0612582300, 3167,1,   0.1230075080, 3168,1,
  0.1229461270, 3169,1,   0.0459436725,
```

```
 3174,1,  -0.1225164600, 3175,1,  -0.2457694920, 3176,1,
  -0.0612582300, 3177,1,  -0.0612582300,
 3178,1,   0.0617492780, 3179,1,   0.0463733395, 3182,1,
  -0.1234985559, 3183,1,  -0.0619948019,
 3184,1,  -0.0617492780, 3262,1,   0.0459436725, 3263,1,
    0.1229461270, 3264,1,   0.0921942500,
 3265,1,   0.0306291150, 3266,1,  -0.0612582300, 3267,1,
  -0.2457694920, 3268,1,  -0.1228847460,
 3269,1,  -0.0306291150, 3270,1,  -0.0612582300, 3271,1,
  -0.1228847460, 3272,1,  -0.0306291150,
 3273,1,  -0.0306291150, 3274,1,   0.0463733395, 3275,1,
    0.0309360200, 3276,1,  -0.0619948019,
 3277,1,  -0.0309974010, 3278,1,  -0.0617492780, 3279,1,
  -0.0309974010, 3280,1,  -0.0308746390,
 3358,1,  -0.0000009920, 3359,1,  -0.0000014920, 3360,1,
  -0.0000005000, 3361,1,  -0.0000014920,
 3364,1,   0.0000039839, 3365,1,   0.0000059679, 3366,1,
    0.0000029839, 3369,1,   0.0000049679,
** More Multiple Point Constraints...
*STEP,NLGEOM,INC=1200
*STATIC,SOLVER=PROFILE
*BOUNDARY
Nsuperelements_0001,1,3,1
*NODE FILE
U
*END STEP
```