APPLICATION OF ENERGY METHODS TO MODELING FAILURES IN COMPOSITE MATERIALS AND STRUCTURES

by
William Joseph Ritter

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Mechanical Engineering

MONTANA STATE UNIVERSITY
Bozeman, Montana

June 2004

APPROVAL

of a thesis submitted by

William Joseph Ritter

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

Dr. Douglas S. Cairns	(Signature)	Date

Approved for the Department of Mechanical Engineering

Dr. Vic A. Cundy	(Signature)	Date

Approved for the college of Graduate Studies

Dr. Bruce R. McLeod	(Signature)	Date

This thesis is dedicated to my parents Gregory and Barbara Ritter. Without the upbringing they gave me or their continuing support none of my achievements would be possible. Thanks Mom and Dad.

I would like to thank my advisor Dr. Douglas Cairns, for his patience and support with this project. I thank the Navy Research Lab for funding this research and I would also like to thank all of those in the office of Mechanical and Industrial engineering for their cheerful help with each of my questions and issues.

TABLE OF CONTENTS

List of Tables

Table Page

List of Figures

Figure Page

ABSTRACT

Characterizing the mechanical properties of composite materials is difficult and expensive. There is a legacy for the scale up from basic materials testing to final structures in composites. Each material architecture and manufacturing technique potentially represents a different mechanical response in a structure. Hence, as new composite material forms and manufacturing techniques become available, a need exists to streamline the characterization process.

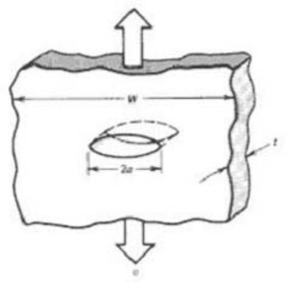
In this study, a new methodology for characterization of composite materials and structures is presented. It has its roots in fracture mechanics, but has been extended to the complexities of composite materials. The methodology is provided along with sample applications. While preliminary, the methodology has the potential for providing a meaningful scaling procedure for the materials / manufacturing / structural performance links for composite materials and structures.

CHAPTER 1: INTRODUCTION

Composite materials and structures are enabling new commercial, industrial, aerospace, marine, and recreational structures. Much previous work has been done to understand the application of these materials for primary structural applications [e.g. 1-4]. Unfortunately, as a new material, material architecture, manufacturing technique, or structural configuration is considered, it becomes necessary to perform an expensive and difficult series of tests for applications to structure.

A rather mature methodology for the scale up of metallic structures from laboratory test to large primary structures exists in the form of fracture mechanics as shown in figure 1. Either a stress intensity factor K is utilized, or the strain energy release rate G is utilized equivalently [5,6] Figure 1 is a rather simplistic representation since factors such as material constitutive behavior, environmental effects etc. need to be introduced for application to primary structure. The point, however, is that a well established framework and hierarchy exists for structures manufactured from homogenous, isotropic materials.

Fracture Mechanics Used With Metallic Structures



For limit loading:

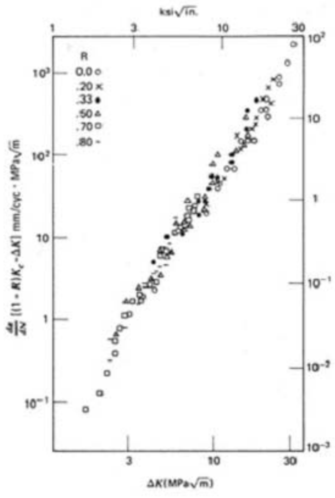
$$K_{1C} = \sigma(\pi a)^{1/2}$$

or critical strain energy release rate:

$$G_{1C} = K_{1C}^2 / E(1 - \upsilon^2)$$

For durability:

Paris' law for crack growth:
da/dn vs. ΔK



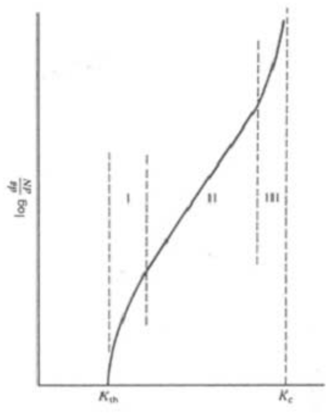


Figure 1. Current Approach for Metallic Structures

Fracture mechanics allow a designer to determine the characteristics of a structure after damage occurs, and determine weather a failure will be catastrophic or not and also the rate at which fractures will grow.

One failure criterion used with composite materials is shown in figure 2. This is the Tsai – Wu quadratic interaction criterion [ref].

Tsai – Wu Quadratic Interaction Criterion

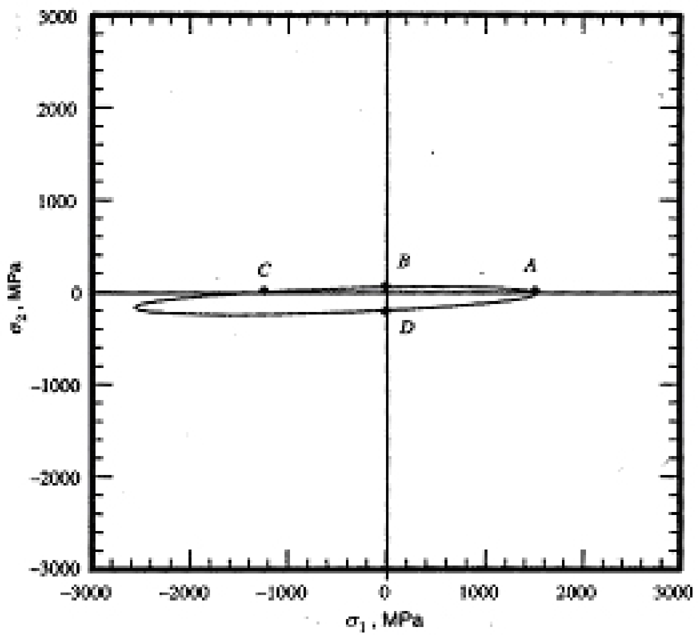


Figure 2. The Tsai – Wu Quadratic Interaction Criterion.

The ellipse in figure two describes the failure of a certain composite material for combinations of σ_{11} and σ_{22} (in two dimensional strain space this surface is an ellipse, in three dimensional space it is a hyper ellipsoid). If the stress state of the material is inside the ellipse no damage occurs. If the stress state is outside of the ellipse, damage has occurred. However, this ellipse fails to tell a designer is what the characteristics of the material are after damage has begun. The post-damage characteristics can be determined for isotropic materials using fracture mechanics.

The approach for developing aircraft structures is shown in Figures 3 and 4. Figure 3 is the basic scaling scheme, known as the building block approach in MIL HDBK 17 [8], while figure 4 is a specific set of tests for the F/A 18 E/F US Navy Hornet aircraft.

This approach is the common methodology for primary aircraft structures and is the current paradigm for ensuring safe, durable composite structures. Estimates for the cost of creating such as database range from $40-80 million dollars [9].

Clearly this paradigm cannot be utilized for composite wind turbine blade structures, but an extension beyond the specific and limited testing currently employed is warranted, especially in light of the costs of large composite structures being considered for military and commercial aerospace, Navy surface ship structures, wind turbine blades, etc.

Current Scale Up Approach For Composites

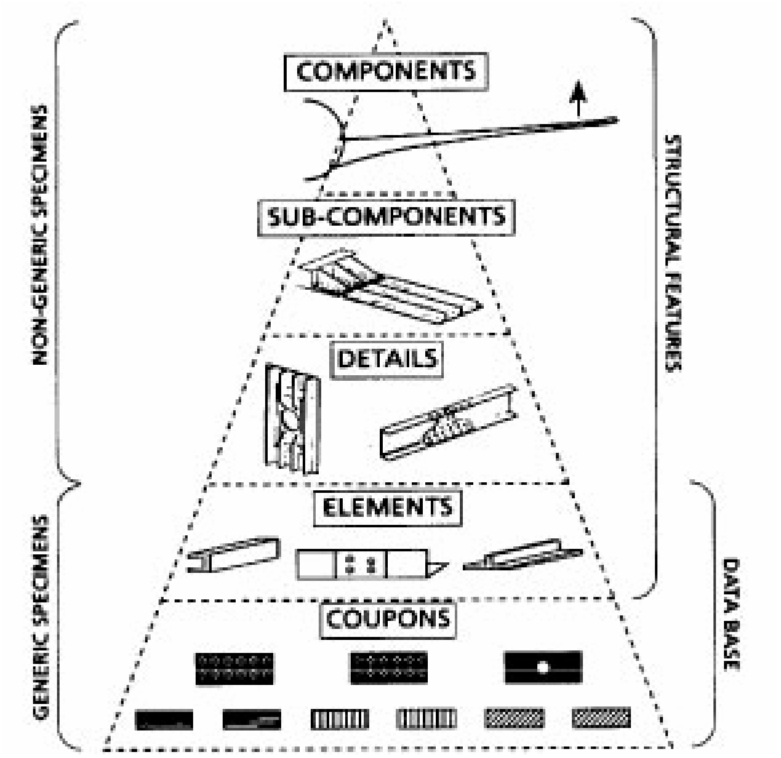


Figure 3. MIL HDBK 17 Building Block Approach for Primary Composite Structures

(www.mil17.org)

Example Elements For Aircraft

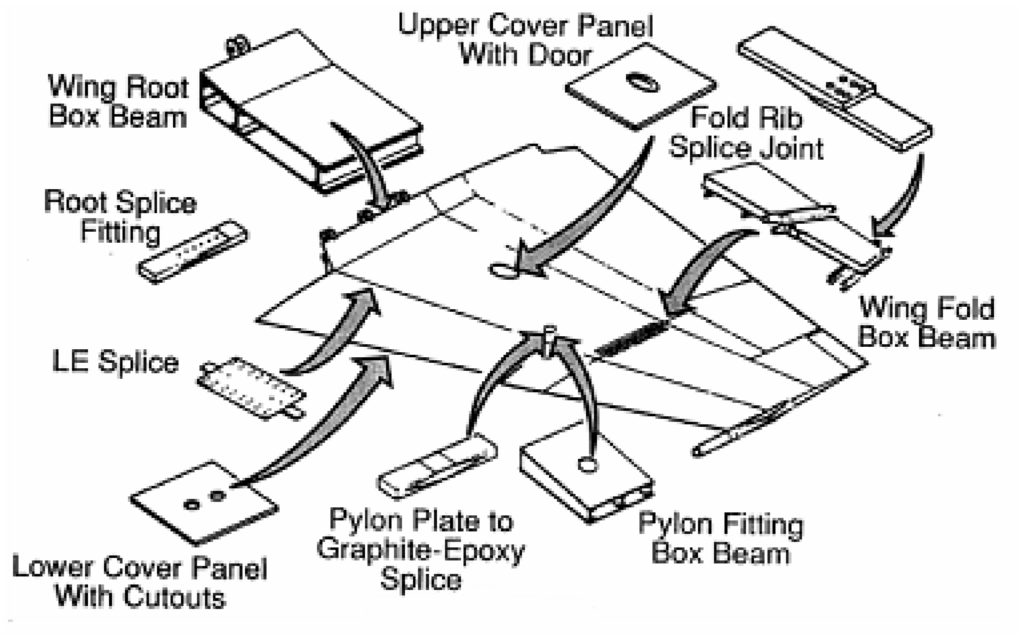


Figure 4. F/A 18 E/F US Navy Hornet aircraft material certification tests.

The goal of this research is to establish a new damage model for composites. This model will deal with the in plane characteristics of composite materials and the damage modes that occur in the plane. The intended use of this model is to parallel for composites what fracture mechanics addresses for metals. The goal is to develop a parameter reduction scheme such that the extensive and expensive tests required for the approach of figures 2 and 3 may be unnecessary. Furthermore, the application of the model and test data can be generic over a wide variety of structures, and not limited to a specific structural configuration.

The organization of this thesis is as follows. First, the basic concepts will be presented which have roots in fracture mechanics and thermodynamics [10,11]. Then a preliminary study for a fracture mechanics type sample will be presented. Finally, a more generic sample and testing methodology will be presented and discussed.

CHAPTER 2: DISSIPATED ENERGY AS A METRIC

Figure 5 is a typical load vs. displacement graph for a composite sample with brittle behavior (in other words no plasticity). In this figure the test specimen response follows line one as the load increases. At point two, the loading is stopped and the sample approximately follows line three to return to the origin, assuming that the nonlinearity is primarily a consequence of damage formation. This last step is approximately correct as composites typically do not exhibit permanent set type behavior. The shaded area between lines one and three represents the dissipated energy in the composite sample. We shall refer to this dissipated energy as Φ.

Load vs. Displacement of Composite Sample

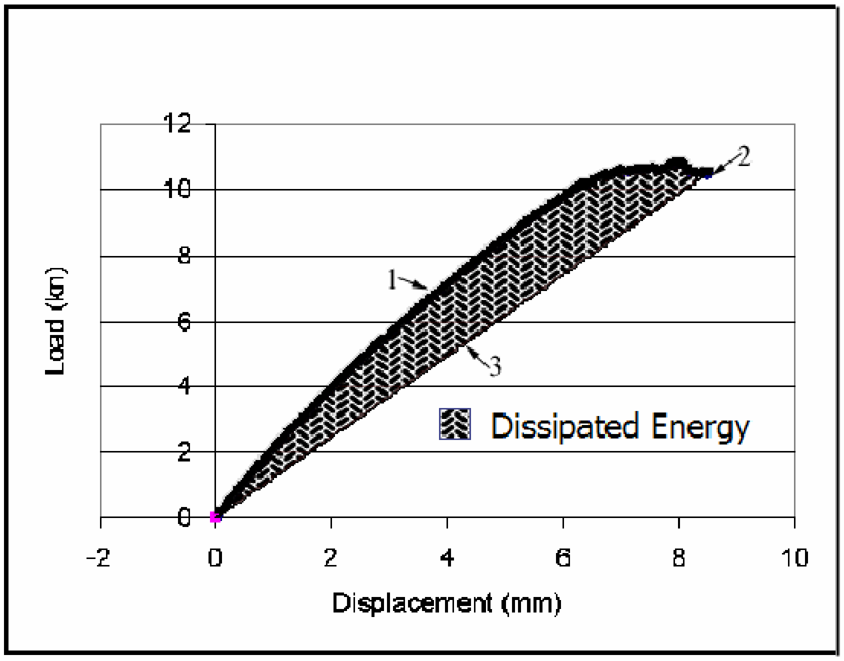


Figure 5. A generic load displacement graph for a composite sample.

The type of displacement and loading has been deliberately omitted from this graph. As energy is a scalar quantity it is independent of a coordinate system. The amount of energy dissipated provides an objective measure of the amount of damage in the sample. The goal of this project is to create a dissipated energy density function such that:

$$\Phi = \int \varphi(\varepsilon)\, dv$$

where v symbolizes the volume of a structure and the dissipated energy density function φ is a function of the strains ε in the differential volume of the structure.

(1)

This function will present several utilities to an analyst. If a dissipated energy density function can be created for a certain material system it can, with the help of finite element analysis, be used to analyze any structure made of the same material. A strain state can be created for some displaced model and the dissipated energy Φ calculated through integration of the dissipated energy density function φ over the structure. If the dissipated energy is greater than zero the analyst now knows that this displacement causes damage to the structure [12].

A second use of this function is in the creation of load vs. displacement relationships for a structure. Armed with the stiffness of a model and the dissipated energy for a number of displacements calculating the load vs. displacement relationship is simple.

The function will also inform the analyst of the areas of the structure that are incurring damage, thus allowing for alteration of the design. The mathematical structure of the dissipated energy density (DED) function is discussed in the next section.

Mathematical Structure of the DE Function

In order to provide a damage model based on the dissipated energy we propose the following steps:

1) Creation of a finite element model will allow the approximate strains to be known through the sample.

2) With this finite element model, a dissipated energy density function, $\phi(\varepsilon)$, can be created from the strains (ε) in the composite. In this study in-plane loads and displacements were considered, thus we have three in-plane strains: ε_{11}, strain in the fiber direction; ε_{22}, strain transverse to the fiber direction; and ε_{12}, the shear strain between ε_{11} and ε_{22}.

3) Finite element modeling for these structures allows several composite plies to be modeled in an element. To find the dissipated energy in a ply of the model the dissipated energy density function, $\phi(e)$, is multiplied by the volume of the ply. In the case of planar structures the volume of the element is equal to the area of the element multiplied by the thickness of the ply.

Thus the dissipated energy in a ply element is approximately equal to:

$$\Phi(e,p) \approx \phi(\varepsilon)*a(e)*t(p)$$

where t(p) is the thickness of the ply and a(e) is the area of the element.

(2)

4) The total dissipated energy in the element of the model can now be found by summation over the number of plies in the element:

$$\Phi(e) = \Sigma(p)\ \phi(\varepsilon)*a(e)*t(p)$$

where a(e) is the area of the element and t(p) is the thickness of the ply

(3)

5) The total dissipated energy in the model can then by found though summation over the number of elements in the model:

$$\Phi = \Sigma(e)\ \Sigma(p)\ \ \phi(\varepsilon)*a(e)*t(p)$$

which approximates the volume integral in equation 1.

(4)

It should be noted that the strains used in equation 4 are the element midpoint strains, the average strain values of the element corner nodes.

Note that the units on the dissipated energy density $\phi(\varepsilon)$ are energy/unit volume released during the fracture process. It is more general than either K, the stress intensity factor or G, the strain energy release rate since there is no a priori assumption of the direction of damage (crack) growth. This assumption of damage direction is required for applications to the critical stress intensity factor K_{IC}, or the critical strain energy release rate G_{IC} types of analyses.

Trial Formulations

Several formulations for the dissipated energy density function were studied. One method was to create a simple polynomial approximation of the form $\phi(\varepsilon)$ =

$$C(1)*\varepsilon 11 + C(2)*\varepsilon 22 + C(3)*\varepsilon 12 + C(4)*\varepsilon 11\char`^2 + C(5)*\varepsilon 22\char`^2 + C(6)*\varepsilon 12\char`^2 + \ldots \tag{5}$$

To accurately model the dissipated energy function over a large enough strain space many coefficients C are needed. This results in strains taken to high orders, which leads to numerical problems as well as the philosophical questions of the significance of a strain raised to the fourth power. This polynomial approach did work well with the compact tension samples discussed in a later section.

Another method was the use of Lagrangian interpolation [13] functions over a strain domain. To achieve the proper resolution these were fifth order interpolation functions. The problem with this approach is that higher order functions are poorly behaved (large spikes) and lead to erroneous results.

The final approach used is a method similar to the interpolation used in finite element formulation. An eight-node element is shown in figure 6.

Interpolation Element

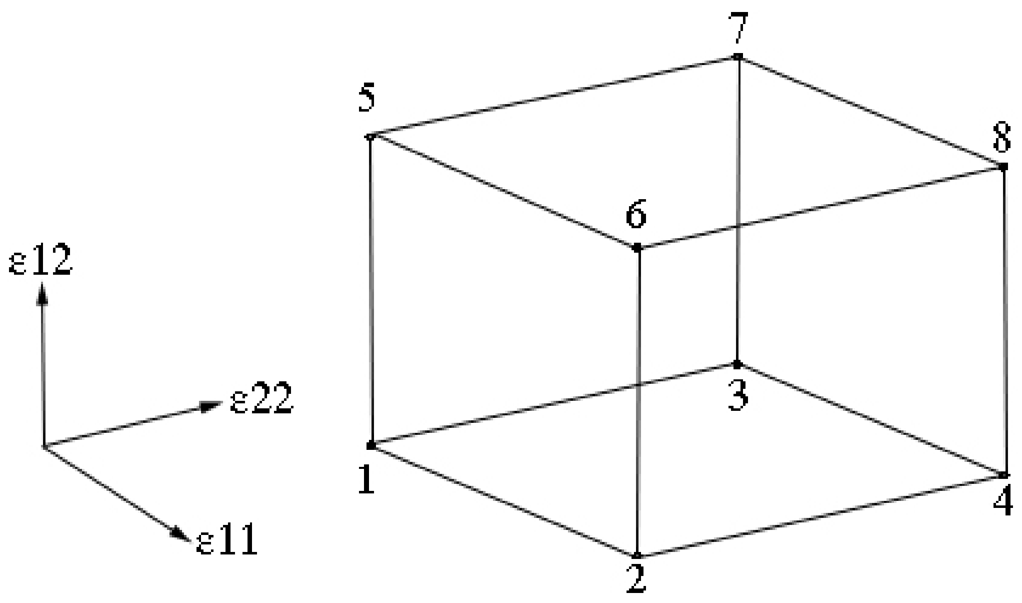


Figure 6. The eight-node interpolation element.

A value, C(n), of dissipated energy density is assigned for each node of the element, which are unknowns to be determined. Eight linear interpolation functions, f(n), can be created from the strain space coordinates of the nodes. These functions have the property of having a value of one at their corresponding node and a value of zero at all other nodes. Suppose the element above spans a dimension Δε11 in the ε11 direction, Δε22 in the ε22 direction, and Δε12 in the ε12 direction. If the strains at node one are ε11(1), ε22(1), and ε12(1) then three local coordinates (α, β and γ) that vary from zero to one can be assigned within the element.

These coordinates are:

$$\alpha = (\varepsilon_{11} - \varepsilon_{11}(1))/\Delta\varepsilon_{11} \quad (a)$$

$$\beta = (\varepsilon_{22} - \varepsilon_{22}(1))/\Delta\varepsilon_{22} \quad (b)$$

$$\gamma = (\varepsilon_{12} - \varepsilon_{12}(1))/\Delta\varepsilon_{12} \quad (c)$$

(6)

The eight interpolation functions are then:

$$f(1) = (1-\alpha)*(1-\beta)*(1-\gamma) \quad (a)$$

$$f(2) = \alpha*(1-\beta)*(1-\gamma) \quad (b)$$

$$f(3) = (1-\alpha)*\beta*(1-\gamma) \quad (c)$$

$$f(4) = \alpha*\beta*(1-\gamma) \quad (d)$$

$$f(5) = (1-\alpha)*(1-\beta)*\gamma \quad (e)$$

$$f(6) = \alpha*(1-\beta)*\gamma \quad (f)$$

$$f(7) = (1-\alpha)*\beta*\gamma \quad (g)$$

$$f(8) = \alpha*\beta*\gamma \quad (h)$$

(7)

The value of the dissipated energy density at any point within the element can then be found through summation:

$$\phi(\varepsilon 11,\varepsilon 22,\varepsilon 12) =$$

$$C(1)*f(1) + C(2)*f(2) + C(3)*f(3) + C(4)*f(4)$$

$$+ C(5)*f(5) + C(6)*f(6) + C(7)*f(7) + C(8)*f(8)$$

$$= C*f \tag{8}$$

To increase the accuracy of the dissipated energy density function the strain space was initially divided into 64 of these elements, with 125 nodes. This approach allows the complex behavior of the dissipated energy density to be approximated in a series of piecewise linear interpolation functions. The nodes for this model are shown in figure 6. This type of linear interpolation is included in many introductory finite element analysis books including “Finite Element Analysis” by David Burnett [14].

125 Node Solution Space

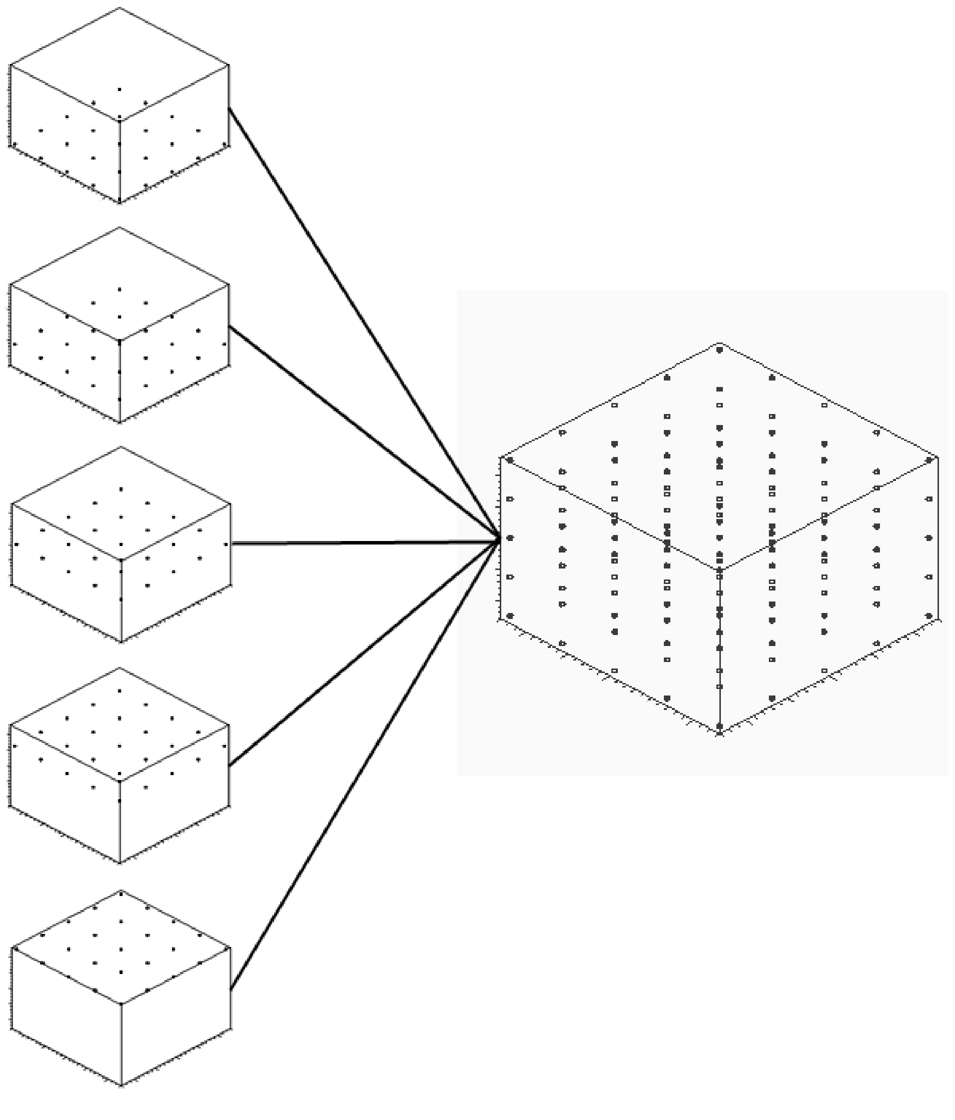


Figure 7. The 125 nodes used in the dissipated energy density function.

With a multi-element system such as this, a mapping procedure is performed between the local nodes in an element (nodes one through eight in figure 5) and global nodes one through 125 in figure 6. This mapping procedure is demonstrated in the two element dissipated energy example in the next section.

Deriving the C vector

Suppose we have a simple planar rectangular system that can be modeled with a single finite element. Such a system is shown in figure 8.

Planar Rectangular System

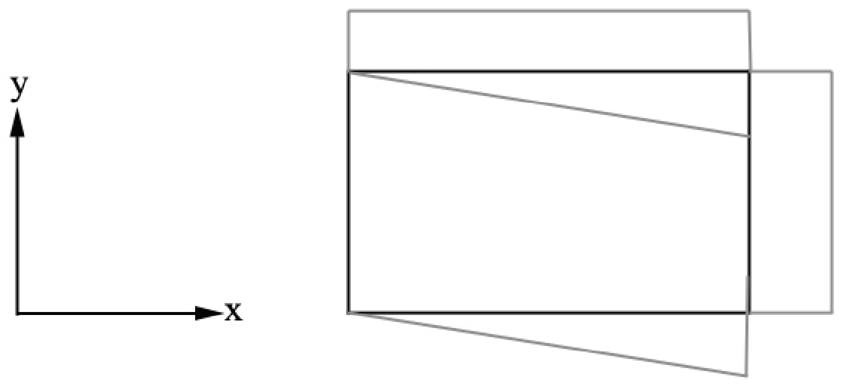


Figure 8. A planar system. Here the unperturbed system is shown in black and several possible displaced shapes are shown in gray.

Suppose any strain state can be applied to this system and the dissipated energy in the system can be measured. For the two element dissipated energy density function, we measure at least 12 data points. This two element system is shown in figure 9.

A Two Element Interpolation System

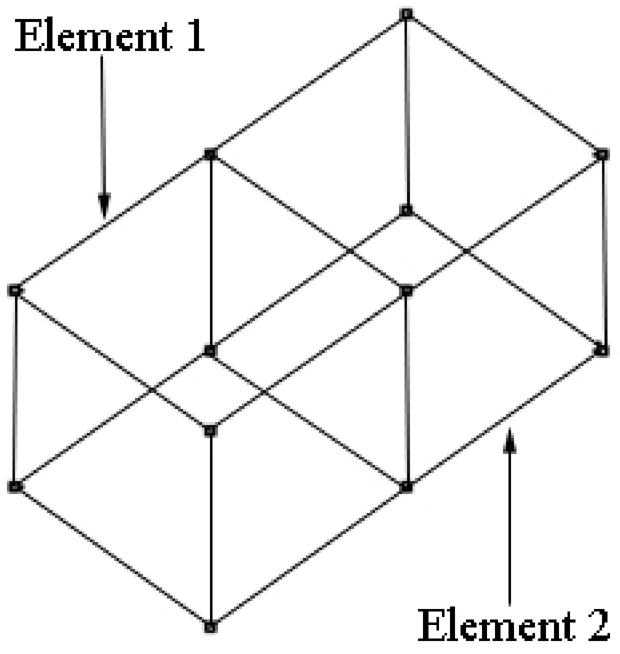


Figure 9. A two element system.

A global numbering system is imposed for the nodes of this system. The numbered system is shown in figure 10.

Global Nodal Numbering Scheme

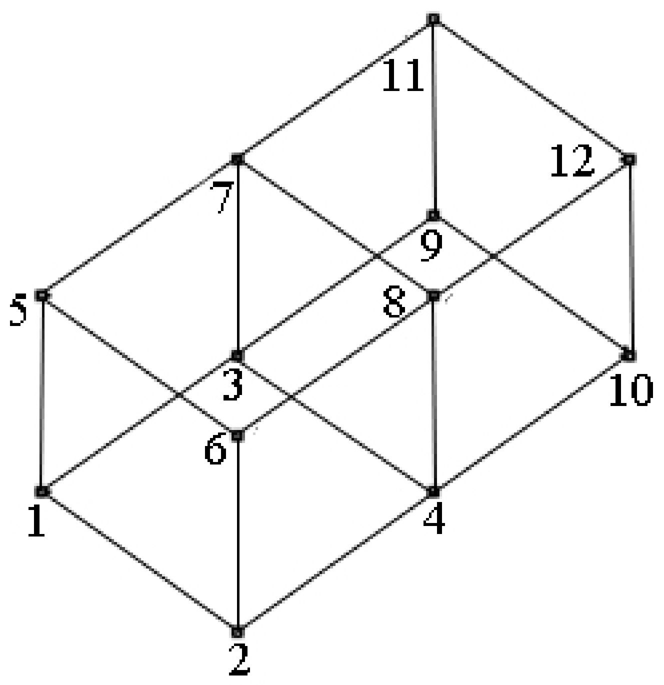


Figure 10. A two element system.

Suppose we wish to use this two element system to create a dissipated energy density function correlating to a one element finite element model. As there are 12 unknowns (the values of the dissipated energy function at nodes 1 through 12) we will need at least 12 data points. Load vs. displacement data are collected for this structure and dissipated energy is calculated.

We can then state:

$$\Phi_i = \{\mu_i\}^T \cdot \{C\}$$

where Φ_i is an experimentally measured dissipated energy value, μ_i is a vector of the element interpolation functions (see eqs 5 and 6) and C is a vector of the dissipated energy function element nodal values.

(9)

This function can be created for each experimental data point where the dissipated energy and displacements of the sample are known. Through vertical concatenation the following system of equations is created.

$$\{\Phi\} = [\mu]\{C\}$$

where Φ is a vector of the experimentally determined dissipated energy values, μ is a matrix of the interpolation function values for each experimental data point, and C is a vector of the dissipated energy function element nodal values.

(10)

The error in the approximation is then:

$$\left| [\mu]\{C\} - \{\Phi\} \right|$$

(11)

The proper values of C can then be found by minimizing this error with the constraint that the values of the C coefficients are positive.

These calculations were performed with Matlab [15] software. Matlab code is included in appendix A.

CHAPTER 3: COMPACT TENSION SAMPLE STUDY INTRODUCTION

Compact tension samples (CTS) were used to develop the initial dissipated energy characterization and methodology. A drawing of a compact tension sample is shown in figure 11. The loading direction and crack opening displacement are shown in figure 12.

Compact Tension Sample

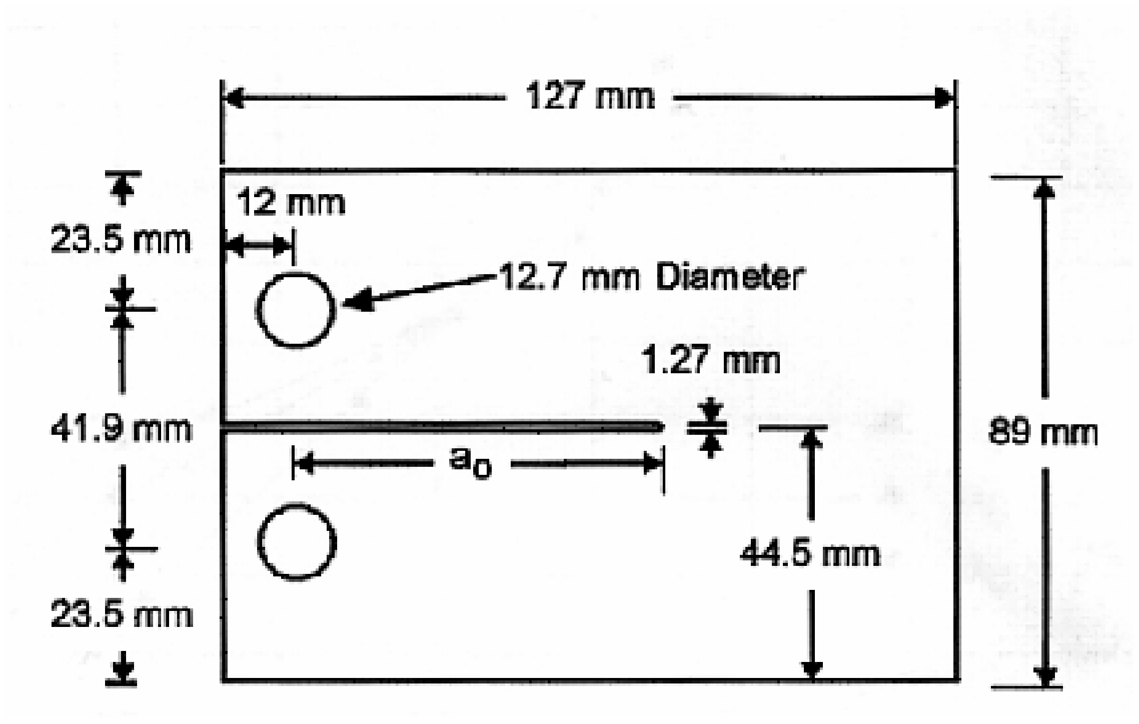


Figure 11. Compact tension sample dimensions.

Loading Direction For A Compact Tension Sample

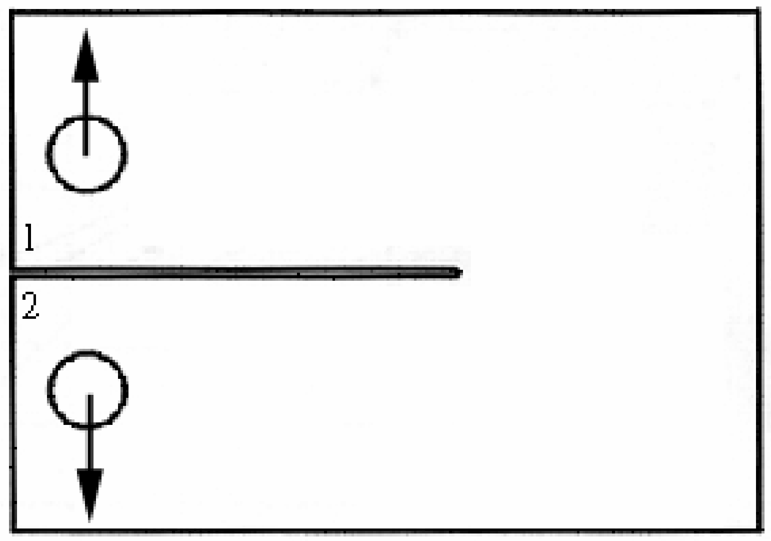


Figure 12. Loading direction for a compact tension sample. The crack opening displacement is measured between corners 1 and 2 of the sample.

These compact tension samples were 12.7 mm thick, with the exception of laminate 2, which was 10.4 mm thick. The fabric used was Knytex D155 e-glass [16]. These samples were manufactured Derakane vinylester resin[17]. Each compact tension sample was created with 28 plies, except for laminate 2 which had 24 plies. Six different laminates were created. These laminates schedules are shown in the following list; angles (degrees) are relative to the loading direction.

1:	90_{28}
2:	$(90_2/0)_{4s}$
3:	$(90_{13}/45)_s$
4:	$(90_7/\pm45/90_5)_s$
5:	$((90_4/\pm45)_2/90_2)_s$
6:	$((90_2/\pm45)_3/90/45)_s$

Load vs. displacement data were collected for each of these laminates. This testing was preformed on an Instron testing machine. Displacements changed slow enough to allow dynamic effects to be ignored.

CHAPTER 4: FINITE ELEMENT ANALYSIS OF CTS

Finite element models were created in Ansys[18] for each laminate and a unit displacement was applied. This allows the strain vs. displacement relationship for each laminate to be known. The Ansys mesh and a displaced model are shown in figures 13 and 14, (a complete description of the modeling process is given in chapters 8 and 9, this section is meant as an introduction to the dissipated energy method). At this point the experimental and finite element data are read into Matlab and the problem formulated as in equation 5. The C vector is then determined by minimization of the error function with the constraint that the values of C are positive.

Finite Element Model For Compact Tension Sample

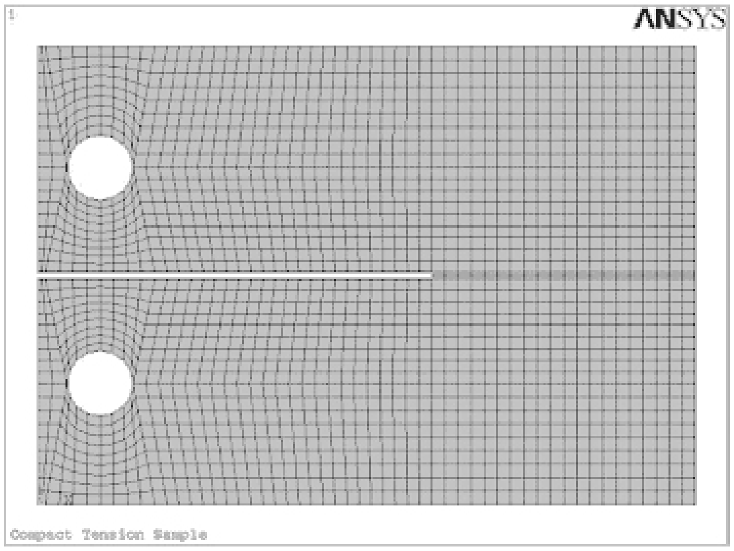


Figure 13. Finite element mesh in Ansys.

Compact Tension Sample Displaced

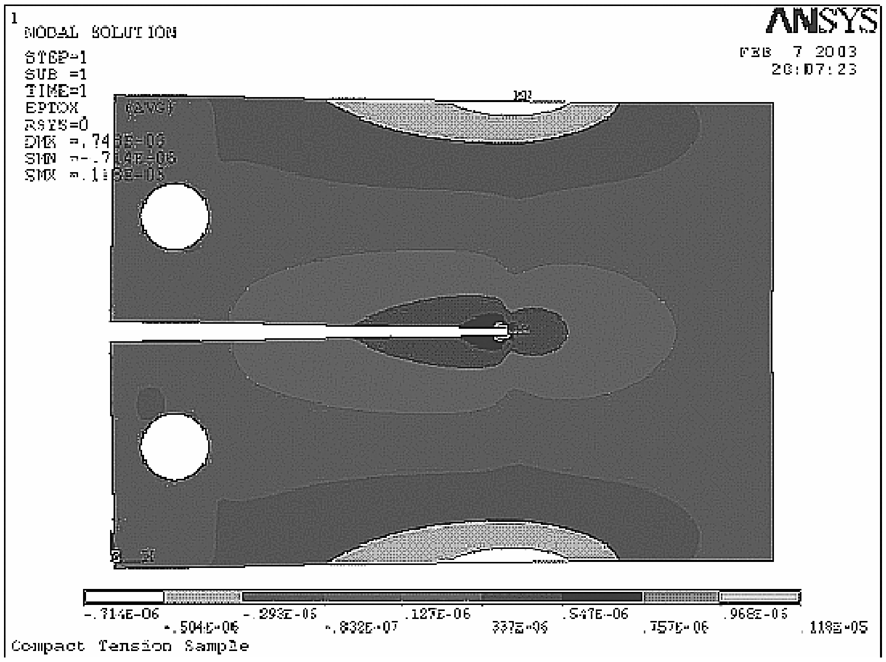


Figure 14. Displaced model generated in Ansys.

CHAPTER 5: APPLICATIONS AND EXAMPLES

Single case study

The load displacement relationship for laminate 2, $(90_2/0)_{4s}$, is shown in figure 15.

Load vs. Displacement For Laminate 2.

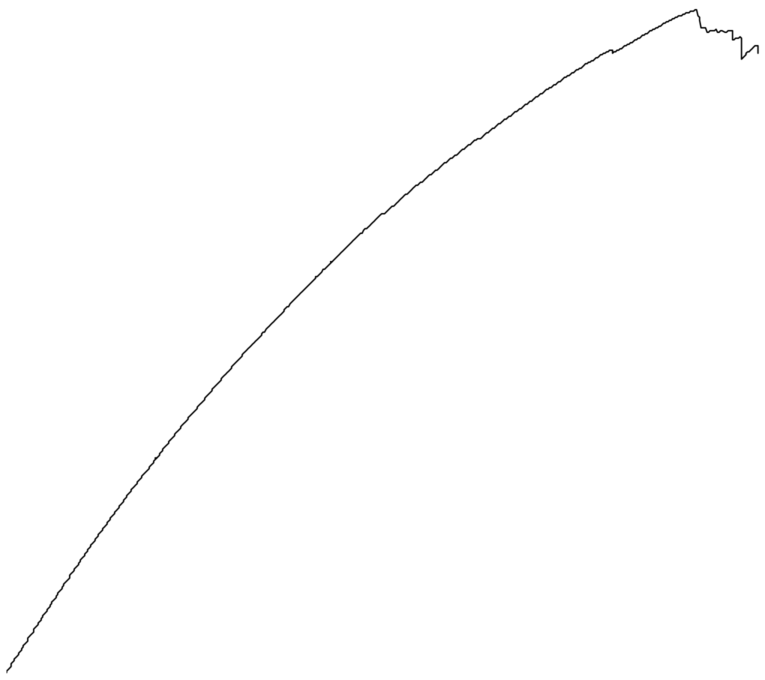


Figure 15. Load displacement graph for laminate two.

This data set contains 1195 data points. The dissipated energy calculated from this data is shown in figure 16.

Dissipated Energy vs. Crack Opening Displacement

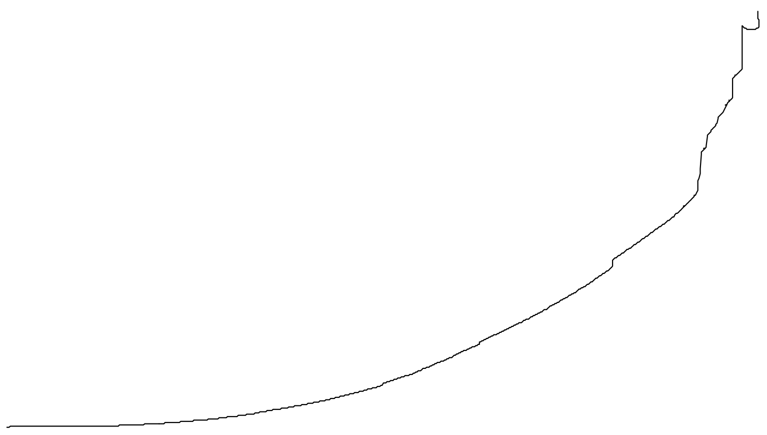


Figure 16. Dissipated Energy vs. Crack Opening Displacement for laminate 2.

The first question to answer about this approach is whether or not this method works for a single laminate. The following results were found for laminate two, using only data from laminate 2 to formulate the problem.

The approximate and experimental dissipated energy are shown in figure 17. Agreement between the two is good. In each case the approximate dissipated energy function is the smoother line.

Dissipated Energy vs. Displacement For Laminate 2

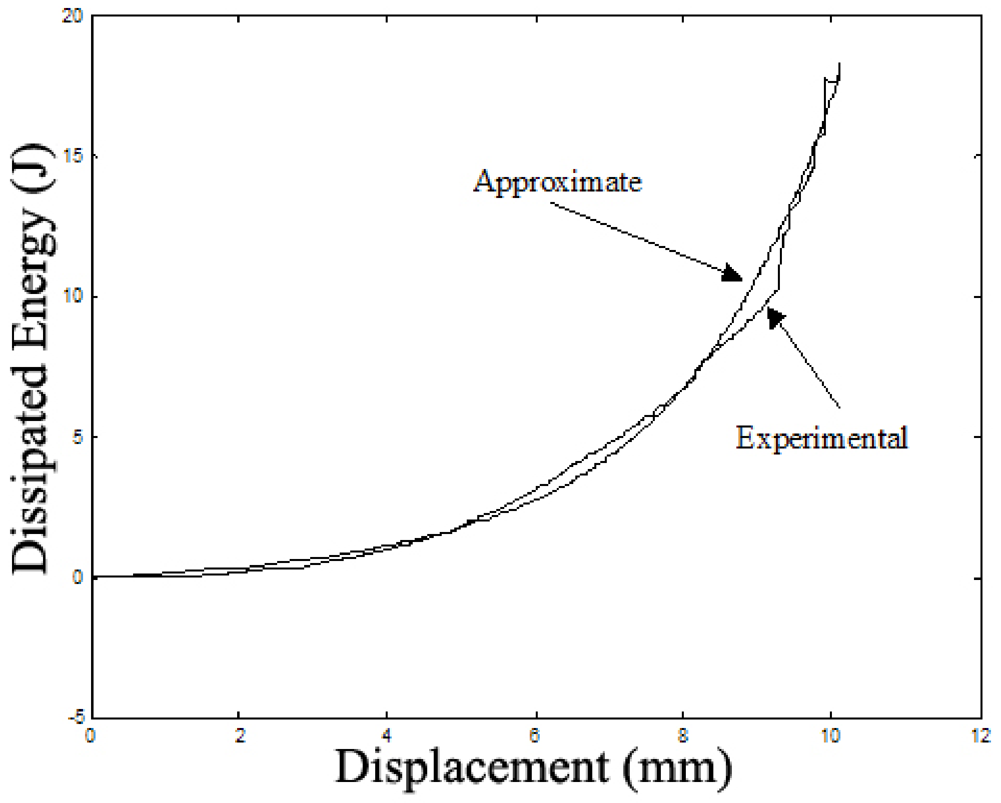


Figure 17. Approximate and experimental dissipated energy.

From this dissipated energy curve and the stiffness of the finite element model the following load vs. displacement relationship can also be calculated. The approximate and experimental loads are shown in figure 18.

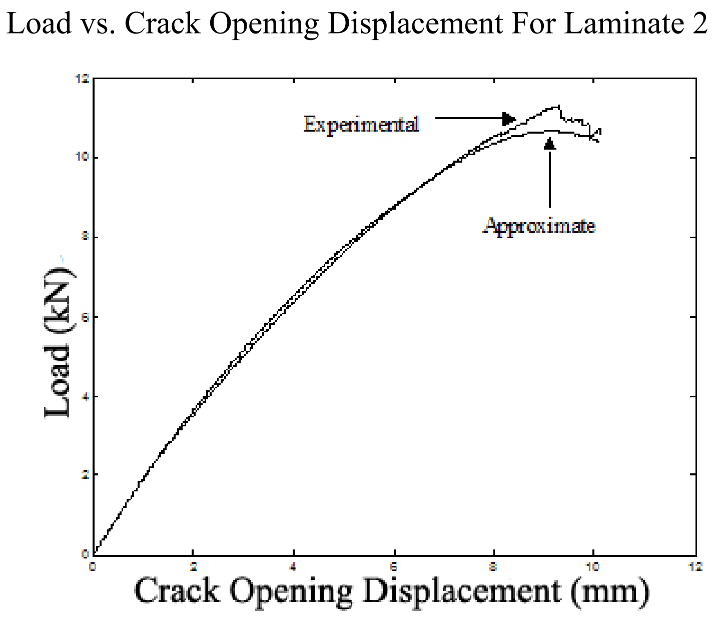


Figure 18. Approximate and experimental loads for laminate two.

In any case with displacements in one direction the load function can be accurately reconstructed if the dissipated energy function is known.

Compiled Case Study

The next test for this approach is to perform a compiled fit. In other words, use all of the available data to arrive at the C vector. The following results are for a compiled study of laminates 1 through 6. All of the individual systems of equations (equation 4) were vertically concatenated and solved for one C vector. The resulting dissipated energy predictions are shown in figures 19 through 24.

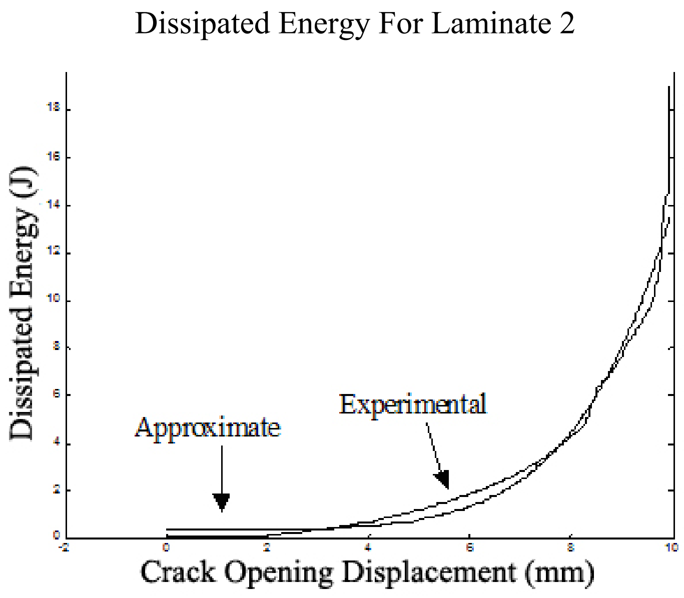


Figure 19. Experimental and approximate dissipated energy for laminate 2.

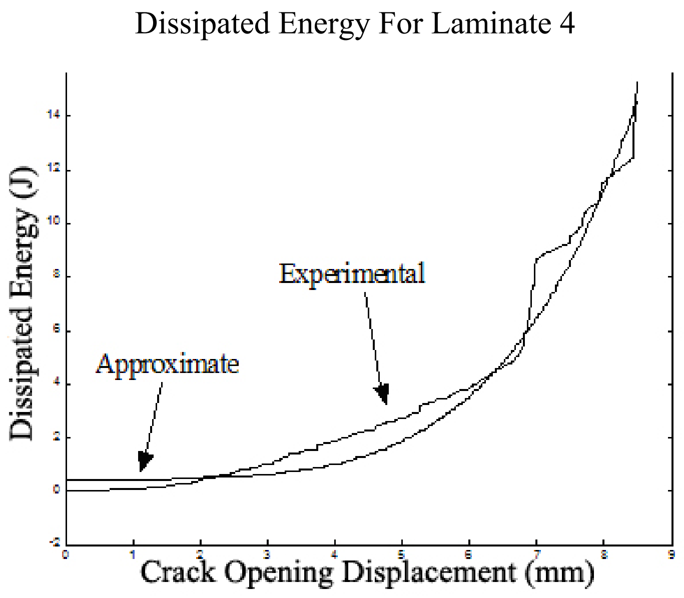


Figure 20. Experimental and approximate dissipated energy for laminate 4.

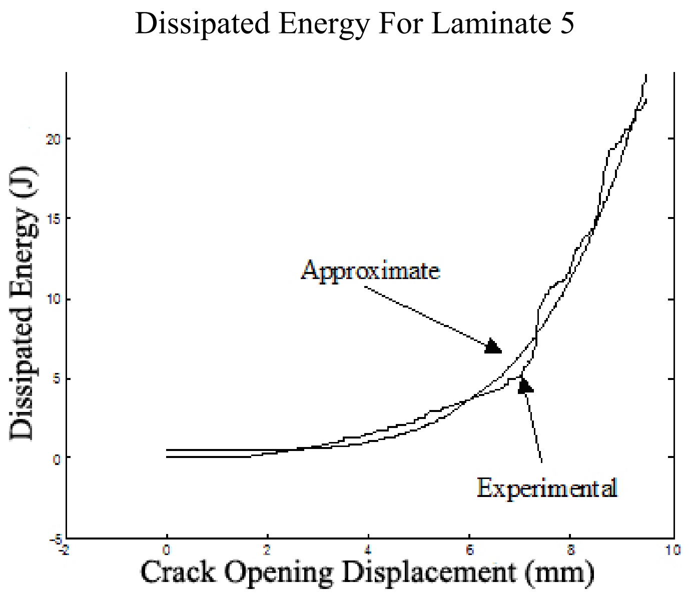


Figure 21. Experimental and approximate dissipated energy for laminate 5.

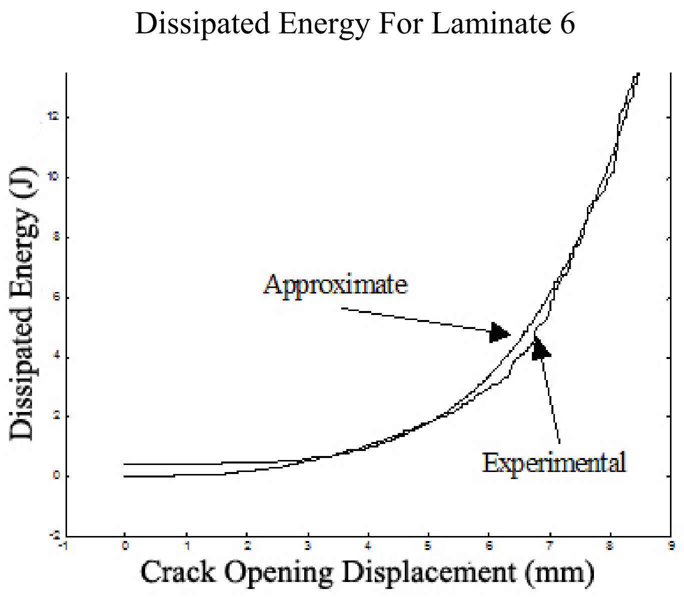


Figure 22. Experimental and approximate dissipated energy for laminate 6.

Results from this compiled study are good. The approximate dissipated energy curves for the previous four laminates are very good and lend confidence to this method.

Laminates one and three represent special cases. These laminates consist of 100% and 93% 90 degree fibers respectively. As these fibers are oriented in the direction of crack growth only resin needs to fracture of the crack to extend. The other four cases had fibers oriented in other directions, thus requiring fibers to break and debond for crack extension to occur which is more common in structural applications. As these phenomena are quite different results for laminates one and three are not as good as the others. These results are shown in figures 23 and 24.

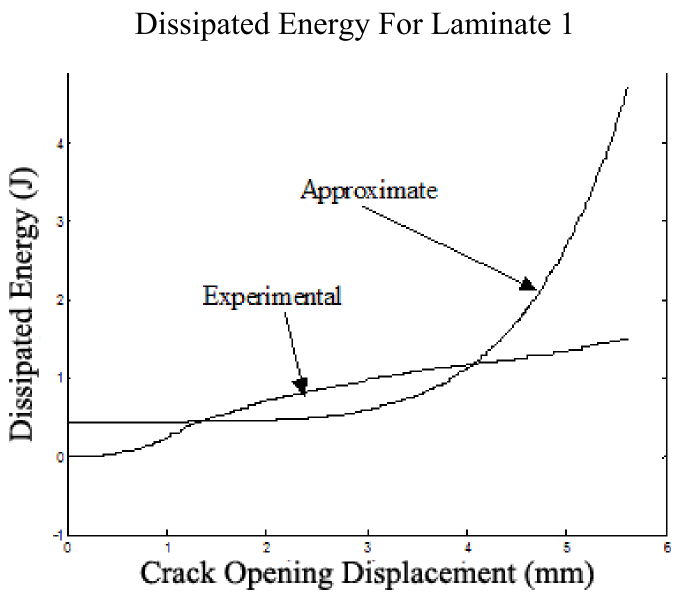


Figure 23. Experimental and approximate dissipated energy for laminate 1.

Dissipated Energy For Laminate 3

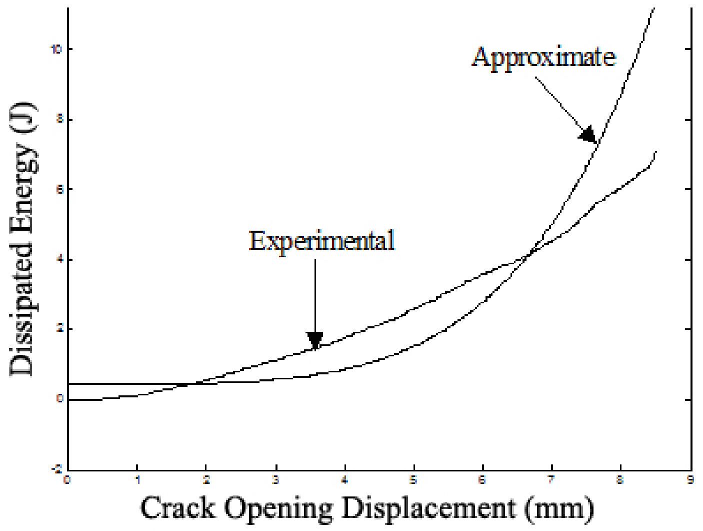


Figure 24 Experimental and approximate dissipated energy for laminate 3.

It should be noted that these special cases can be modeled accurately using traditional fracture mechanics for self similar crack growth through resin [19]. This could be used in conjunction with the dissipated energy function to arrive at a more accurate prediction of behavior.

Fully Predicted Dissipated Energy

The next analysis performed is to compile 5 of the data sets and calculate a C vector, and use this C vector to predict the dissipated energy of the 6th data set. A compiled study of the first 5 data sets was performed to find the C vector. This vector was then used with the Φ function (calculated fully from the finite element data of laminate 6) to arrive at an approximate dissipated energy function. The results of this exercise are shown in figure 25.

Predicted Dissipated Energy vs. Crack Opening Displacement Laminate 9

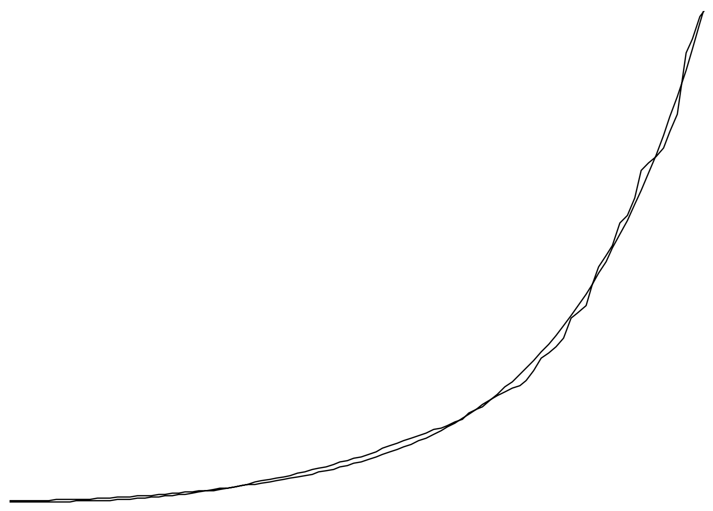


Figure 25. Fully predicted dissipated energy function for laminate 9.

These results are very exciting. They lend confidence to the hypothesis that the dissipated energy function is a material property of the plies. At this point a summary of the procedure is warranted. 1) Samples are made and testing is performed to acquire load vs. displacement data. 2) A finite element model is developed and element strains are resolved into ply principle axes. 3) the dissipated energy function is created using equation 11. 4) a finite element model is created for a CTS sample not used in the data set to derive the dissipated energy function. 5) the dissipated energy function is applied to the finite element model and the approximate dissipated energy is calculated.

It is very encouraging to see that a designer could simply calculate ply principle strains to utilize the dissipated energy function and determine:

1) if energy has been dissipated for a certain loading case

2) the locations in the structure where energy is being dissipated

3) the magnitude of energy dissipated in the structure.

Conclusions from CTS Study

Through preliminary work with compact tension samples, we have shown that the dissipated energy density function has promise as a material property as a function of principal strains in the material. The potential of this technique could mitigate the extensive testing necessary under the current paradigm for ensuring the durability and damage tolerance of a composite structure as illustrated in figure 3.

In this study, the laminate schedule of the compact tension samples was varied with the aim of creating every possible combination of principal strains within the sample. While effective, this approach is time consuming, and does not bound the breadth of potential loadings of a composite. Another approach would be to use a single laminate and instead, apply a mixture of displacements to a sample that would allow the full strain space of the material to be explored, producing a full dissipated energy density function for the breadth of loading in a complex structure.

Such a machine has been developed at Montana State University. The Montana State University in-plane loader (IPL) was inspired by NRL's in-plane loader [12], though much different in design. It utilizes an external loading frame, with off-the-shelf actuators controlled by National Instruments' Labview programs [20]. This testing machine is capable of independently applying any combination of two orthogonal displacements and a rotation to a sample.

CHAPTER 6: STUDY WITH THE IN PLANE LOADER

In Plane Loader

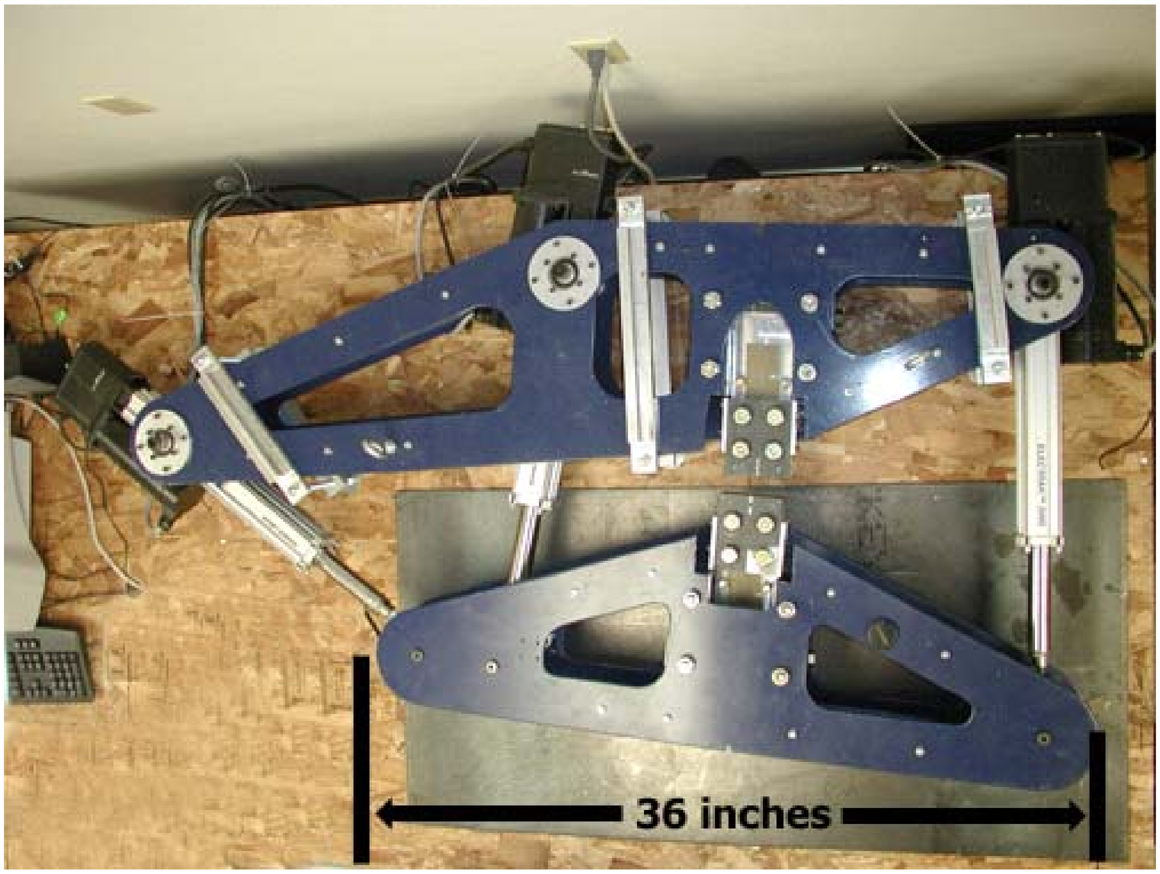


Figure 26. View of the In Plane Loader from above.

The IPL or In Plane Loader is shown in figure 25. This testing machine can provide any combination of tensile, shearing, or rotational displacement in the plane. The large frame of the machine is securely anchored to the table, the smaller frame rolls on casters on a steel plate. This machine was designed originally as an ME 404 senior design project by Eric Booth, Marc Schaff, and Kim Higgins[21]. Construction of the machine was then completed by Will Ritter, Jeremy Kingma, Sarah Grochowski, Eric Booth, and

Bryan Bundy. This machine was built with off the shelf components whenever possible. All of the machining was completed at MSU, the majority of which was performed by students in the student machine shop. Additional machining was performed by the CIM lab and the college of Engineering Tech Services at MSU. The three displacements the IPL can provide are shown in the figure in figure 27.

IPL Displacements

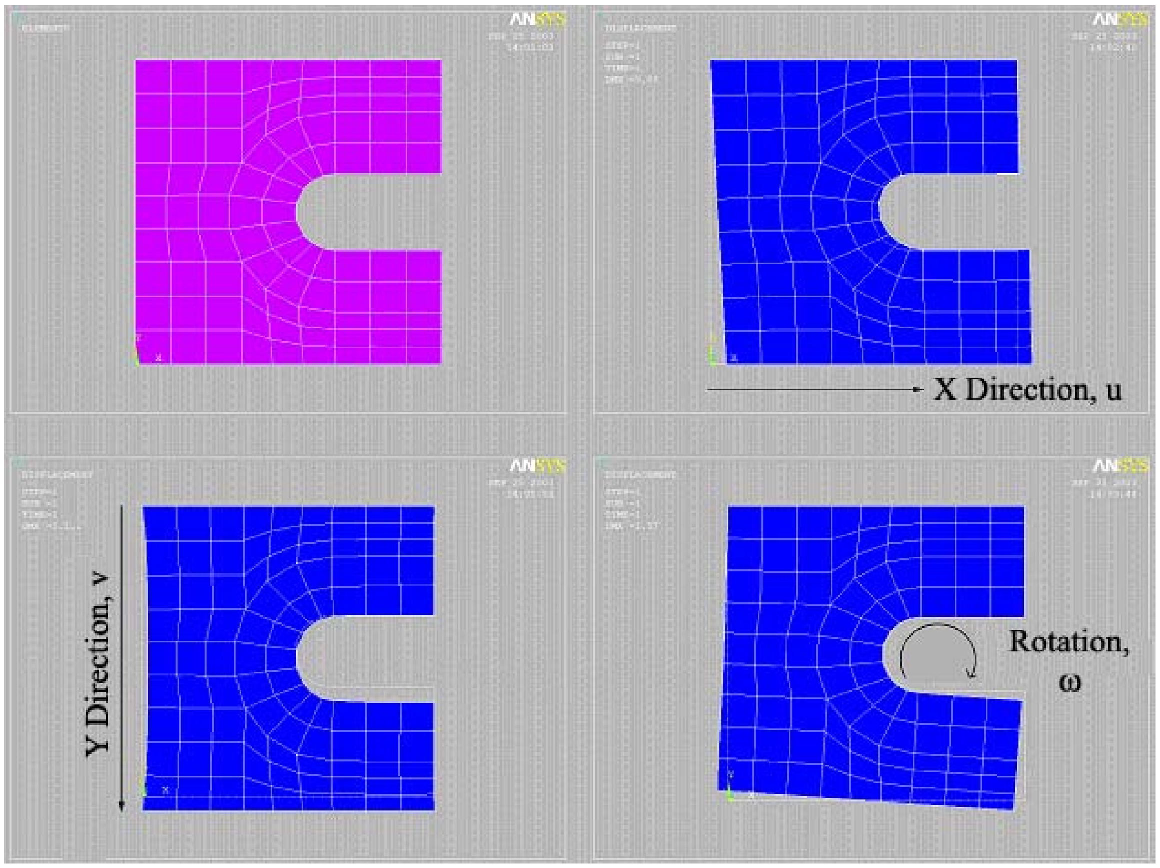


Figure 27. An IPL coupon and the three displacements the IPL can provide.

CHAPTER 7: IPL COUPON DESCRIPTION

A coupon different from the CTS was used with the IPL. This coupon and its dimensions are shown in figure 28.

IPL Coupon

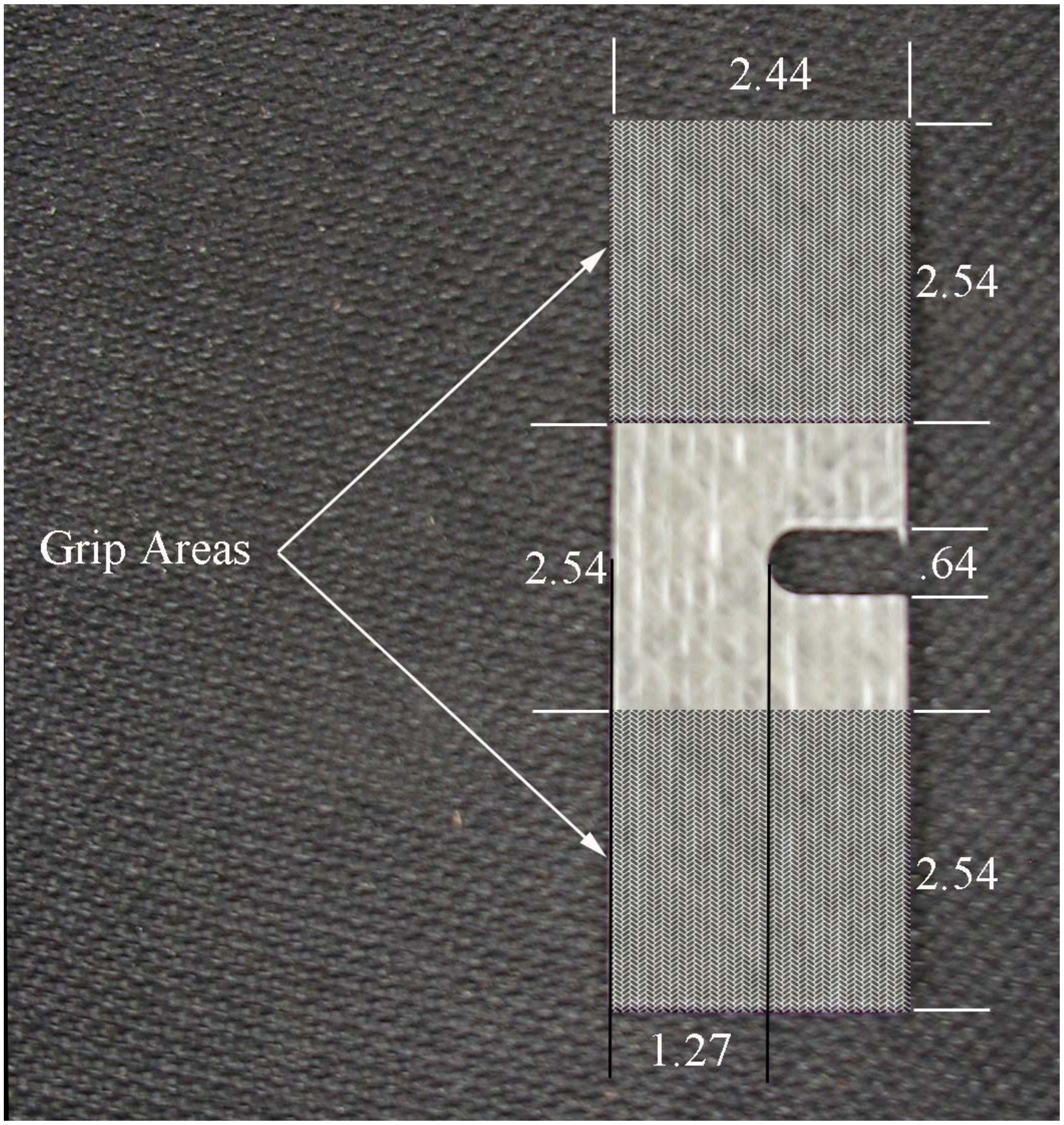


Figure 28. The IPL coupon and its dimensions (cm).

The notch was used to create a stress concentration in the sample far from the grips. Without this notch all of the failures could occur at the grips, providing data that is difficult to interpret. It is important to emphasize that these samples are not fracture mechanics samples, per se. They are analyzed via finite element analysis to develop the dissipated energy function φ in equation one.

These samples were created using resin transfer molding[1]. In this process dry fibrous reinforcements are placed in a mold and then resin is injected into the mold cavity. This process is advantageous as it produces composites with consistent thickness and high quality surface finishes. The samples used for the initial round of testing have a $[0,\pm 45]_s$ laminate schedule. In other words the laminate consists of six plies with the following angular orientations: 0/+45/-45/-45/+45/0. This can be expressed more compactly using the $[0,\pm 45]_s$ style notation. In this notation the s stands for symmetric as the laminate is symmetric about the -45 degree plies. These angular orientations are shown in figure 29.

IPL Coupon Ply Angles

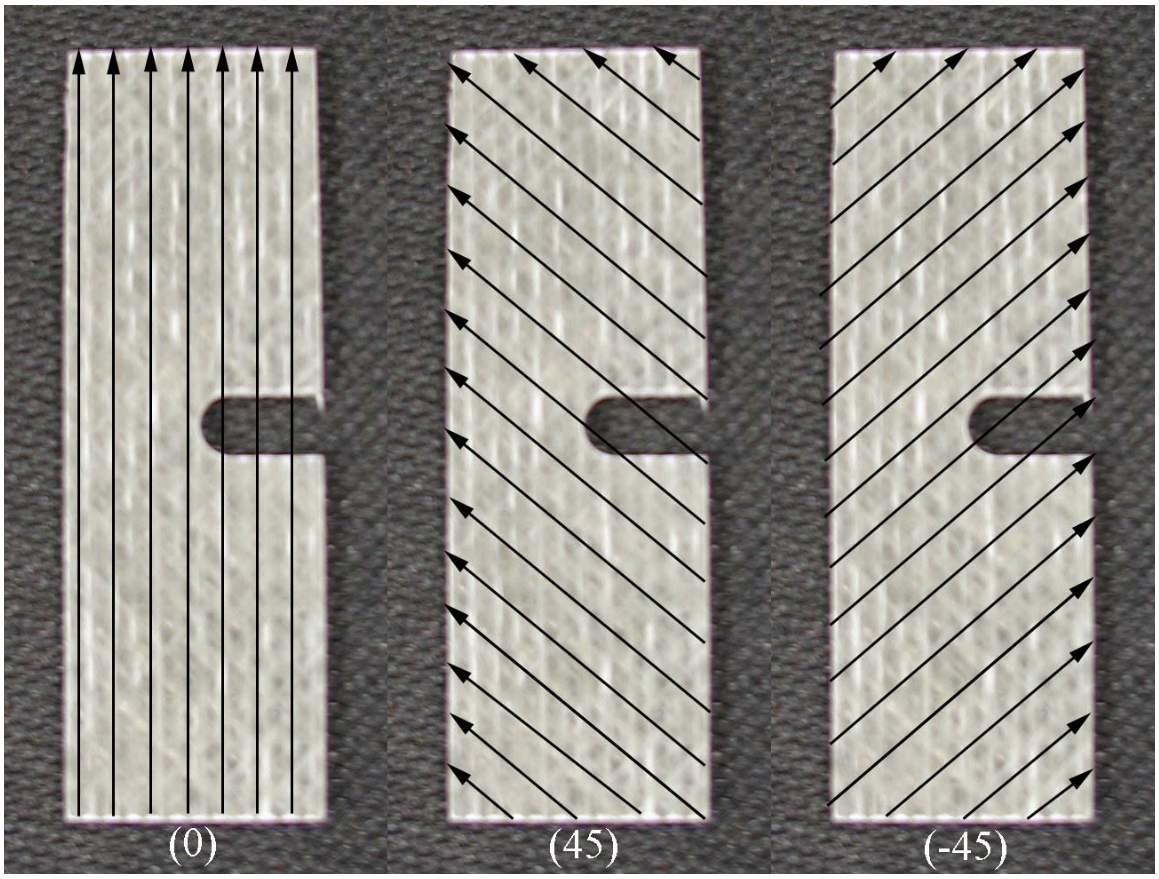


Figure 29. Fabric orientations for IPL coupon.

A coupon in the grips of the IPL is shown in figure 30.

Coupon Mounted In IPL

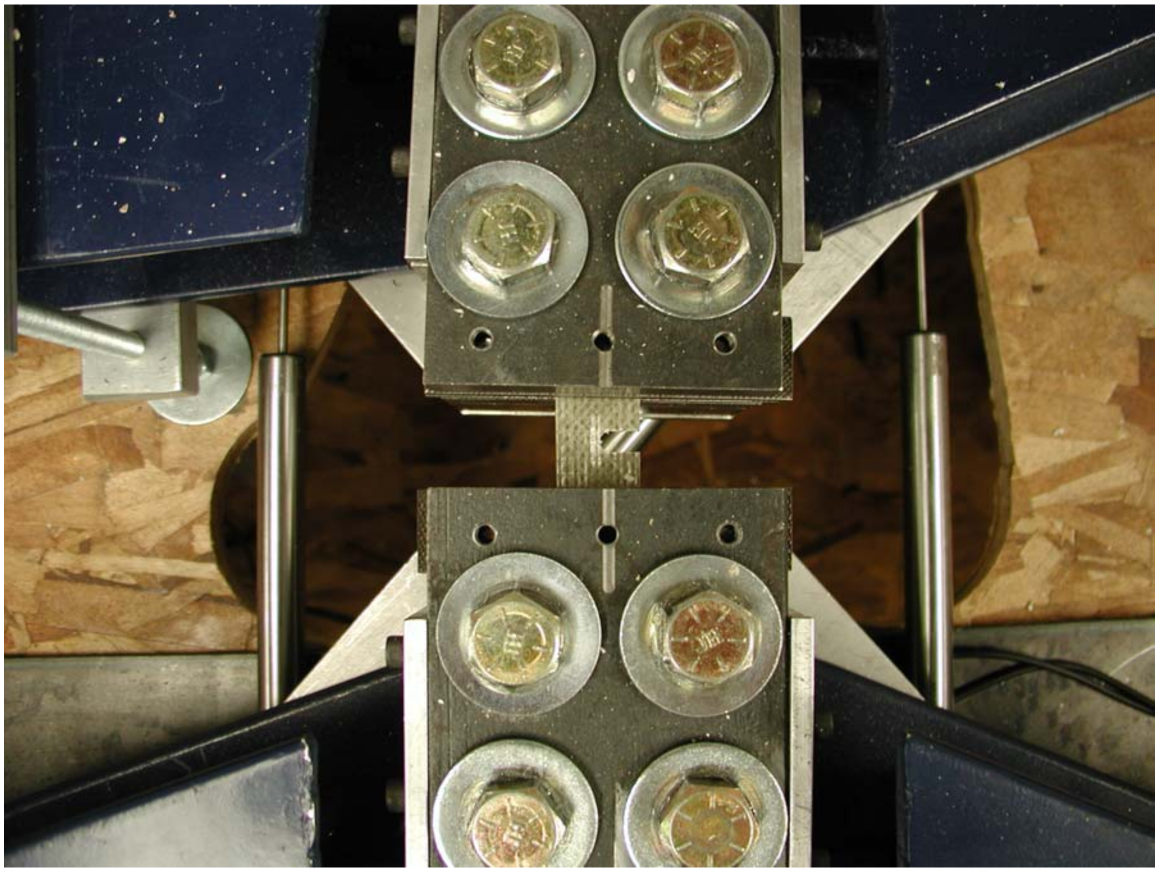


Figure 30. A coupon in the grips of the IPL.

CHAPTER 8: FINITE ELEMENT ANALYSIS OF IPL COUPON

Finite element analysis for this specimen was performed using the Ansys 6.1 finite element package. Code used with Ansys is attached in appendix B. Shell 91 elements were used. These elements are designed to be used with composite structures and have a number of features to make working with composites easier. To create this finite element model first the points of interest from the sample were created. These points were then joined by lines, and then the lines joined into areas. The areas used are shown in figure 31.

Areas Used In Ansys

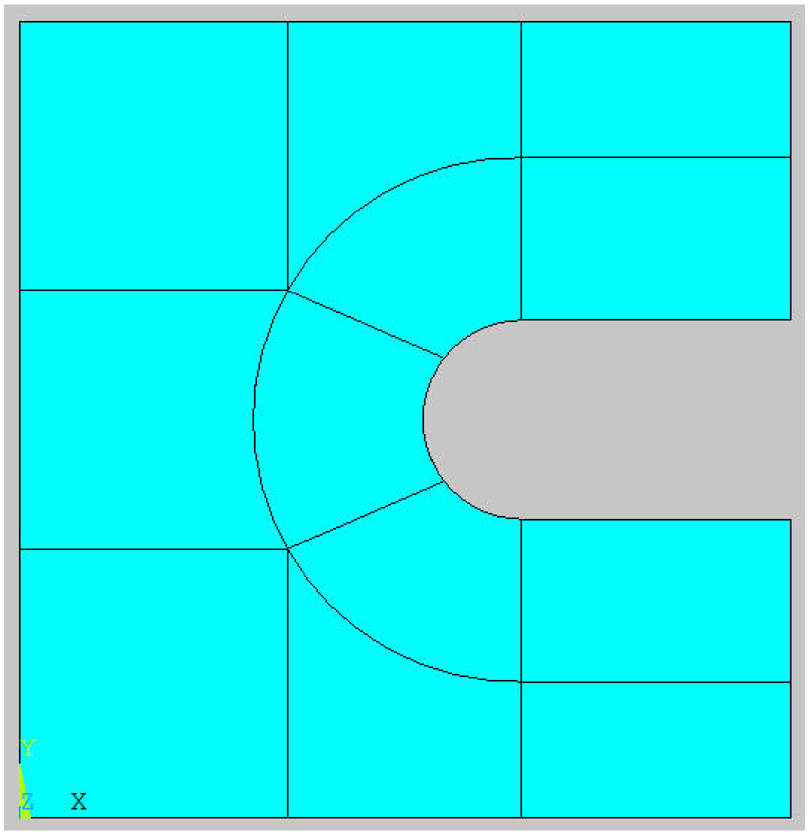


Figure 31. Areas used in creation of the finite element model.

The shapes of these areas were chosen to make the meshing of the model easier. Each area used in this model has four sides. When using a mapped mesh (Ansys help) this allows the same number of elements to be assigned to each edge of the area. Consequently, the model can be built without triangular elements, and lead to more accurate results. If a mapped mesh cannot be used it is said to be a "free" mesh. An example of a mapped and free mesh from the Ansys help guide is shown in figure 32.

Mapped vs. Free Mesh

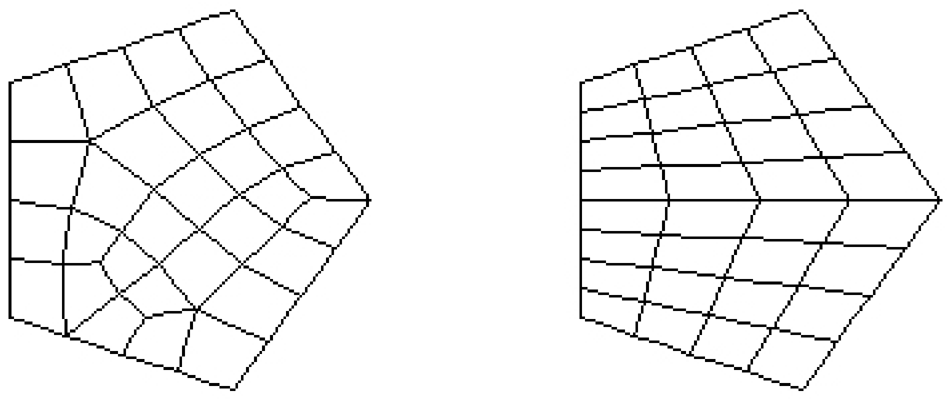


Figure 32. The mesh on the right is a "free" mesh while the mesh on the left is a "mapped" mesh

Mapped meshing was used to control the size and distribution of elements. A mapped mesh of the IPL sample is shown in figure 33.

IPL Mapped Mesh

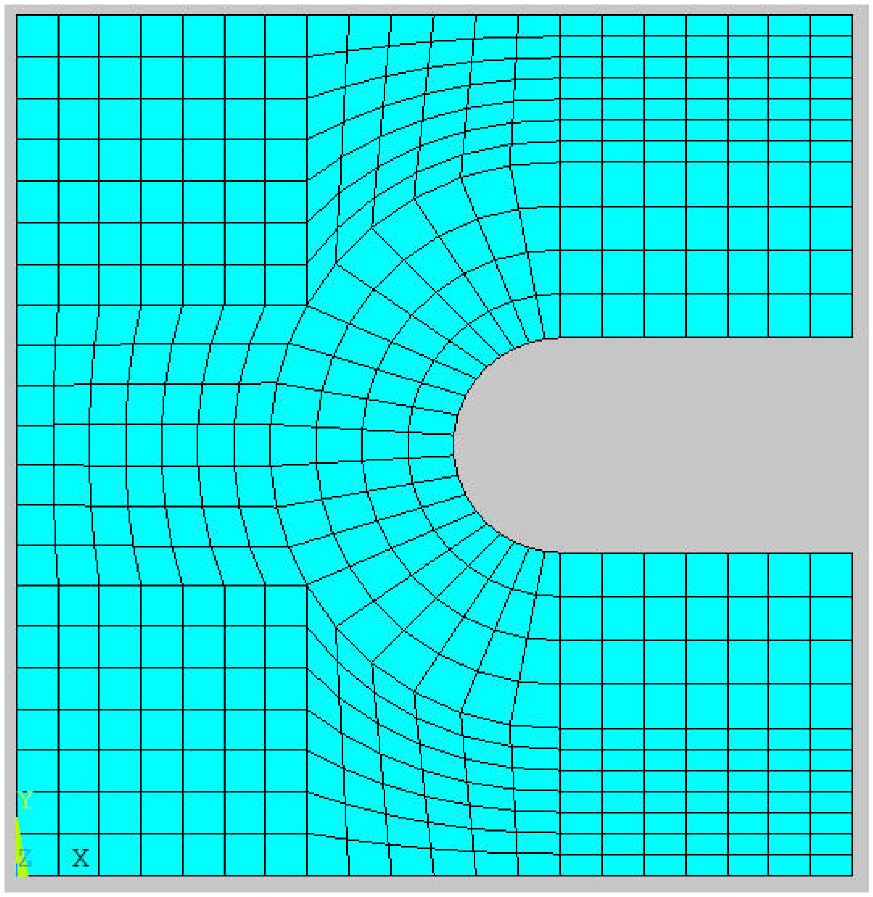


Figure 33. Mapped mesh of IPL sample.

The resolution of this mesh is controlled by a setting the number of elements that are on an edge of the areas. The mesh above was created using 20 elements along the left edge of the sample. Ansys can also show the orientation of the plies used in the model. The plies in this model are shown in figure 34.

Ansys Ply Orientation

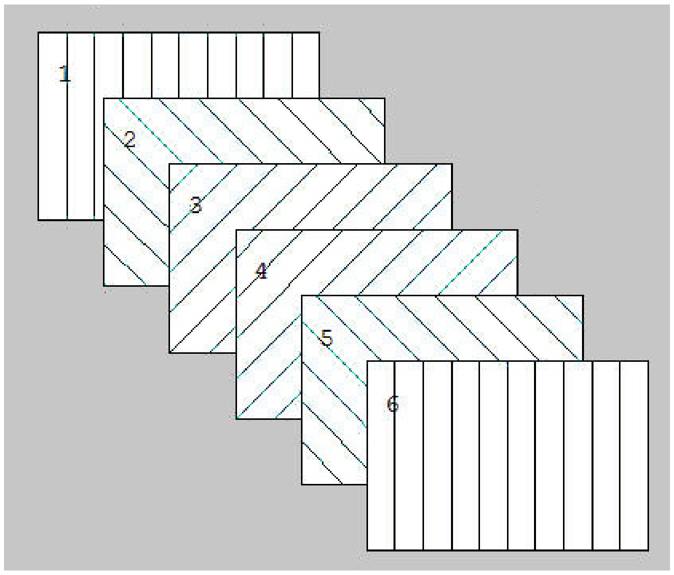


Figure 34. Plies used in the IPL model.

Meshes with 10, 20, 50 and 100 elements along the left edge are shown in figure 35. Using a mapped mesh system like this one allows the resolution of the mesh to be adjusted very easily.

IPL Meshes of Increasing Resolution

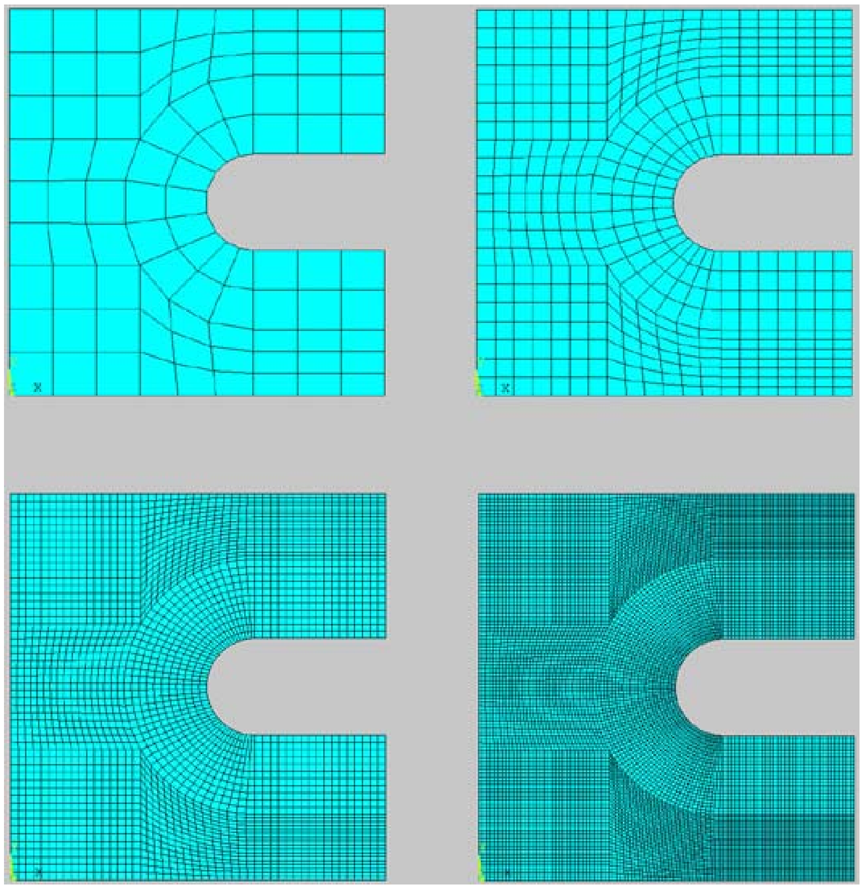


Figure 35. Four mapped meshes of increasing resolution.

At this point a convergence study is necessary to determine when the mesh is converged, or when the resolution of the model is high enough to provide accurate results. The convergence study was performed in the three displacement directions (extension, shearing, and rotation) independently. For each of the three modes the model was displaced some amount and the resolution of the model increased. The three in-plane strains were then found at a query point. When the change between strain values is negligible for an increasing resolution the model is said to be converged. The model displaced in the extension direction (Y) is shown in figure 36.

Extension of IPL Coupon

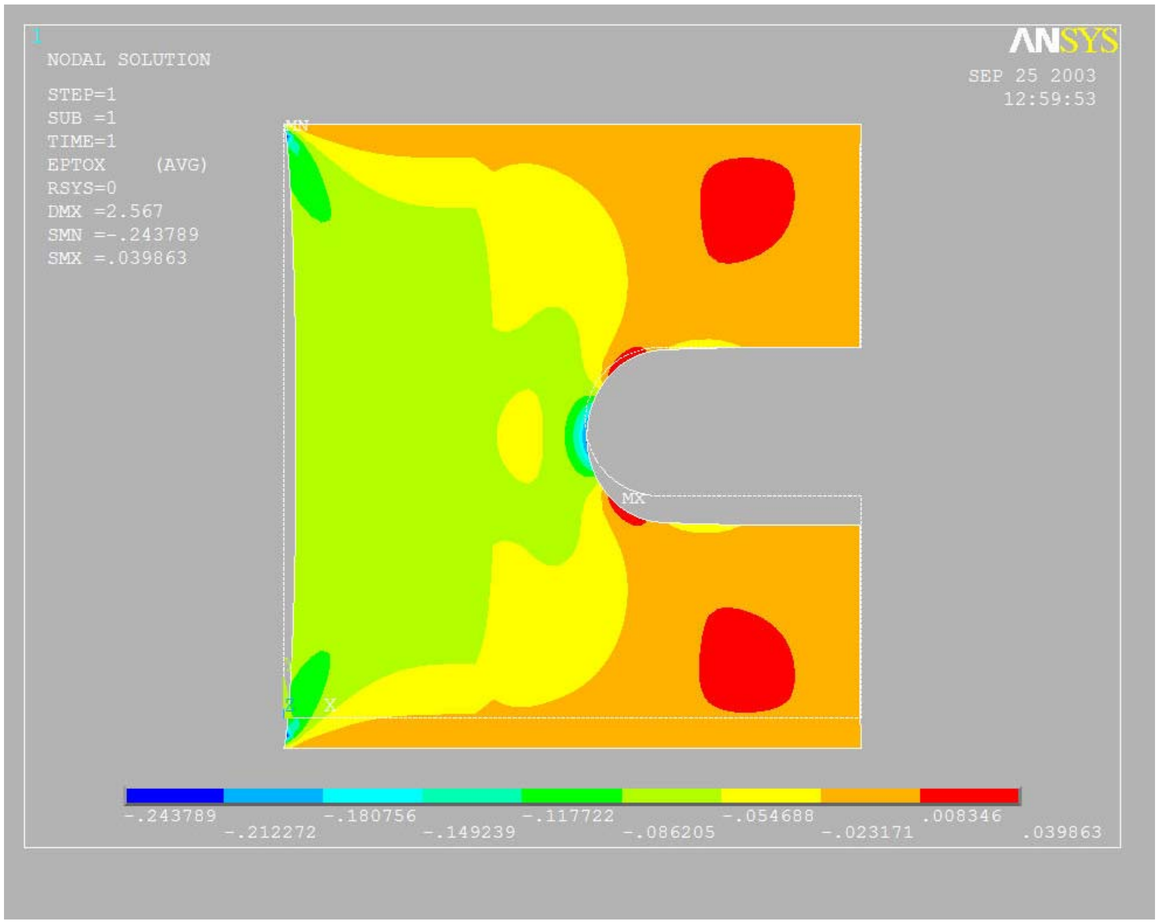


Figure 36. X direction strain of extended sample.

For each of the finite element models the grips were simulated by constraining the nodes on the top and bottom edge of the model in three dimensions. In each case the top edge was fixed and the bottom edge displaced depending on the desired deformation (as this is how the machine moves when you are standing in front of it). A picture of the constrained model is shown in figure 36.

Constrained IPL Model

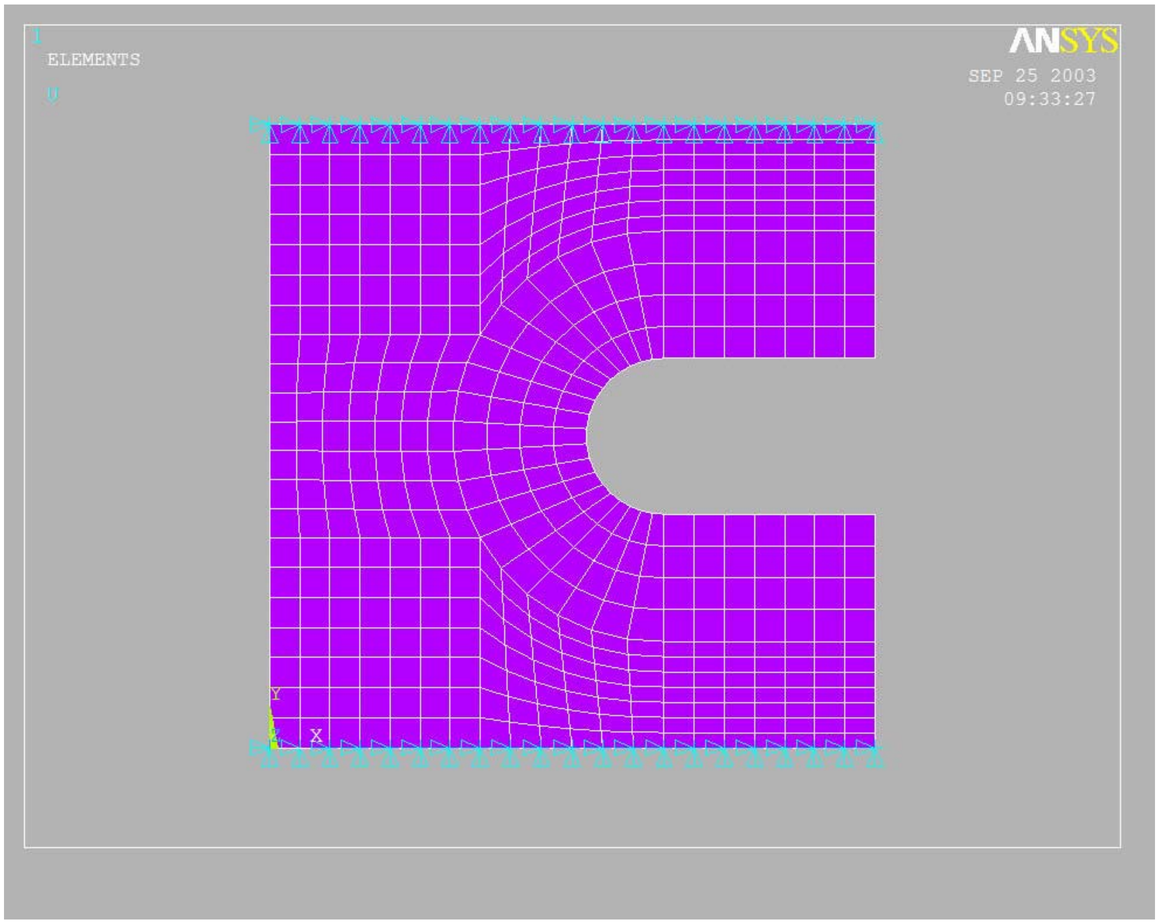


Figure 37. The applied finite element constraints.

The triangles in figure 37 represent a displacement (a displacement of zero in the case of the fixed grip) applied to a node. The query point used for the convergence study was chosen along the edge of the notch on the border of two of the areas used in mesh creation. The query point was chosen at the stress concentration notch as these areas require the highest element resolution for convergence. The point was chosen at the boundary of two areas to guarantee that there would be a node at the point of interest and therefore strains would be available at that point. The chosen query point is shown in figure 38.

Strain Query Point

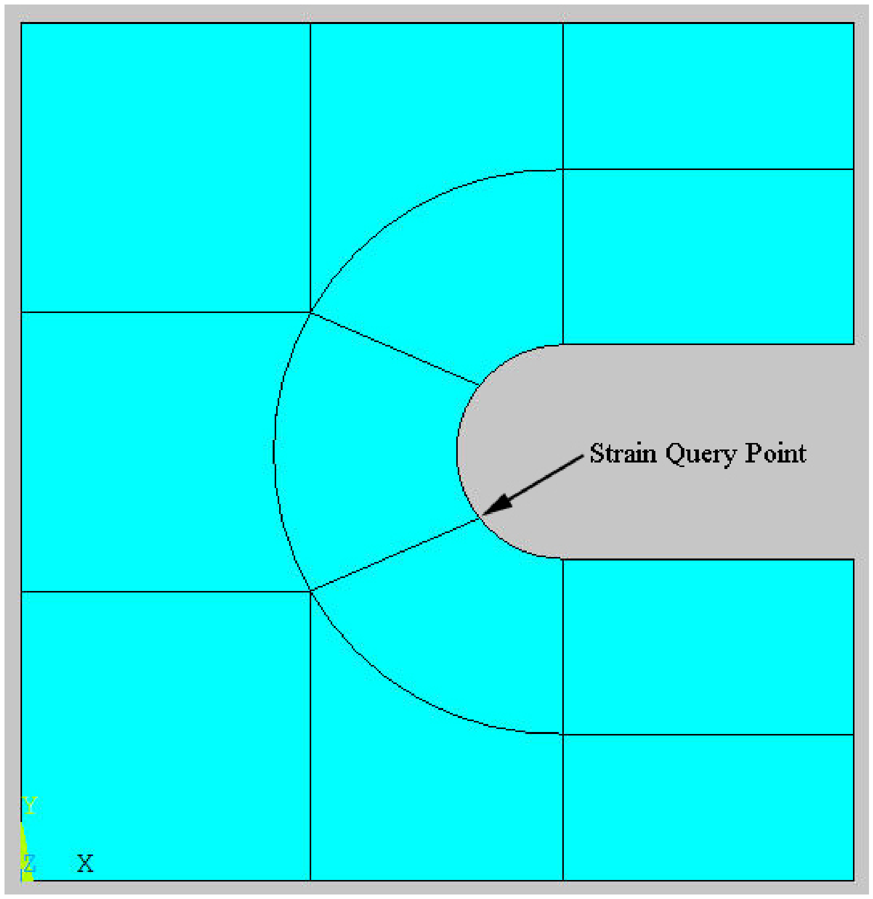


Figure 38. The strain query point for the strain convergence study.

Figures 36, 40 and 42 show that this query point is in an area of high strain. The model was extended ten percent of the gage section length (the gauge section is 2.54 cm) for the extension convergence study. The in-plane strains were then determined for models containing ten to eighty elements on the left edge. Data from this study are shown in Table 1.

Strain Data For Extension

Edge elements	x strain	y strain	shear strain
10	-4.52E-02	8.71E-02	-0.253
20	-3.37E-02	7.15E-02	-0.286
30	-2.86E-02	6.53E-02	-0.297
40	-2.49E-02	6.10E-02	-0.303
50	-2.36E-02	5.95E-02	-0.305
60	-2.28E-02	5.85E-02	-0.306
70	-2.22E-02	5.78E-02	-0.307
80	-2.17E-02	5.73E-02	-0.307

Table 1. Strain convergence data for extension.

A plot of these strains vs. edge element number is shown in figure 39.

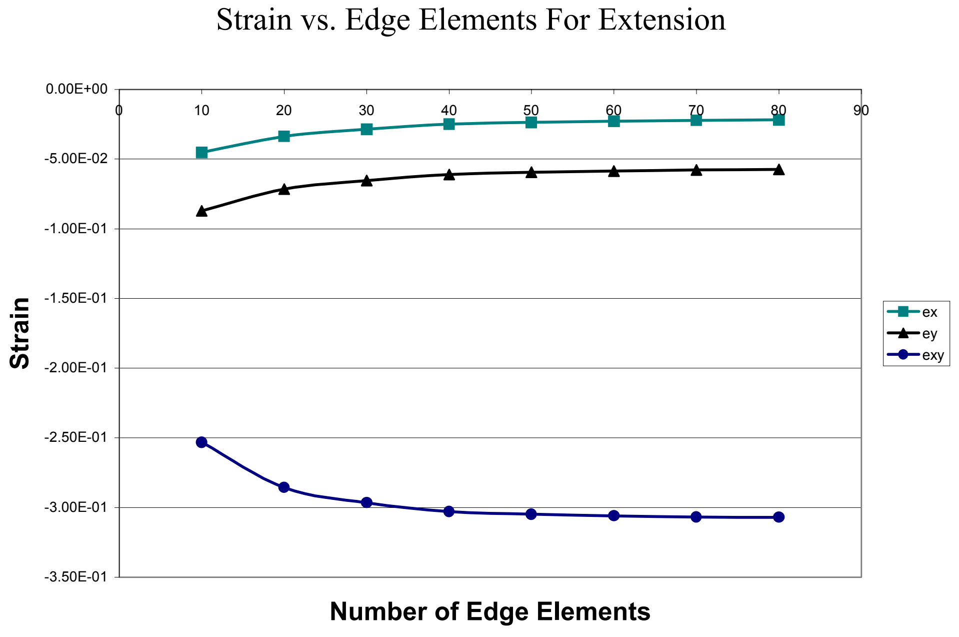


Figure 39. Strain vs. edge element number for extension.

From the tabular and graphic data, it can be seen that the model is asymptotically converging. The strains appear satisfactorily converged with 60 edge elements. As this is the convergence at the stress concentration the remainder of the model should have converged with a lower resolution of elements.

The model deformed due to a shearing (X) deformation is shown in figure 40.

<u>Shearing of IPL Model</u>

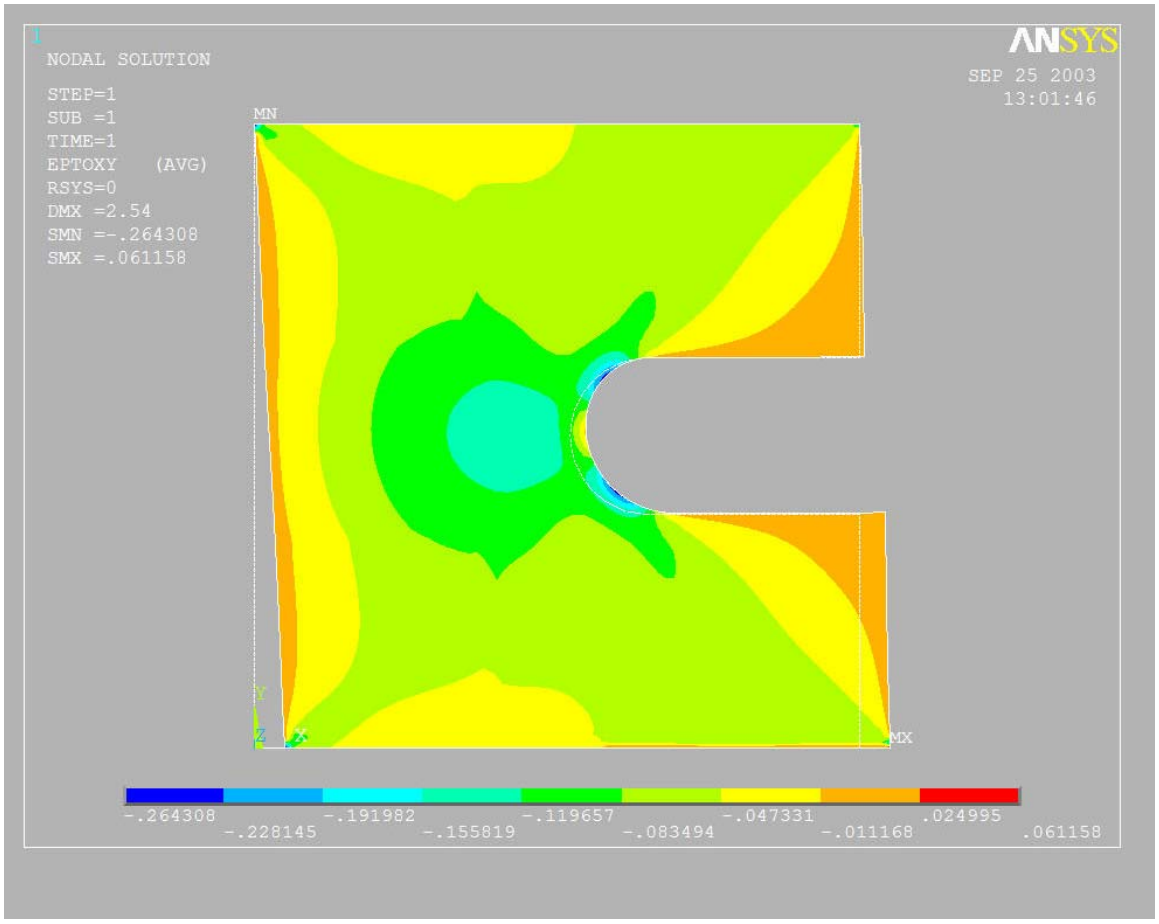


Figure 40. The model experiencing shear deformation.

Data from this convergence study are shown in table 2.

Strain Data For Shearing

Edge elements	x strain	y strain	shear strain
10	3.77E-03	-3.40E-02	0.164
20	1.01E-02	-3.82E-02	0.193
30	1.24E-02	-3.95E-02	0.204
40	1.36E-02	-3.99E-02	0.212
50	1.38E-02	-3.99E-02	0.214
60	1.40E-02	-3.98E-02	0.216
70	1.40E-02	-3.98E-02	0.217
80	1.40E-02	-3.97E-02	0.218

Table 2. Strain data for shear deformation.

A plot of this data is shown in figure 41.

Strain vs. Edge Elements For Shearing

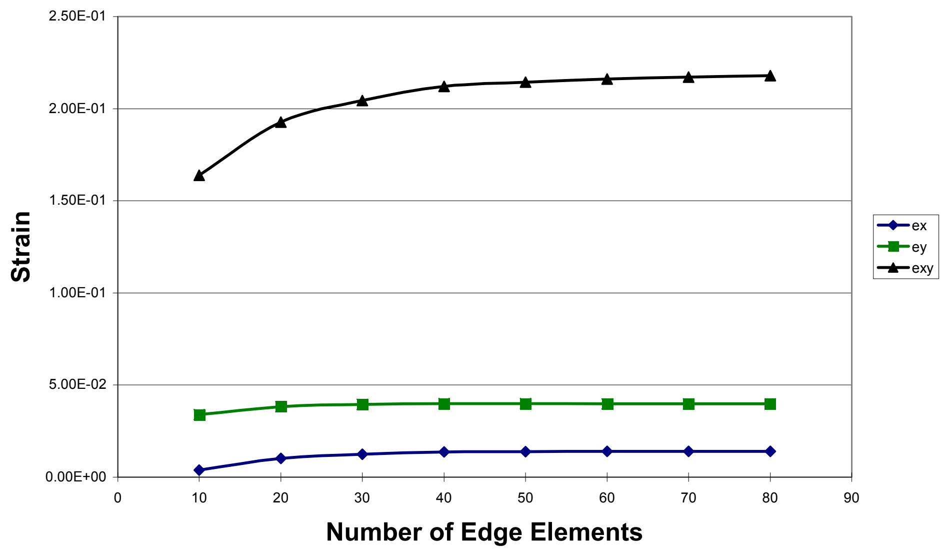


Figure 41. Strain data for shear convergence.

Again with this deformation the model seems satisfactorily converged with 60 edge elements.

The model undergoing rotation is shown in figure 42.

Rotation of IPL Model

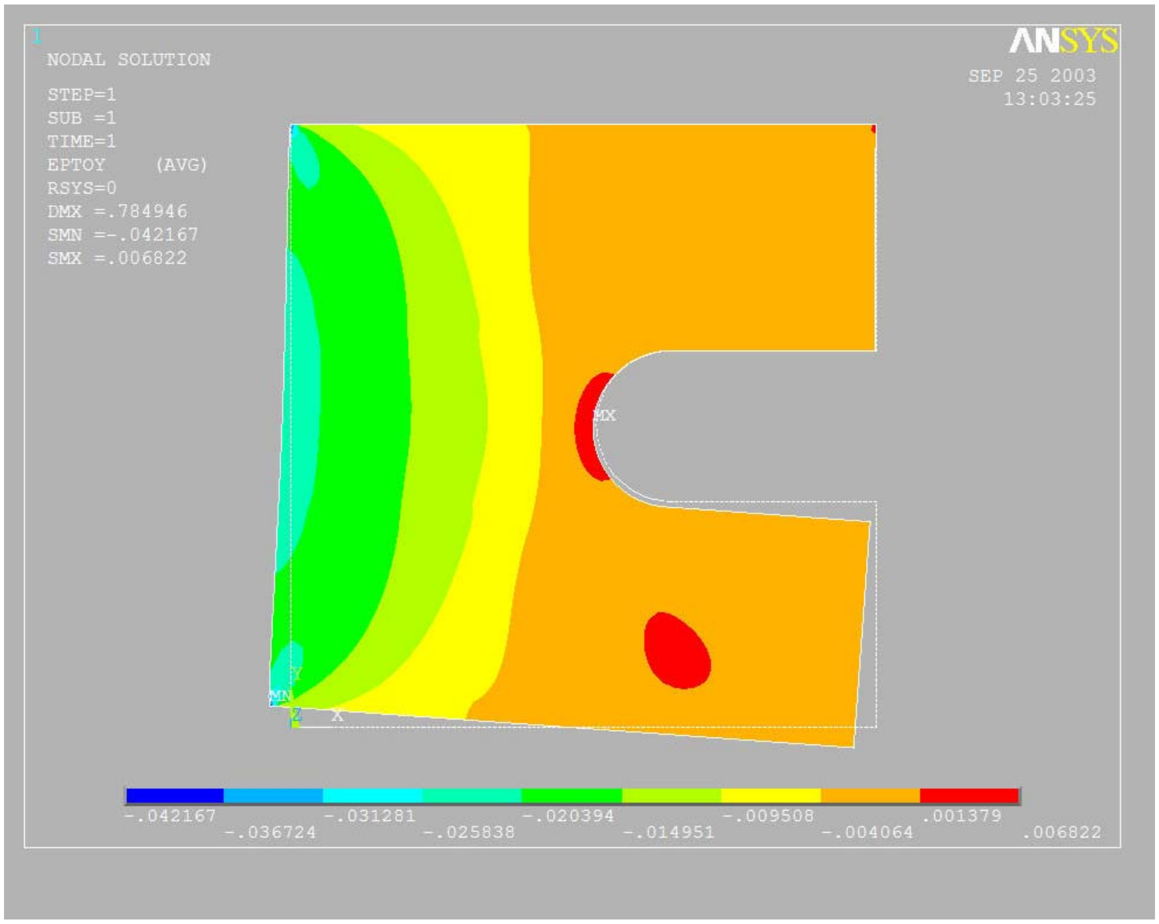


Figure 42. The model undergoing rotation.

Data from this convergence study are shown in table 3.

Strain Data for Rotation

Edge elements	x strain	y strain	shear strain
10	-1.44E-03	3.92E-03	-1.30E-02
20	-1.37E-03	3.60E-03	-1.58E-02
30	-1.31E-03	3.46E-03	-1.67E-02
40	-1.25E-03	3.35E-03	-1.73E-02
50	-1.23E-03	3.31E-03	-1.74E-02
60	-1.21E-03	3.28E-03	-1.75E-02
70	-1.20E-03	3.26E-03	-1.76E-02
80	-1.19E-03	3.25E-03	-1.76E-02

Table 3. Data from the rotation convergence study.

A plot of this data is shown in figure 43.

Strain vs. Edge Elements For Rotation

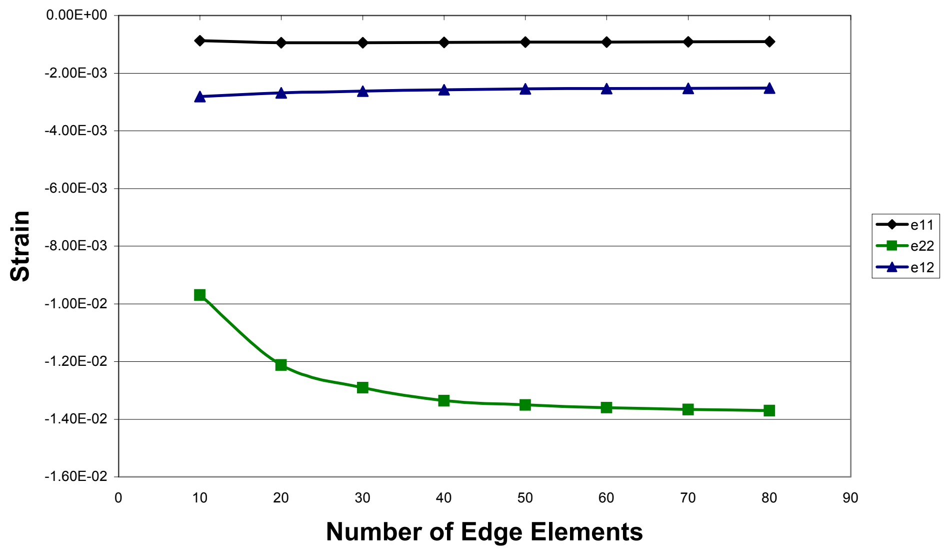


Figure 43. Strains for rotation convergence study.

This model as well seems reasonably converged with 60 edge elements. As this amount works well for each of the three models it was used for the remainder of the study. There is an interest in limiting the number of elements as this allows Ansys and later Matlab to run at a faster rate.

These are the only three models that need be created for the initial dissipated energy study using the IPL. The strain field at an arbitrary displacement in the IPL can be created using the principle of superposition. This is shown in figure 44.

Principle of Superposition

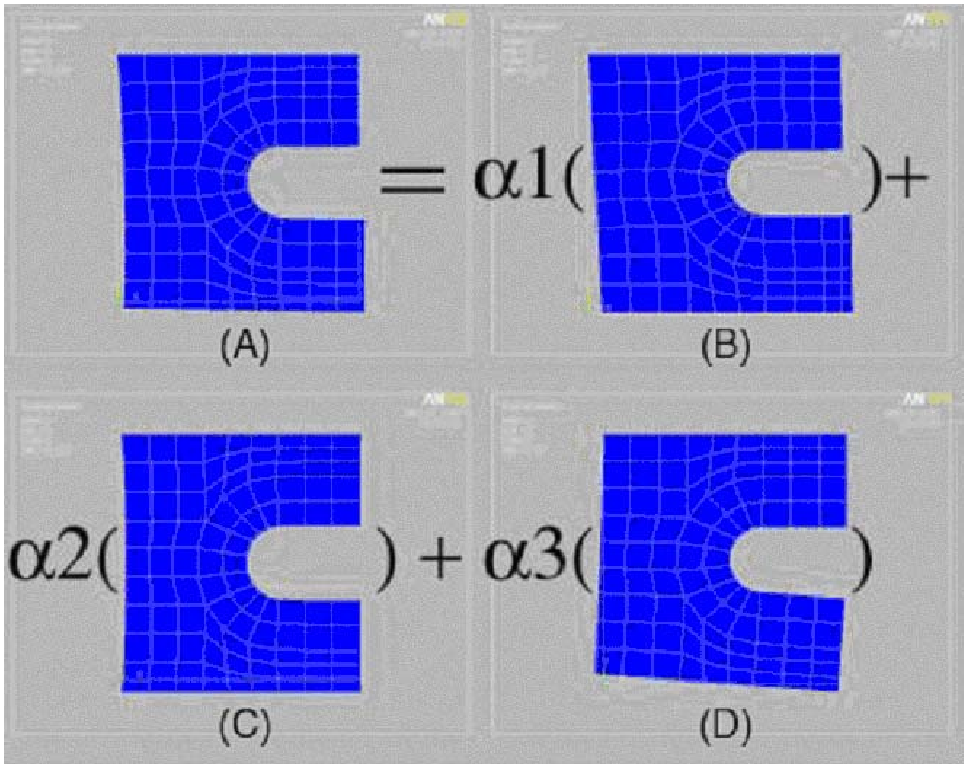


Figure 44. Here an arbitrary displacement is composed of the three orthogonal displacements provided by the IPL.

The principle of superposition allows an arbitrary displacement (fig 44(a)) to be expressed as a scaled combination of the three in-plane displacements (fig 44 b,c and d).

In this case the scale factors are $\alpha 1$,2 and 3. Similarly, the strains in any element the arbitrarily displaced model (fig xx (a)) can be found by using the same scale factors on the strains of the corresponding elements in the models of the orthogonal displacements. Explicitly:

$$e_{11}a = \alpha 1 * e_{11}x + \alpha 2 * e_{11}y + \alpha 3 * e_{11}r$$

$$e_{22}a = \alpha 1 * e_{22}x + \alpha 2 * e_{22}y + \alpha 3 * e_{22}r$$

$$e_{12}a = \alpha 1 * e_{12}x + \alpha 2 * e_{12}y + \alpha 3 * e_{12}r$$

(6)

where a denotes the arbitrary displacement, x denotes the strains from the model with only x displacement, y denotes the strains from the model with only y displacement, and r denotes the model with only rotational displacement.

This property is important for this study as only three strain fields need be imported in Matlab for performing the least squares fit for any displacement. If this were not the case a separate finite element model would have to be created for each displacement created in the IPL.

CHAPTER 9: IPL EXPERIMENTAL METHODS

The IPL's motion is governed by three non-orthogonal actuators shown in figure 45.

Actuator Detail

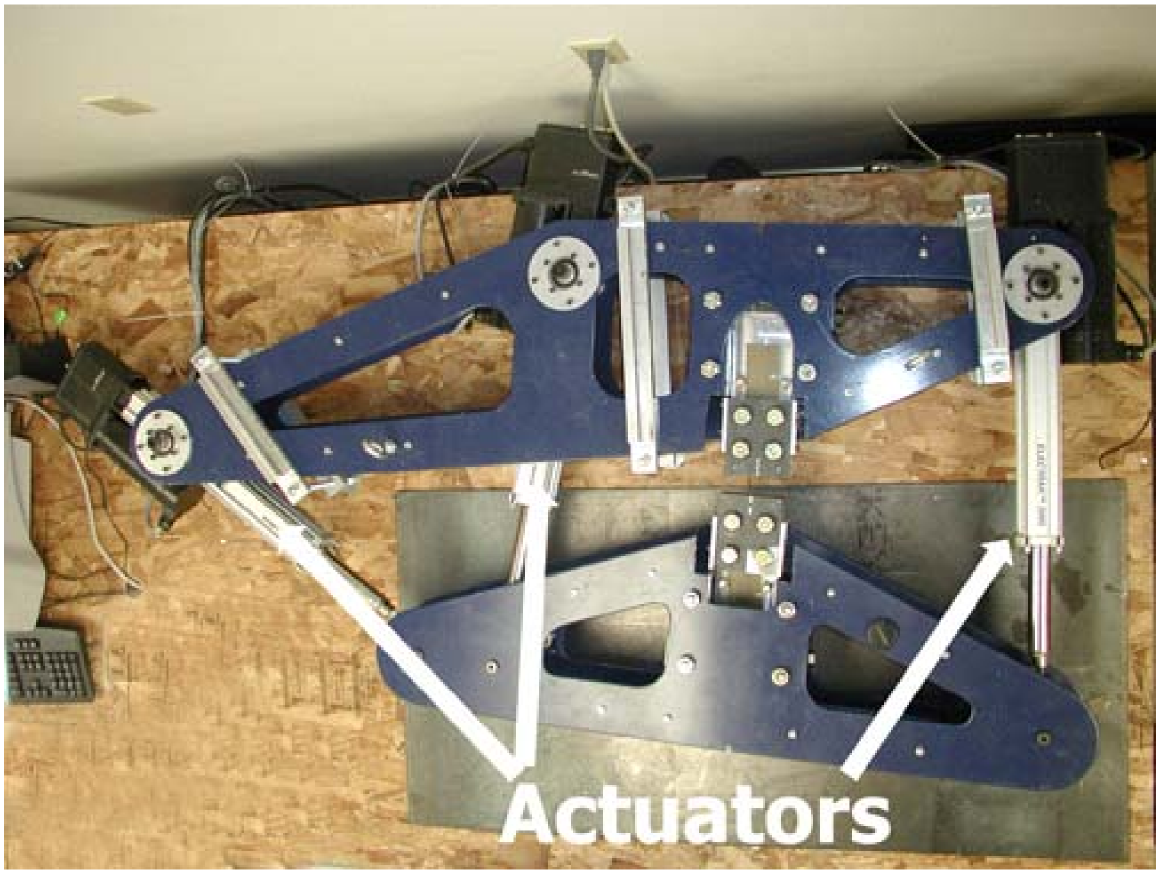


Figure 45. The three actuators that control the IPL.

The non-orthogonality of the actuators allows any unique combination of displacements to be created. This non-orthogonality also means that control of the machine is non-linear and coupled for any of the orthogonal displacements u, v, and ω. The dimensions of the machine and the location of the actuator pivots must be known very well as the motion of the machine has to be controlled very precisely to provide useful data. In order to accomplish the measurement a digitization arm was used. This is shown in figure 46.

Digitization of IPL

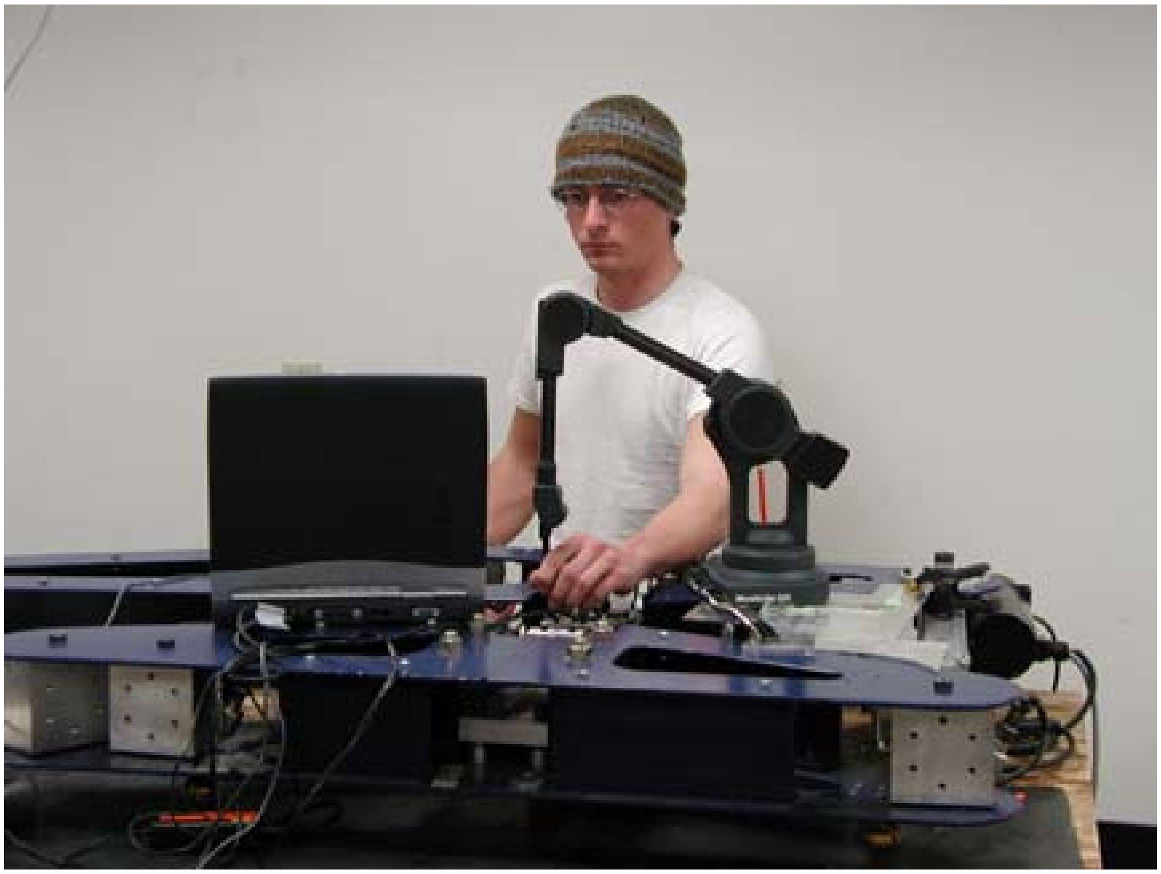


Figure 46. Using a digitization arm to measure the IPL.

This digitization machine was used in conjunction with Rhino software [22] and Pro/E [23] to create a model of the IPL. This software was extremely useful for performing the various measurements for a desired testing scenario. It will be used in the future for configuring different gauge lengths. Dimensions from digitization are attached in appendix C.

The Pro/E model can also be used to determine a number of displacements and to create a material test. The tests used in this study contained 100 displacement points, using the Pro/E model to create these would be too time consuming. A program was created in Maple mathematics software [24] to batch a number of displacement positions. This program also accomplishes the task of determining the angles of the actuators with respect to the grips at each displacement point. This is necessary to resolve the loads measured with the load cells into the x and y forces and the moment at the center of the gauge length. This code is attached in appendix D.

Seven load paths were used in the IPL data sets. These paths are:

1. Pure x displacement (u)
2. Pure y displacement (v)
3. Pure rotation (ω)
4. Combination of x and y displacement (u + v)
5. Combination of x displacement and rotation (u + ω)
6. Combination of y displacement and rotation (v + ω)
7. Combination of all three displacements (u + v + ω)

IPL Software and Control

The motion of the IPL is controlled is controlled by National Instrument's Labview software and a PCI board. This board is capable of controlling four stepper axes. Conveniently this board also has four 12 bit digital inputs. These inputs are used to acquire the data from the load cells. These pancake load cells are in line with each actuator. A load cell and actuator combination is shown in Figure 47.

Actuator and Load Cell Detail

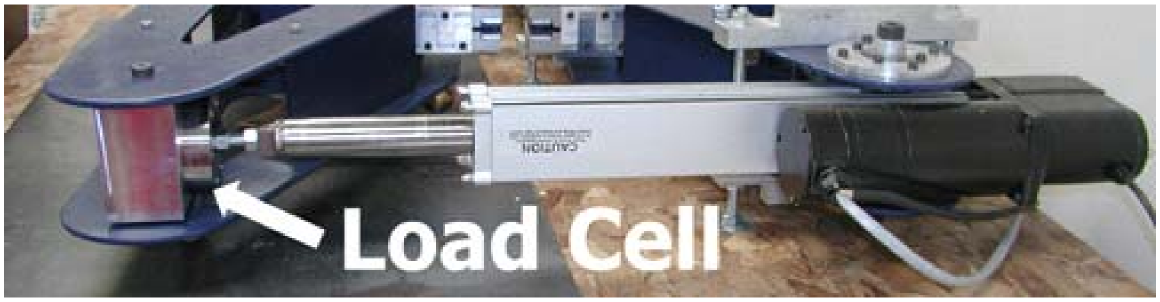


Figure 47. One of the IPL's load cells.

The PCI inputs read a voltage from -10 to 10 volts. Being 12 bit inputs the -10 to 10 voltage range is divided into $2^{12} = 4096$ levels. This is convenient as the actuators are capable of producing forces of $\pm$ 2000 lb, therefore the resolution of the loads is about one pound.

The load cells used with the IPL produce a voltage of 15 millivolts per 1000 pounds of load. In order for these load cells to work with the PCI board an amplifier was designed and built [25]. This amplifier is shown in figure 48.

Load Cell Amplifier

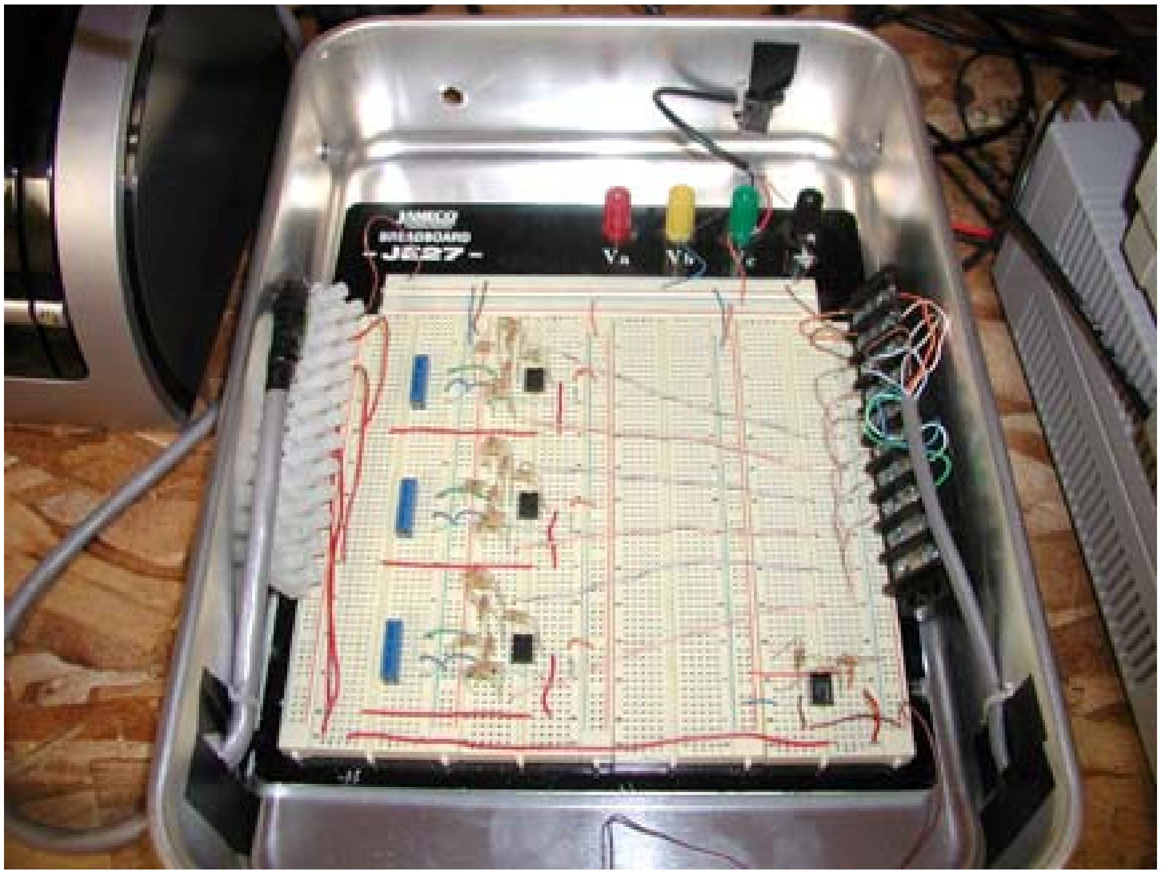


Figure 48. Circuit created to amplify load cell signals.

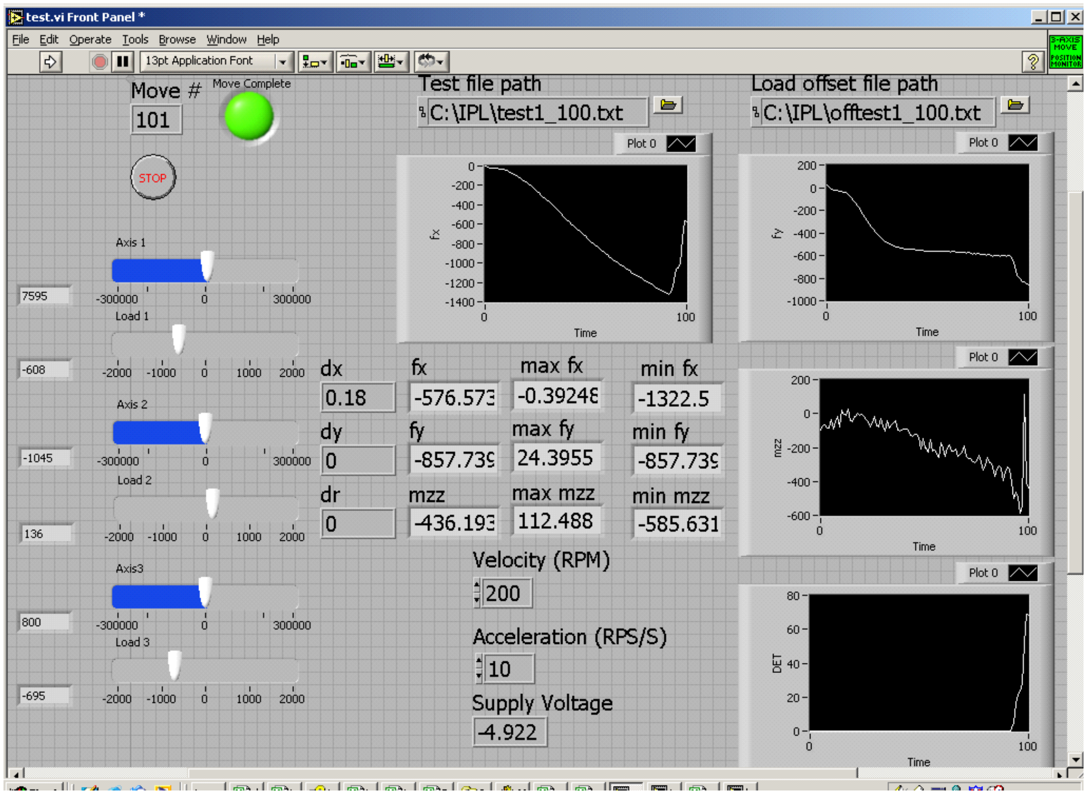
This amplifier brings the load cell signals up to the -10 to 10 voltage range. The software that controls the IPL was derived from sample software available from National Instrument's website. This proved to be a very useful resource, as their library is quite complete. Separate files that they had to move actuators simultaneously as well as read from the digital input channels were combined along with other codes to arrive at the testing software. The main functions of the testing software are as follows.

- Send the successive actuator positions, monitor the current actuator positions

- Set the actuator speeds and accelerations
- Read the voltages from the load cells
- Resolve the loads into x, y and moment components and display graphically
- Calculate the dissipated energy during the test and display graphically
- Assemble all of the test data into one file and save it to a specified location

The interface for the testing software is shown in figure 49.

IPL Control Panel



Move # 101

Test file path C:\IPL\test1_100.txt

Load offset file path C:\IPL\offtest1_100.txt

7595
-608
-1045
136
800
-695

dx 0.18
dy 0
dr 0

fx -576.573
max fx -0.39248
min fx -1322.5

fy -857.739
max fy 24.3955
min fy -857.739

mzz -436.193
max mzz 112.488
min mzz -585.631

Velocity (RPM) 200

Acceleration (RPS/S) 10

Supply Voltage -4.922

Figure 49. The IPL testing interface written in labview.

Labview code used to control the IPL is attached in appendix E.

In each case the samples were loaded into the IPL using a gauge block to ensure that the sample was aligned in the same spot on the grips. A clamped sample is shown in figure 50.

Clamp Detail

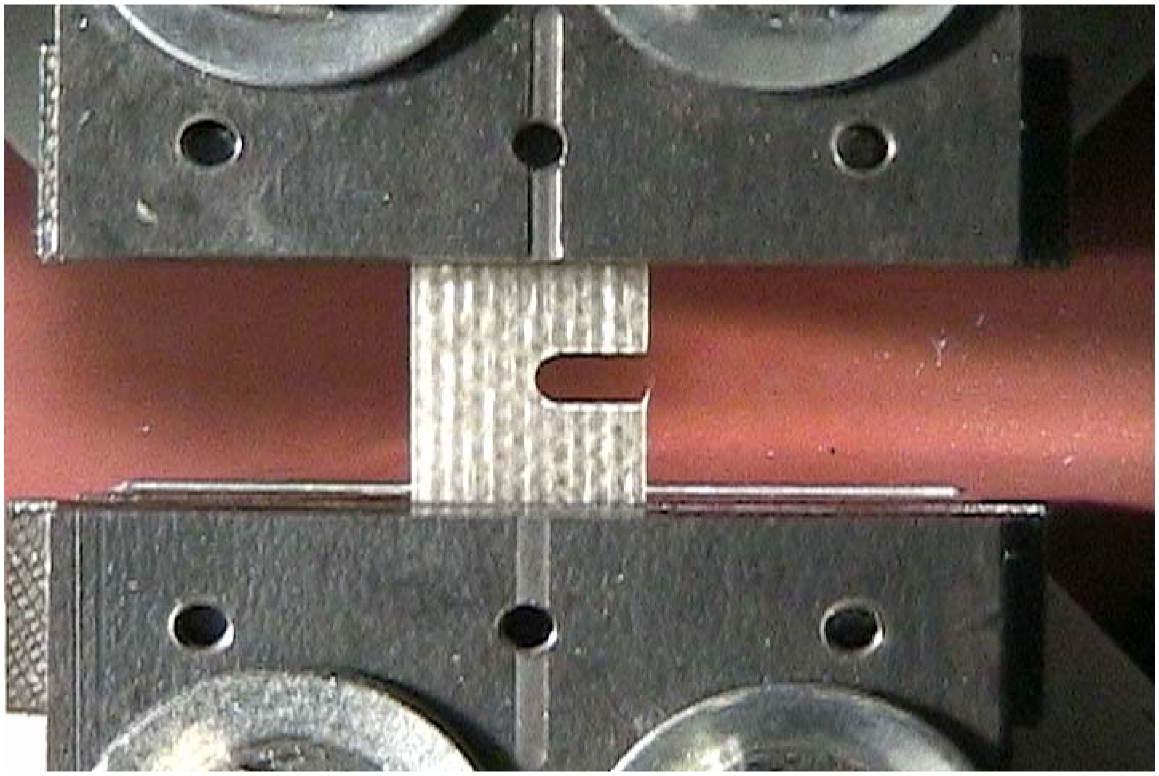


Figure 50. A sample clamped in the IPL.

IPL Testing Images

Progressive images from the IPL tests are shown in the following figures. A pure x displacement test is shown in figure 51.

X Displacement (u)

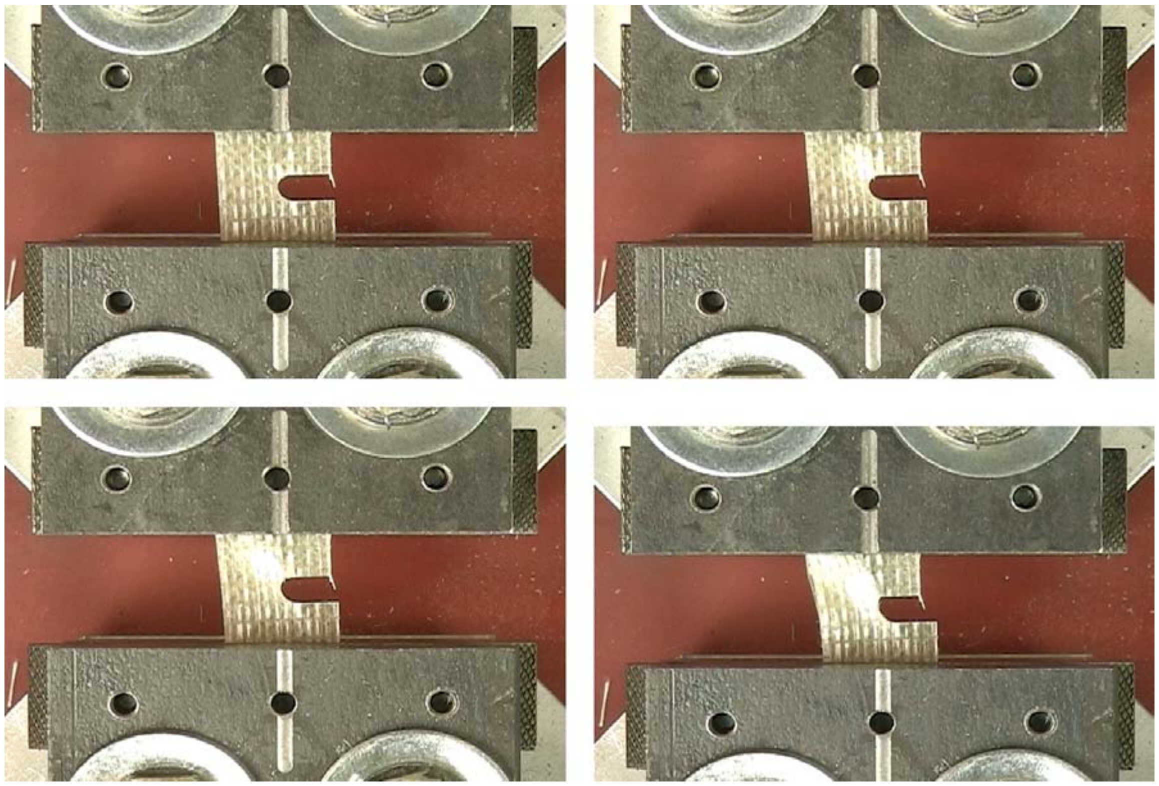


Figure 51. Progressive pictures from an x displacement test.

Note the evolution of damage from the notch of the specimen to the edge of the grip region. At the end of this sequence the sample has experienced extensive damage beyond a useful limit for structural applications.

A pure y displacement test is shown in figure 52.

Y Displacement (v)

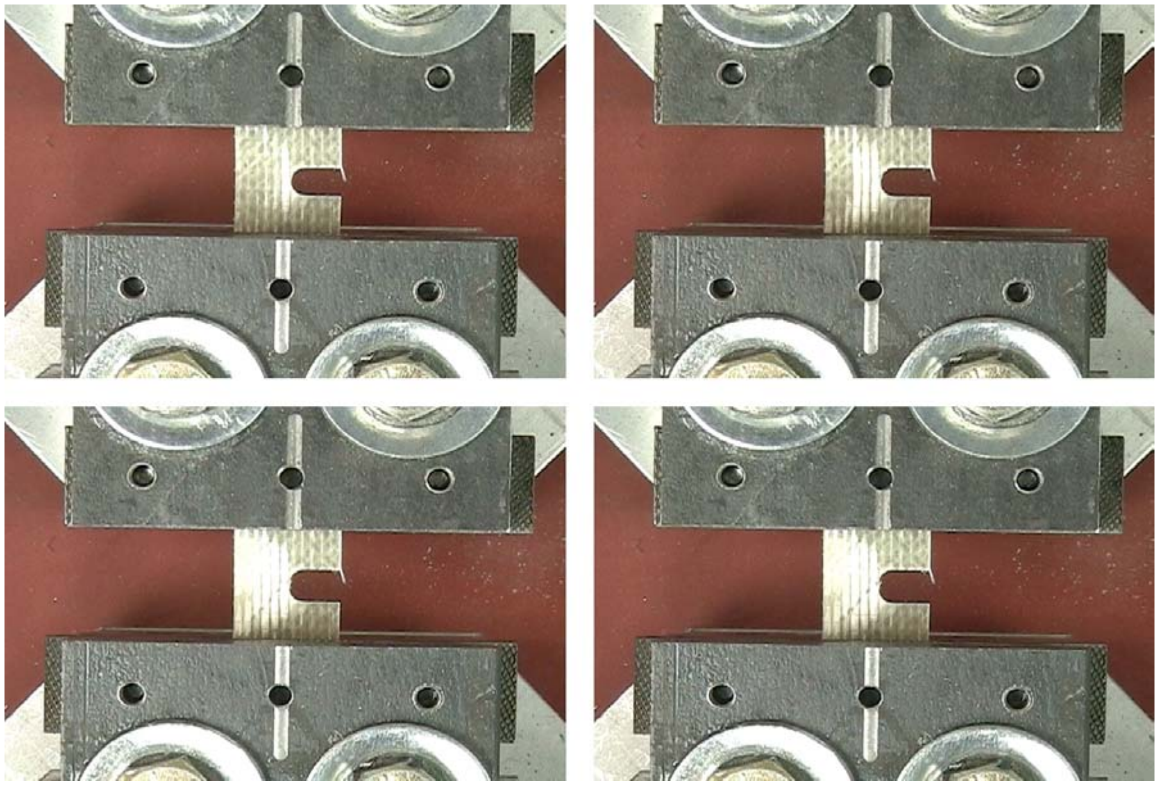


Figure 52. A pure y displacement test.

Note the relatively symmetric damage development. Also note the rather small displacement compared to figure 52. This is a consequence of this test being controlled by fiber damage rather than resin damage, which dominates the x direction test. This contrast between fiber and resin failures will be discussed in a later section.

A pure rotation test is shown in figure 53.

Rotation (ω)

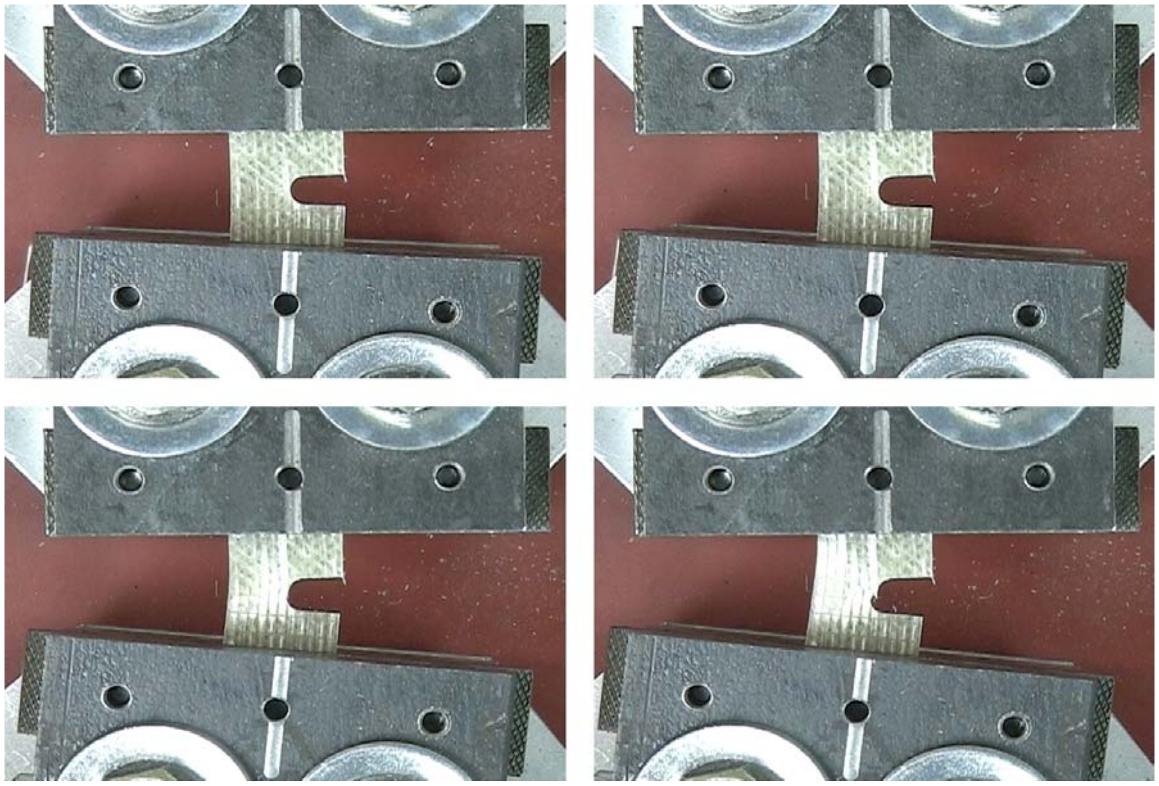


Figure 53. Progressive pictures of a rotation test.

Note the relatively large displacement when compared to the y displacement test.

Test six is an example of one of the mixed modes. This test is shown in figure 54. Test six is a combination of y displacement and rotation.

Test 6 (v + ω)

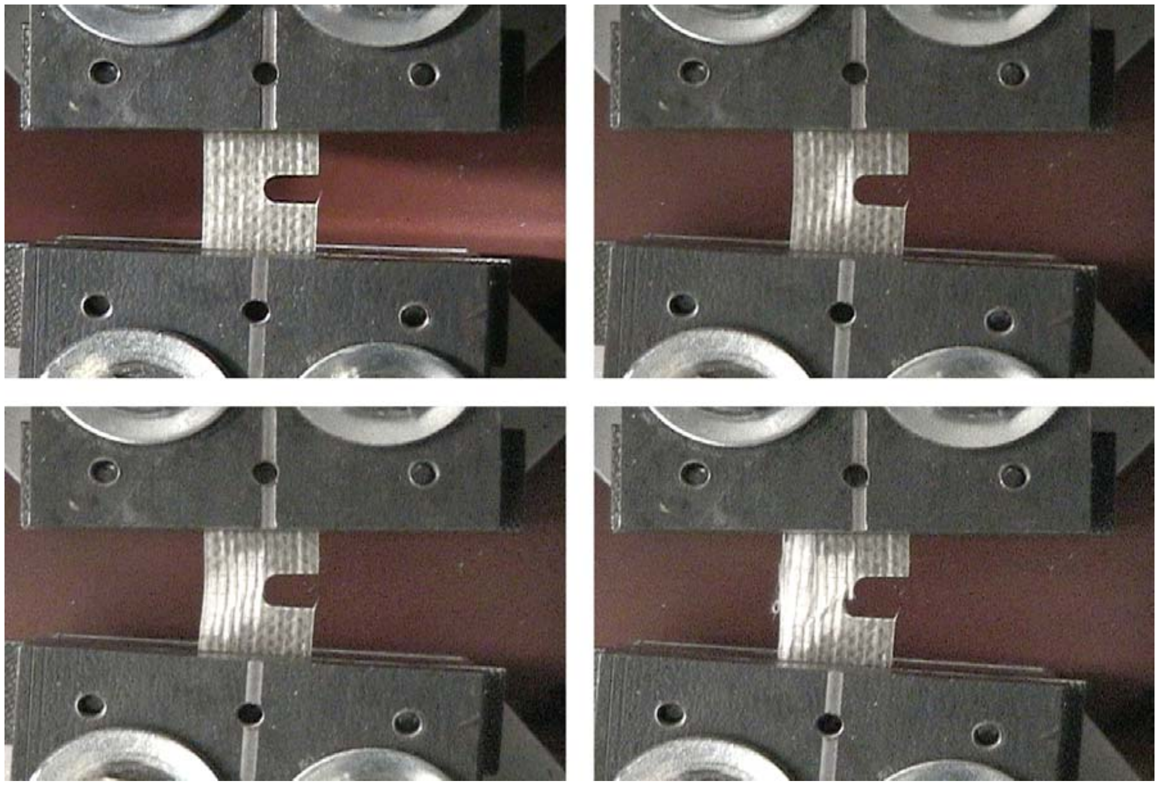


Figure 54. Progressive pictures from test 6, a combination of y displacement and rotation.

IPL Experimental Data

In each test case, three samples were tested at minimum and the two tests with the most agreement were kept. At this point we refer back to figure 5. This figure is shown again for reference.

Load vs. Displacement of Composite Sample

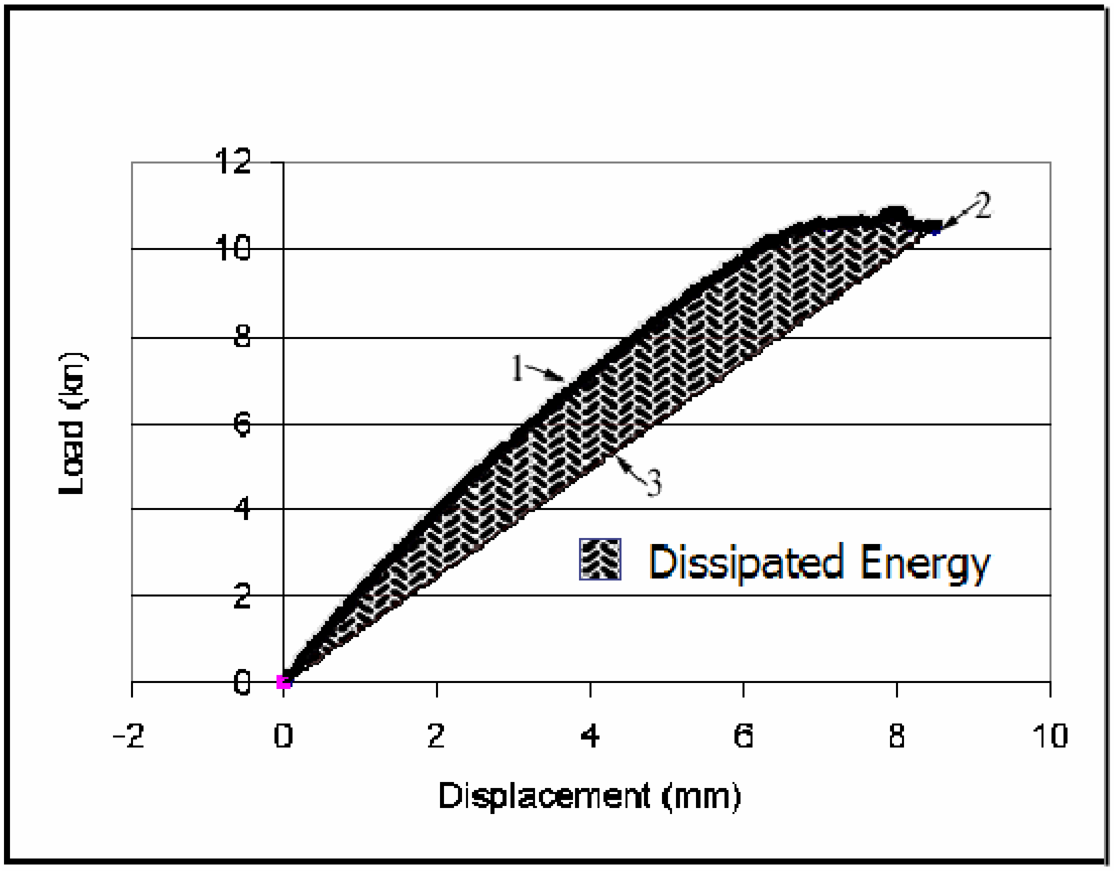
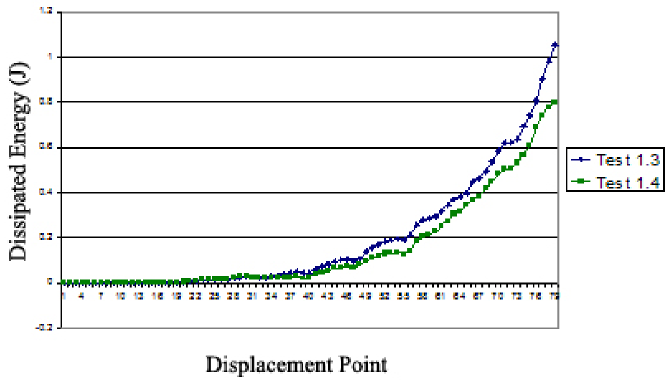


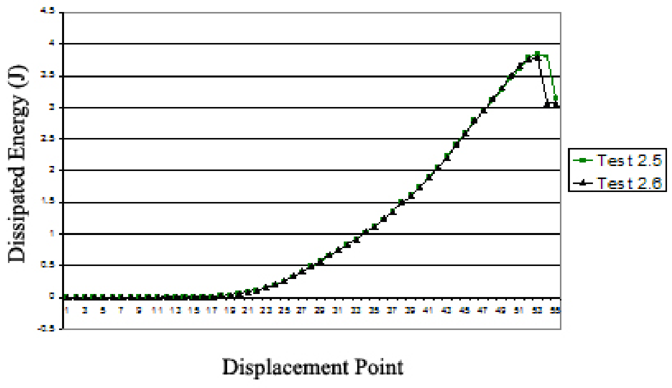
Figure 5. A generic load vs. displacement graph and its dissipated energy.

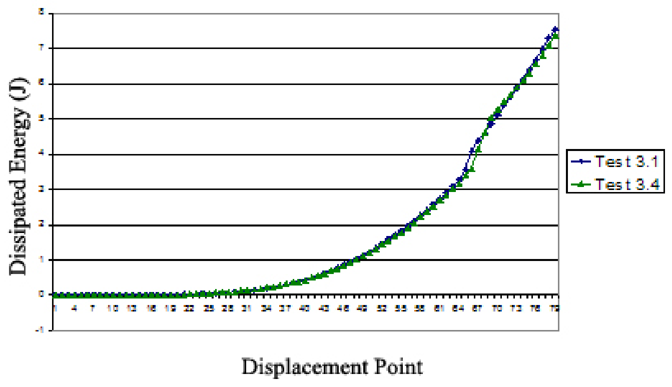
For each test with the IPL three of these graphs can be created for the three orthogonal displacements. The total dissipated energy can be found by simply adding these three dissipated energy components together.

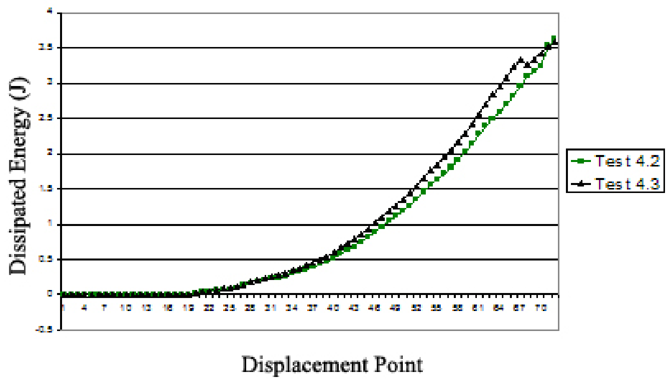
The total dissipated energies for each of the seven tests are shown in figure 55.

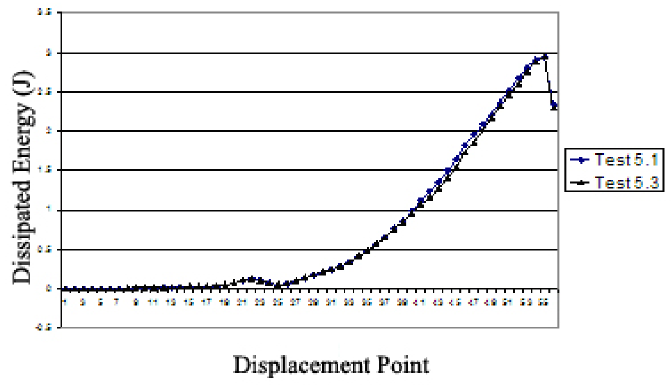
Dissipated Energies For Tests 1 Through 7

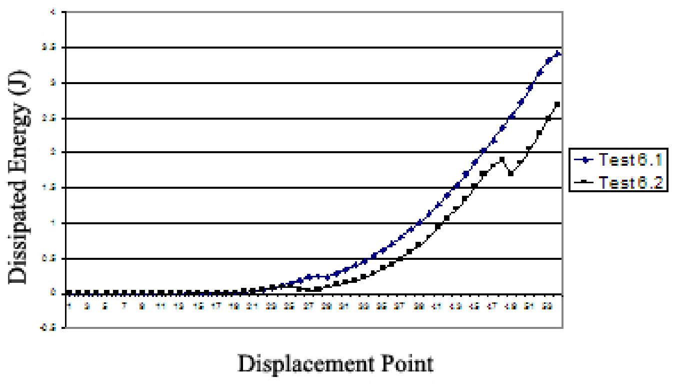





Dissipated Energy (J)
Displacement Point
Test 3.1
Test 3.4


Dissipated Energy (J)
Displacement Point
Test 4.2
Test 4.3


Dissipated Energy (J)
Displacement Point
Test 5.1
Test 5.3



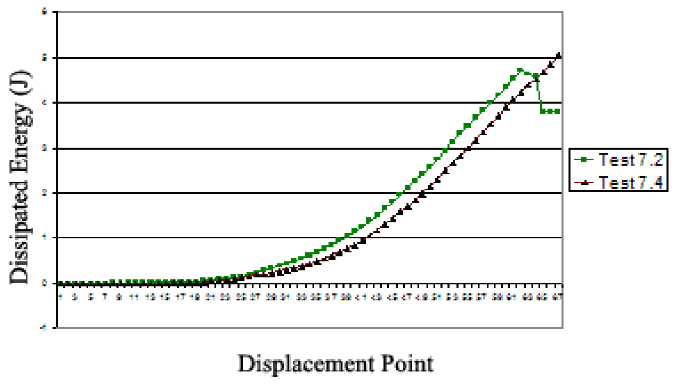


Figure 55. Total dissipated energy plots for IPL tests.

These seven tests proved to be reproducible. At this point the displacements and dissipated energies are exported from Excel. Matlab is then employed to calculate the strain fields for each displacement point, using the principle of superposition and the finite element models for each of the pure displacement modes. With the strain fields now known for each experimental displacement point the dissipated energy density function can be formulated and the coefficients found. This process is again performed in Matlab. Matlab code is attached in appendix A.

CHAPTER 10: RE-EVALUATING THE ENERGY CRITERIA

Dissipated energies were reasonably close to each other when working with the CTS samples. Dissipated energies with the IPL samples varied over a large range. These are shown in figure 56.

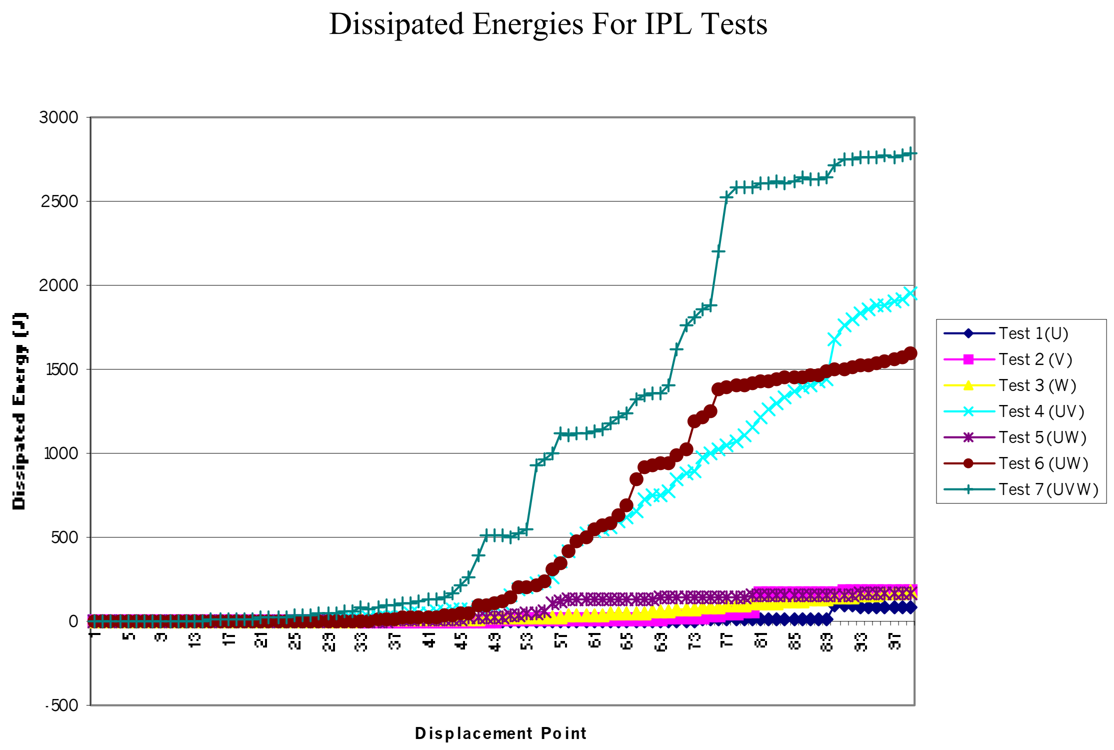


Figure 56. Dissipated energies for all IPL tests.

Test 7 dissipates thirty times the energy of test 1. As could be expected, results using the same DED approach as the CTS samples were poor over the entire load range. Due to the disparity of dissipated energies, some kind of normalization is required to allow for a more accurate model to be created. A finite element model will be able to calculate the theoretical strain energy for a given displacement, thus the theoretical (and also recoverable) strain energy is a convenient value for normalization.

On further discussion the significance of the experimental dissipated energy normalized by the theoretical recoverable energy is unclear. It makes more sense to normalize the loss of recoverable energy by the theoretical recoverable energy. This method will hereafter be referred to as NDRE or the normalized difference in recoverable energy. NDRE is discussed in the following chapter.

CHAPTER 11: DERIVATION OF NDRE

In this chapter a new metric, normalized difference in recoverable energy, is discussed. An arbitrary test is shown in figure 57.

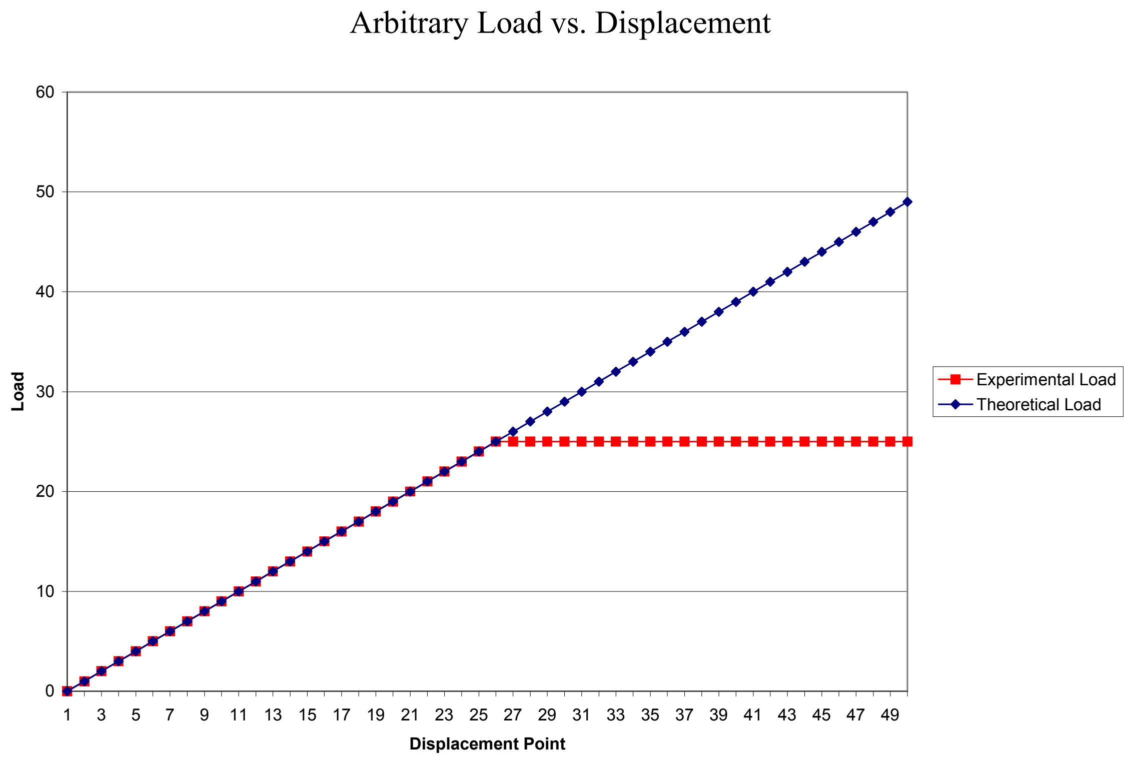


Figure 57. An arbitrary test.

In figure 57 it can be seen that up to point 25 the theoretical and experimental loads are the same. After this point there is a departure in the values of the loads. The recoverable energies are shown for point 30 in figure 58.

Recoverable Energies For Point 30

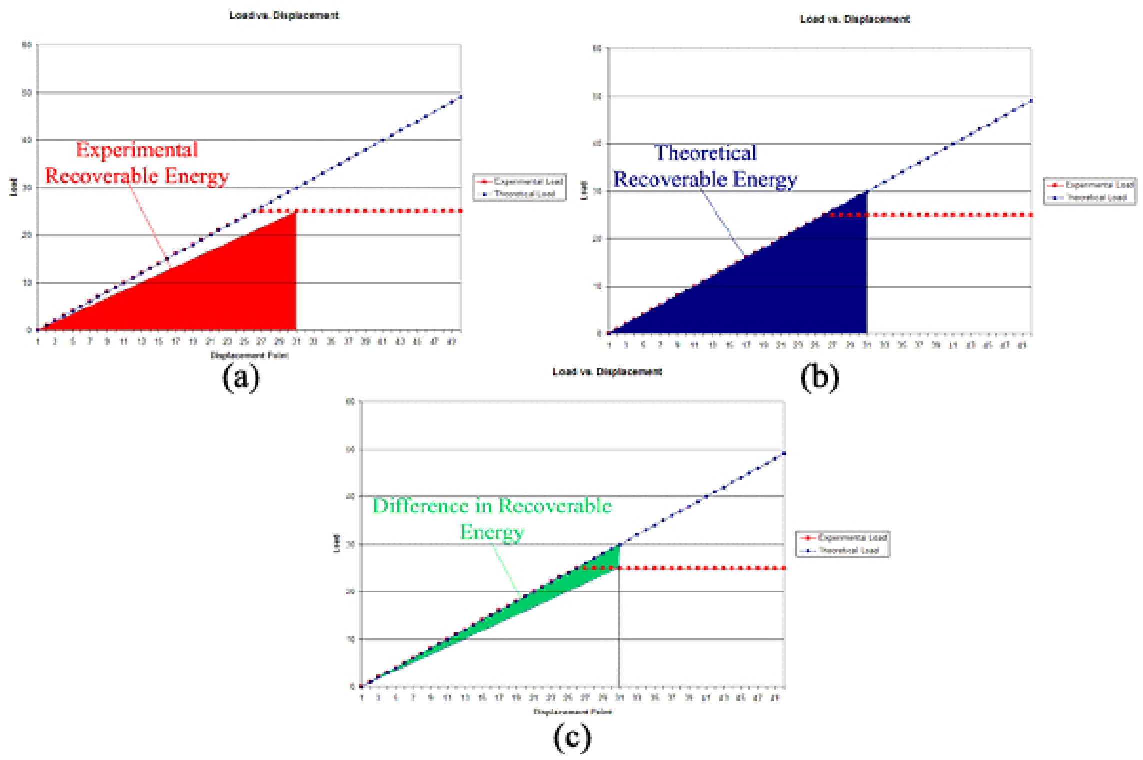


Figure 58. Recoverable energies for point 30.

In figure 58 the red area represents the recoverable energy after damage has developed; the same as line 3 in figure 4. The blue area in figure B is the theoretical recoverable energy, based on a linear-elastic undamaged sample. The green area in figure C is the difference between these recoverable energies. The energies for points 35 and 40 are shown in figures 59 and 60.

Recoverable Energies For Point 35

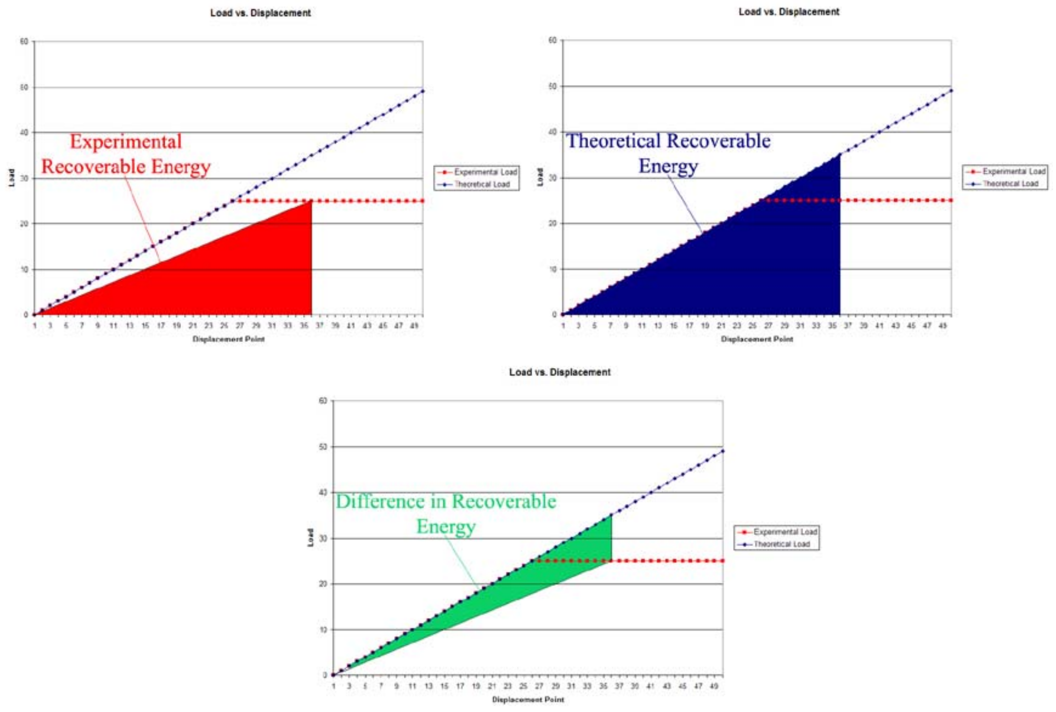


Figure 59. Energies for point 35.

Further extension results in a new set of curves. This is shown for point 40 in figure 60.

Recoverable Energies For Point 40

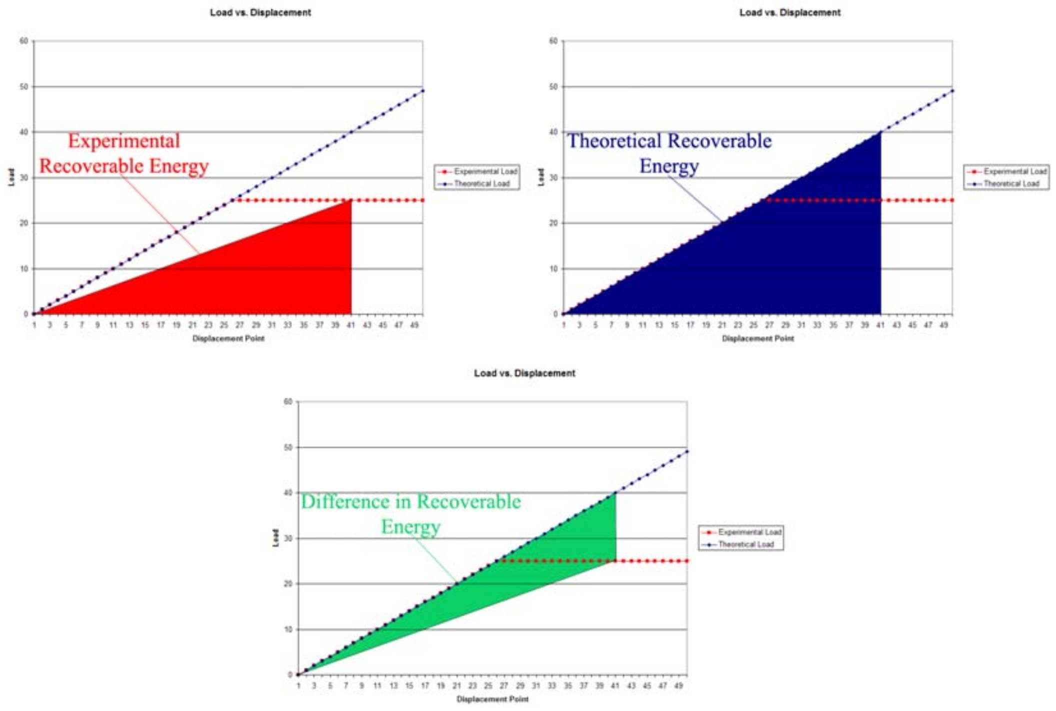


Figure 60. Energies for point 40.

A plot of the energies for all data points are shown in the following figure. It should be noted that the dissipated energy function can easily be recreated from the above energy curves.

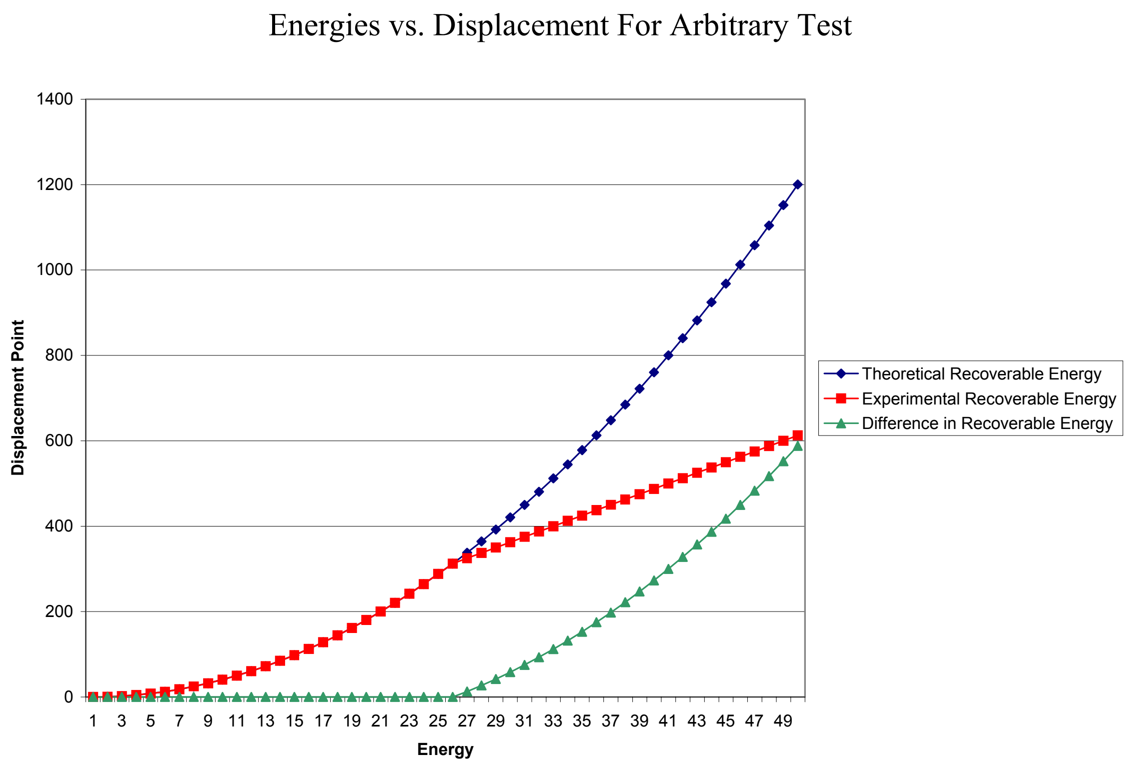


Figure 61. Energies for the arbitrary test.

Dividing the difference in recoverable energy by the theoretical recoverable energy allows for a convenient measure of the health of the system based on elastic analyses. This metric varies from 0 to 1 and measures how much of the available energy in a system has been absorbed by the damage process. This normalized difference in recoverable energy is shown in figure 62.

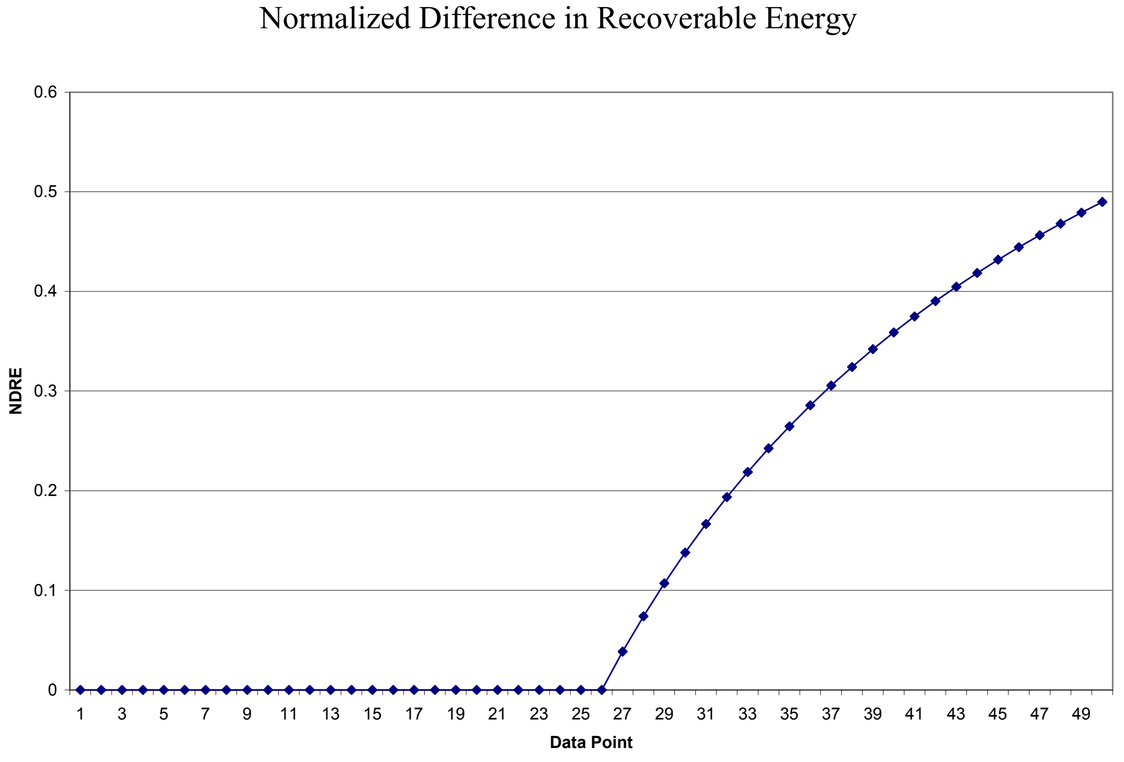


Figure 62. Normalized difference in recoverable energy.

At the end of the test above the designer can say “damage to the specimen has absorbed 50% of the available energy”.

Mathematical Formulation of NDRE

A change in the problem formulation must be performed as the normalized difference in recoverable energy or NDRE is independent of the volume of the test coupon. The total dissipated energy in a finite element model was formulated by summing over the number of plies and elements in the model the product of the dissipated energy density function and the volume of the element. This was stated in equation 3.

$$\Phi = \Sigma(p)\ \Sigma(e)\ \ \phi(\varepsilon)*a(e)*t(p)$$

where a(e) is the area of the element, t(p) is the thickness of the element and $\phi(\varepsilon)$ is the dissipated energy density function. For NDRE the formulation is as follows in equation 6.

$$\Omega = \Sigma(p)\ \Sigma(e)\ \ (\psi(\varepsilon)*\ a(e)*t(p))\ /\ v$$

where Ω is the measured NDRE, $\psi(\varepsilon)$ is the difference in recoverable energy fraction and v is the volume of the coupon.

(6)

The function $\psi(\varepsilon)$ states "A differential volume of material subjected to the strain state ε will absorb a fraction $\psi(\varepsilon)$ of the available energy through damage". A procedure and system of equations similar to that used in the DED approach can be used to determine the nodal values of the NDRE function (equations 8 through 11).

Applying the NDRE criteria to the experimental data from the IPL is shown in figure 63. This data has much less test to test variance than the dissipated energy shown in figure 56.

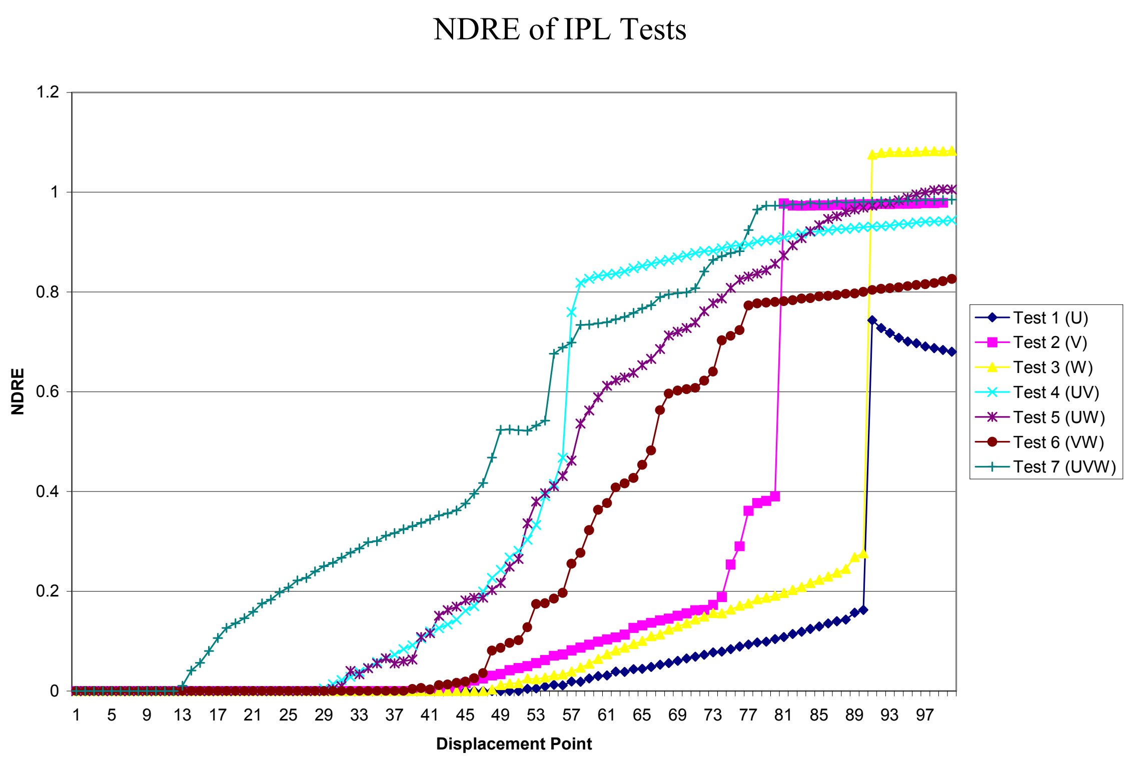


Figure 63. NDRE for each of the IPL tests.

Modeling the NDRE function ψ(ε) requires less mathematical flexibility than modeling the dissipated energy density function ϕ(ε).

NDRE RESULTS

Results from fitting the function ψ(ε) to the seven IPL tests are shown in figure 64. Experimental data is shown in green while the NDRE function is shown in blue.

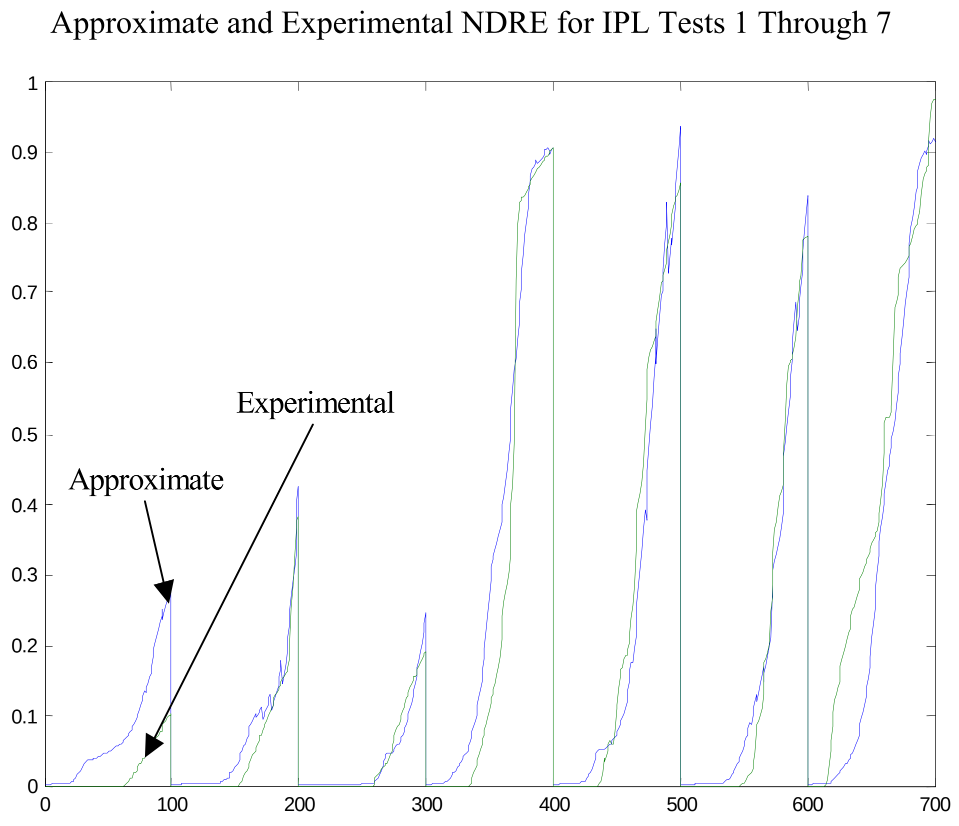


Figure 64. Experimental and theoretical NDRE.

These results have a varying degree of success. Surely, the derived function gives a qualitative impression of the health of a system, and in better cases, a quantitative impression as well. These results are much better than any results using the DED approach with the IPL data.

Examination of Test 3 (ω)

Results from test 3 are the most accurate. This test shown by itself in figure 65.

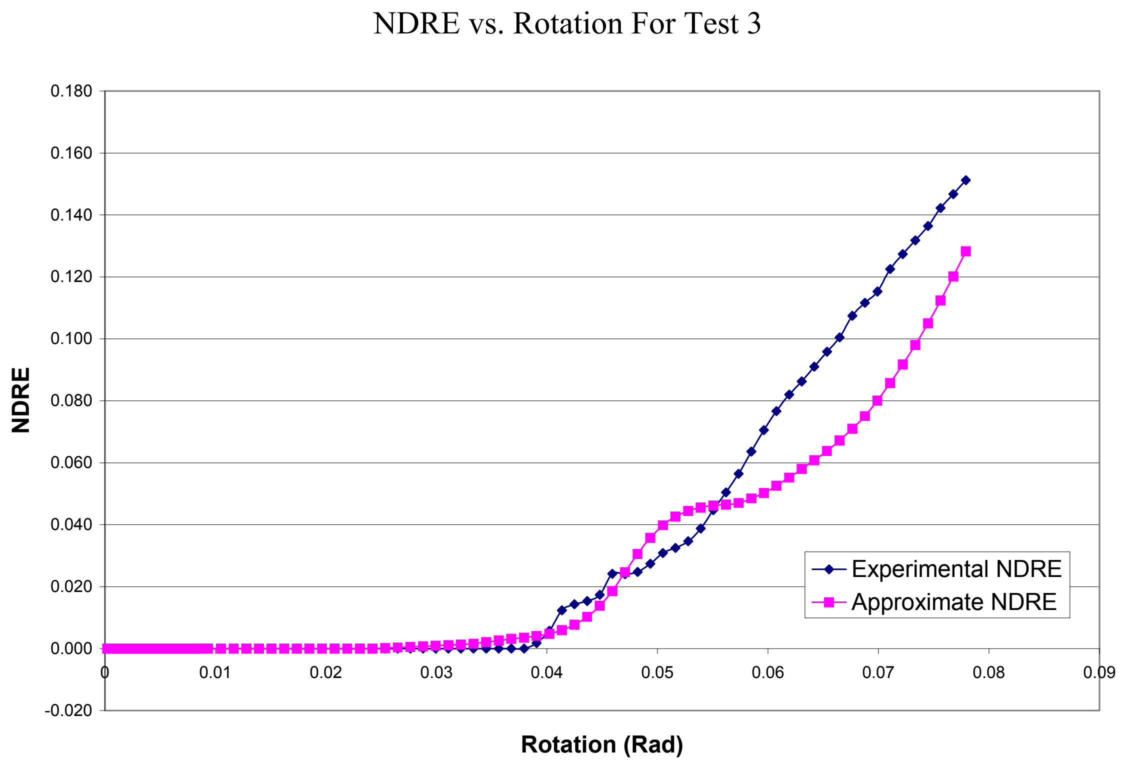


Figure 65. The experimental and approximate NDRE for test 3.

Tabular data of the experimental and approximate NDRE is shown in table 4.

NDRE Data For Test 3

Rotation (Rad)	Approximate NDRE	Experimental NDRE
0.000	0	0
~	~	~
0.025	0.000	0.000
0.027	0.001	0.000
0.028	0.001	0.000
0.029	0.001	0.000
0.030	0.001	0.000
0.031	0.001	0.000
0.032	0.002	0.000
0.033	0.002	0.000
0.035	0.003	0.000
0.036	0.003	0.000
0.037	0.004	0.000
0.038	0.004	0.002
0.039	0.005	0.006
0.040	0.006	0.012
0.041	0.008	0.014

Table 4. Tabular NDRE data for test 3.

There are several ways to determine when the approximation indicates that initial failure occurs. The approximate NDRE becomes nonzero at a rotation of .027 radians. The experimental NDRE becomes nonzero at a displacement at .038 radians. This corresponds to a prediction error of 29%, this error is a conservative one. Each approximation in figure 64 predicts the onset of damage conservatively. Mathematically speaking, the approximate NDRE is obliged to be a smooth function whereas the experimental NDRE curves often have a “sharp” start. With further experience working with this criterion the analyst may decide to ignore approximate NDRE values below some negligible value. For example, the analyst could chose to ignore approximate NDRE values below .5%. In this case the approximate NDRE would predict the onset of damage at .039 radians, where it occurs experimentally.

The moment vs. rotation relationship for test 3 is shown in figure 66.

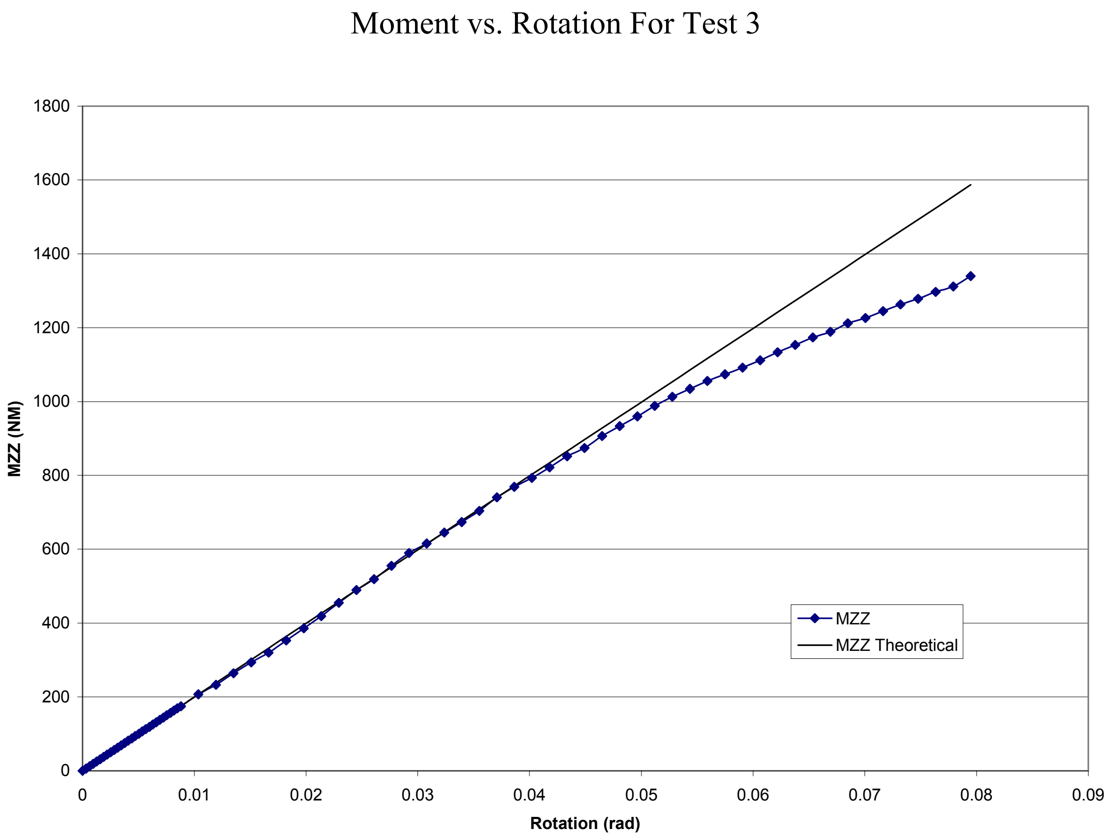


Figure 66. Moment vs. Rotation for test 3. Here the experimental data is shown in blue and the theoretical (if no damage were to occur) moment is shown in black.

The theoretical moment shown in figure 66 was determined from the stiffness of the finite element model. The experimental and approximate recoverable energies are shown in figure 67. The theoretical recoverable energy (TRE) was calculated using the stiffness from the finite element model as well. This is an important point. The analyst needs only the stiffness from the FEA model and the NDRE curve for each of these analyses. The approximate recoverable energy (ARE) is found with equation 7.

$$ARE(i) = (1 - NDRE(i))*TRE(i)$$

where ARE is the approximate recoverable energy, NDRE is the normalized difference in recoverable energy and TRE is the theoretical recoverable energy.

(7)

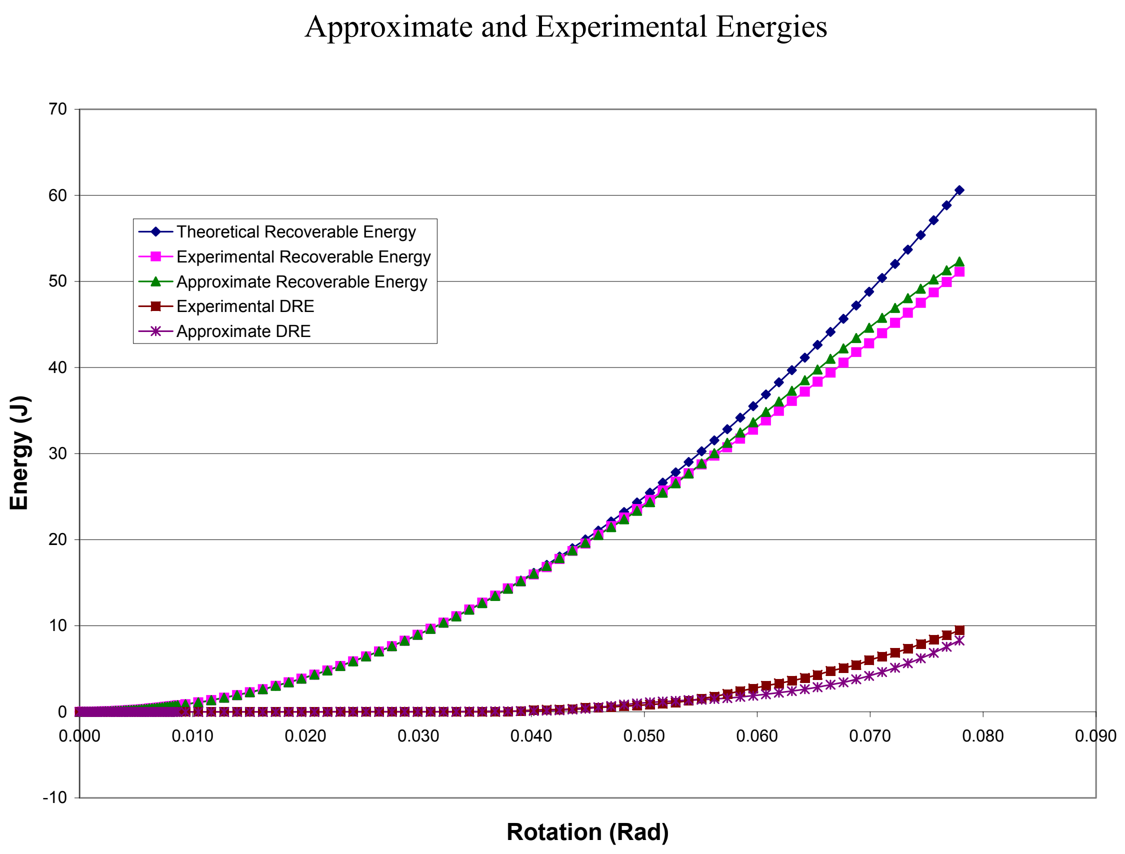


Figure 67. Approximate and experimental energies for test 3.

The approximate moment (AM) vs. rotation relationship can be created from the approximate recoverable energy with equation 8.

$$AM(i) = 2*ARE(i)/R(i)$$

where AM is the approximate moment, ARE is the approximate recoverable energy, and R is the rotation.

(8)

The moment vs. rotation relationships are shown in figure 68.

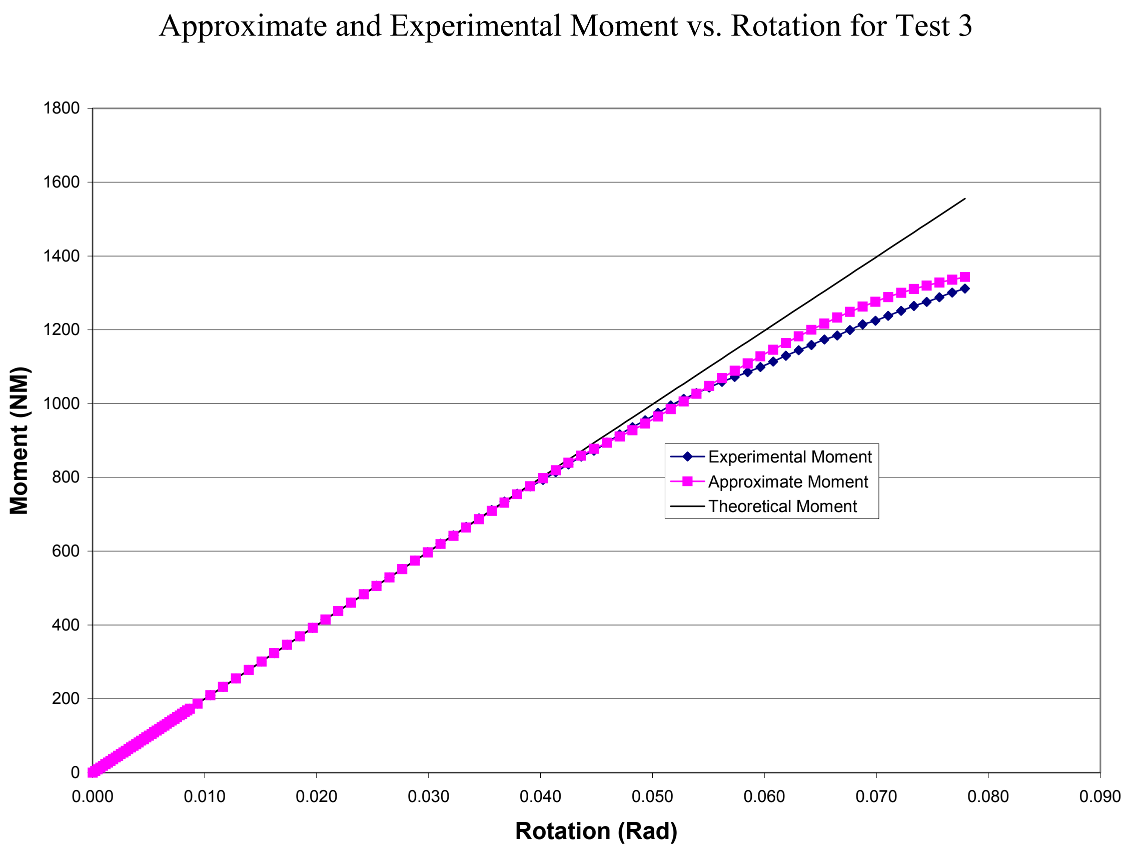


Figure 68. Approximate and experimental moment vs. rotation for test 3.

These results are good. The moment can be accurately predicted while damage is occurring using only the NDRE function and the stiffness from the finite element model. The calculations to perform there analyses are also very simple.

Examination of Test 4 (u + v)

A separate NDRE function was calculated for tests one through four. This increases the accuracy of the approximation for each of these tests. Results for test four are shown in figure 69.

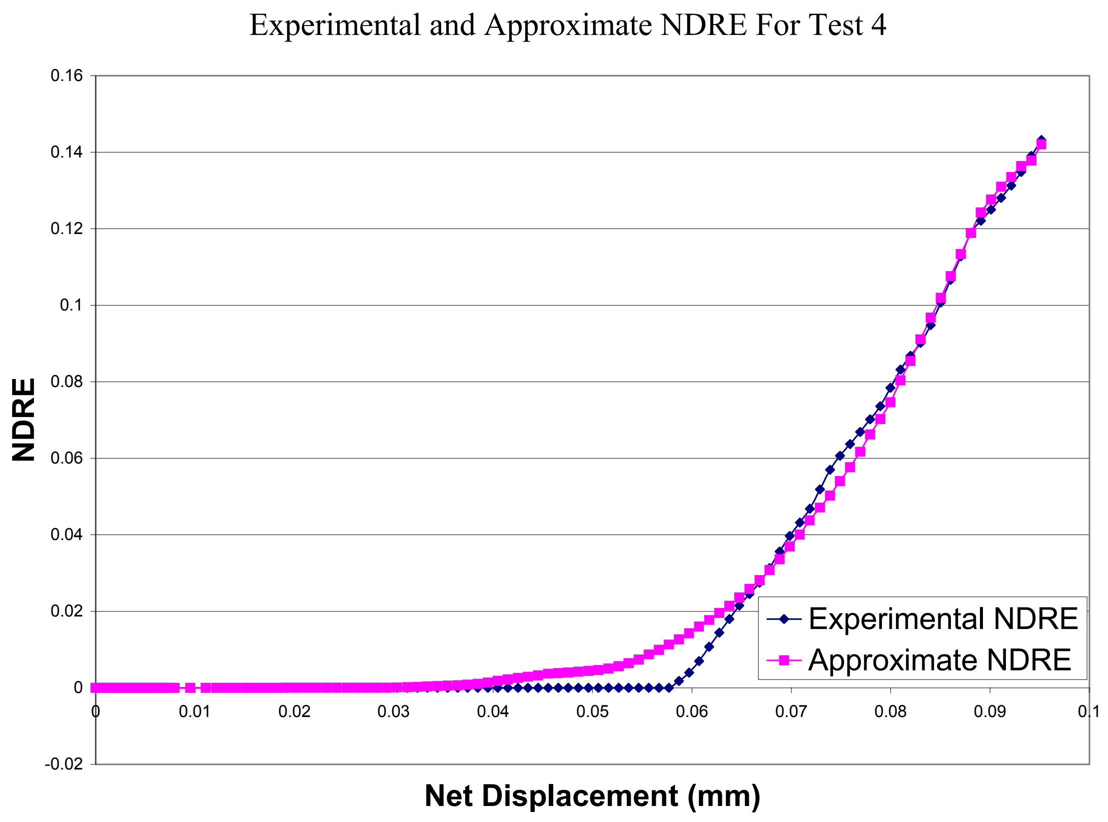


Figure 69. Experimental and Approximate NDRE for test 4.

Test 4 applies a displacement that is a combination of x and y displacements. The net displacement in figure 70 is the vector sum of the applied displacements u and v. Tabular data from this test are shown in table 6.

NDRE Data For Test 4

Net Displacement	Approximate NDRE	Experimental NDRE
0	0	0
~	~	~
0.040471396	0.0018293	0
~	~	~
0.034396426	0.00044897	0
0.03540809	0.00056279	0
0.036421151	0.00065853	0
~	~	~
0.057687219	0.0113	0
0.058699325	0.012687	0.0017914
~	~	~
0.095164312	0.14205	0.14319

Table 5. Tabular data for test 4.

If levels of NDRE below .5% are ignored as in the previous examination of test 3, the approximation predicts the onset of damage at a displacement of .035 inches, while experimentally it occurs at a displacement of .059 inches. The approximate NDRE curve is much smoother than the experimental NDRE curve. Several methods to increase the sharpness of the approximate NDRE function are proposed in the following sections.

The approximate and experimental energies are shown in figure 70.

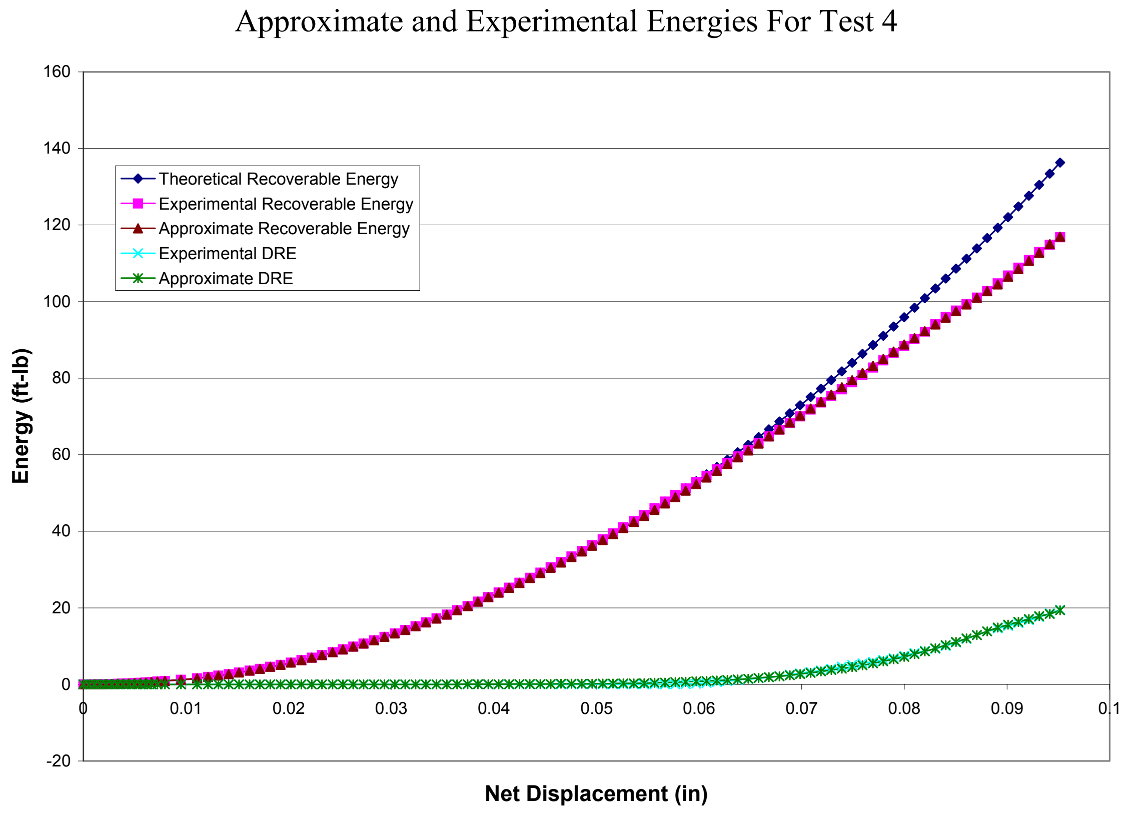


Figure 70. Approximate and experimental energies for test 4.

These results are good. The approximate and experimental energy curves are almost indiscernible. Equation 8 can now be used with the approximate recoverable energy and the net displacement to calculate the approximate net force. The experimental and approximate net forces are shown in figure 71.

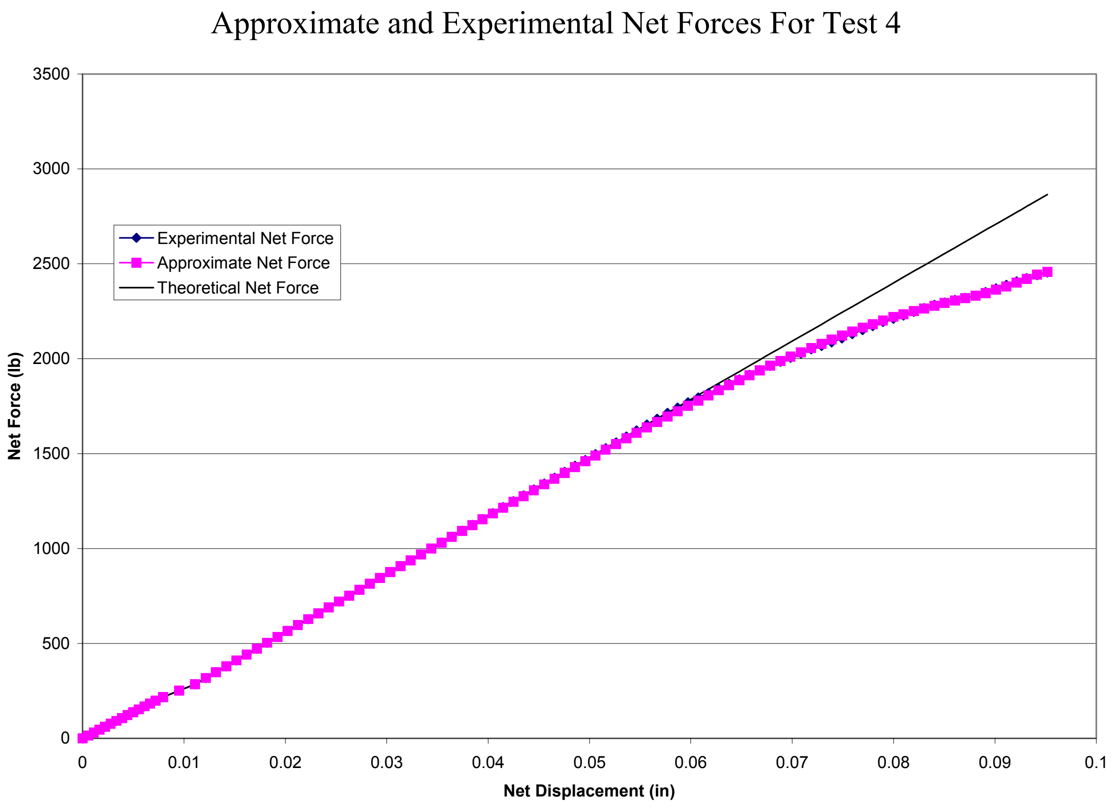


Figure 71. Approximate and Experimental net forces for test 4.

These results are very encouraging. If the NDRE can be modeled to a high level of accuracy so, in turn, can the loads. Several factors that control the quality of the energy fits are discussed in the next section.

CHAPTER 12: FACTORS THAT CONTROL FIT QUALITY

There are a number of factors that control the quality of the data fit. These factors apply to fitting data using the NDRE approach as well as the DED approach.

Number of Interpolation Nodes

The number of interpolation nodes used in the function has a large impact on the quality of the fit, the accuracy of modeling anything should increase with the number of variables. The NDRE function used in figure 64 contains 216 nodal values; six nodal divisions in the e_{11} direction, six nodal divisions in the e_{22} direction and six nodal divisions in the e_{12} direction. This NDRE function provides a more accurate solution than one with 125 nodal values. The disadvantage of increasing the number of nodes is that the computation time increases exponentially with the number of nodes. A 216 node solution requires about 1.5 hours of computation on a 2.4 GHz computer. Moving to a 7 node solution requires about 2.4 hours of computation, with a negligible increase in accuracy.

Interpolation Node Spacing

An advantage exists for solutions with an even number of interpolation node divisions. For these solutions there is an element centered on the origin in strain space. The dimensions of the solution in strain space can be adjusted so the central element spans ± 2 percent in each of the strain axes, where most fiberglass materials begin to fail. The value of NDRE can be set to zero within this element, increasing the accuracy of the solution. For solutions with odd numbers of element divisions there are four elements at the center of strain space. This means that either all the nodes of these four elements have zero valued coefficients (27 nodes, which can be a significant number of the nodes in the solution) or that the NDRE increases linearly from the origin of strain space, which is incorrect.

Another issue is the dimensions in strain space of the solution. In the real world, the material undergoing a test reaches strains that cause damage and then the material softens or comes apart, causing the strain field to increase in adjacent areas and damage new materials. Material that exceeds strains of about two percent will absorb a finite amount of energy through damage. With the linear finite element model, this damage process does not occur. Rather, the strains increase linearly to whatever the applied displacement requires. The approximation in this case is that material can continue to absorb energy at strain states that don't correspond to their physical behavior. As damage occurs the linear finite element model becomes a worse and worse approximation. It has been found in this study that a strain space of ± 10 percent has led to the highest quality fits. If the strain state of a finite element model is outside of this range it is given the value of NDRE of the node with the closest strain state. This approximates the physical phenomena that a structure can only contain a finite amount of energy.

Strain Space Uniqueness

An investigation into the uniqueness of the strain states was performed. In one experiment an artificial strain state was created using a random distribution from -10 to

10 percent for strain fields for tests one through three (the three orthogonal displacements) and then superimposed for the remaining tests. Results of fitting these artificial homogenous strain states are shown in figure 72.

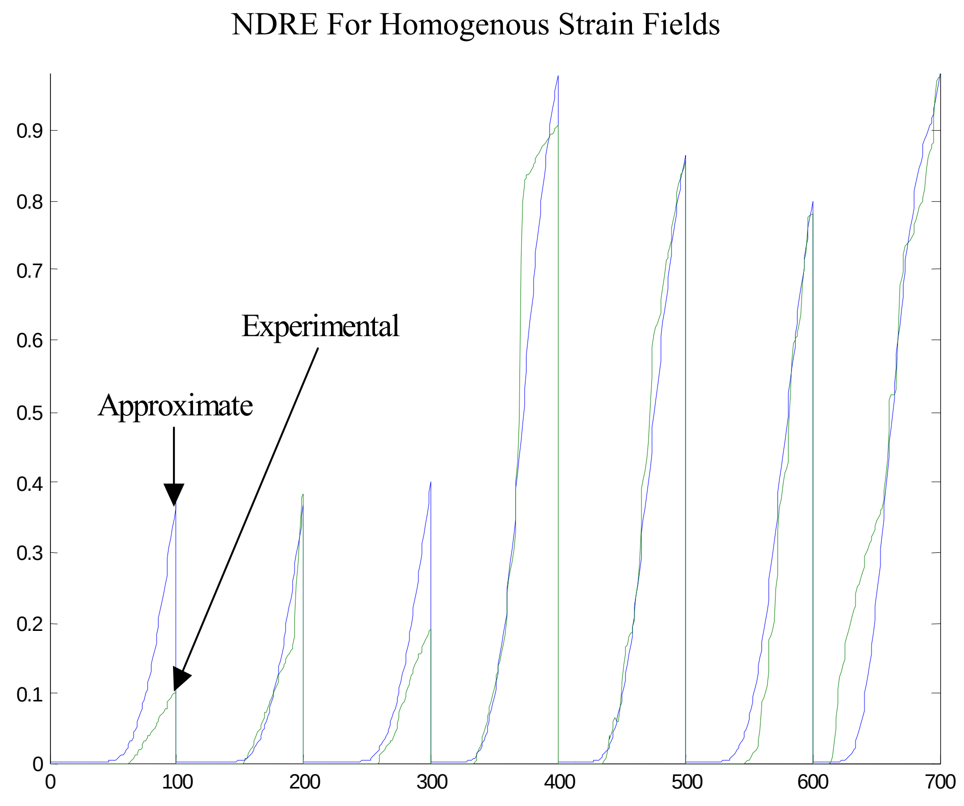


Figure 72. Results of fitting a homogenous random strain field to the experimental NDRE data.

As could be expected, these homogenous strain fields lead to somewhat homogenous results. The flexibility of the solution is decreased as these strain fields all overlap.

The experimental strain fields for the 0 degree fibers in the first the three tests are shown in figure 73.

Strain Fields For IPL Tests

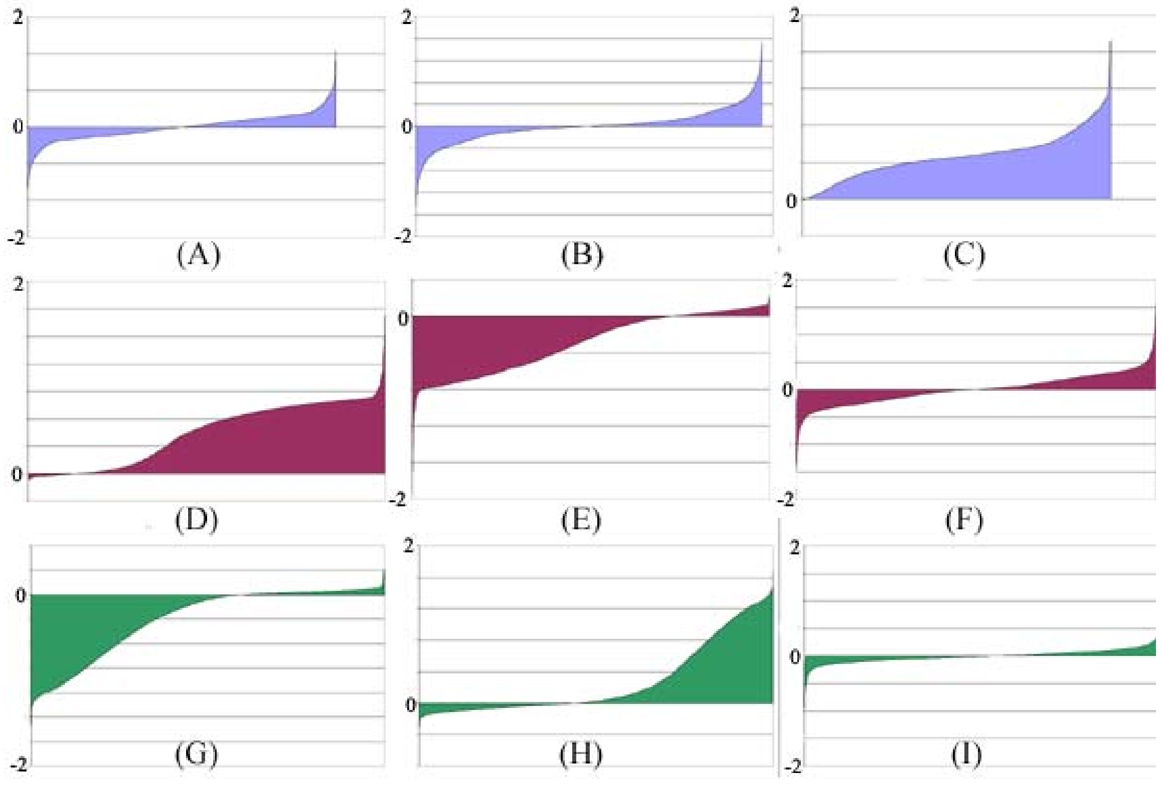


Figure 73. Strains from the three orthogonal tests:

(A) e11 for x displacement test (B) e22 for x displacement test (C) e12 for x displacement test

(D) e11 for y displacement test (E) e22 for y displacement test (F) e12 for y displacement test

(G) e11 for rotation test (H) e22 for rotation test (I) e12 for rotation test

The units in this figure are percent strain.

From examination of figure 73 it can be seen that there are portions where these strain fields overlap. Strain fields in figure 73 are for the 0 degree or longitudinal plies of the IPL sample. These IPL samples also had fibers running ± 45 degrees to these longitudinal fibers as well. The strain fields for these plies are like those in figure 73, although rotated. Including these plies in the derivation of the energy functions decreases the uniqueness of the strain fields of each test and thus may make determining the proper nodal values more difficult for the MATLAB optimization. This is certainly not to say that NDRE cannot be used to model the behavior of these samples but rather the inverse, this coupon and testing procedure may not be the best for deriving the nodal values of the NDRE function. It may be easier mathematically to determine the nodal values for an experimental approach with the following characteristics.

1) Fibers running in only one direction.

2) A different geometry where the strain fields are more uniform, such as a "dogbone" shape commonly used in material testing [1].

3) A set of displacement tests that lead to a uniform strain fields: i.e. a test for positive e_{11}, a test for negative e_{11}, a test with positive e_{11} and positive e_{22} etc.

4) A high number of these tests to explore the whole strain space.

This approach may also make it easier to address the issues outlined in the next section. With highly individualized tests as listed above an energy function can be defined that is smooth is some regions and a step function in others.

Fiber failure vs. Resin failure

One problem with the energy methods is that they do not necessarily describe fiber failures well. A fiber load vs. displacement relationship can be idealized in the figure 74.

Load vs. Displacement For Fiber Failure

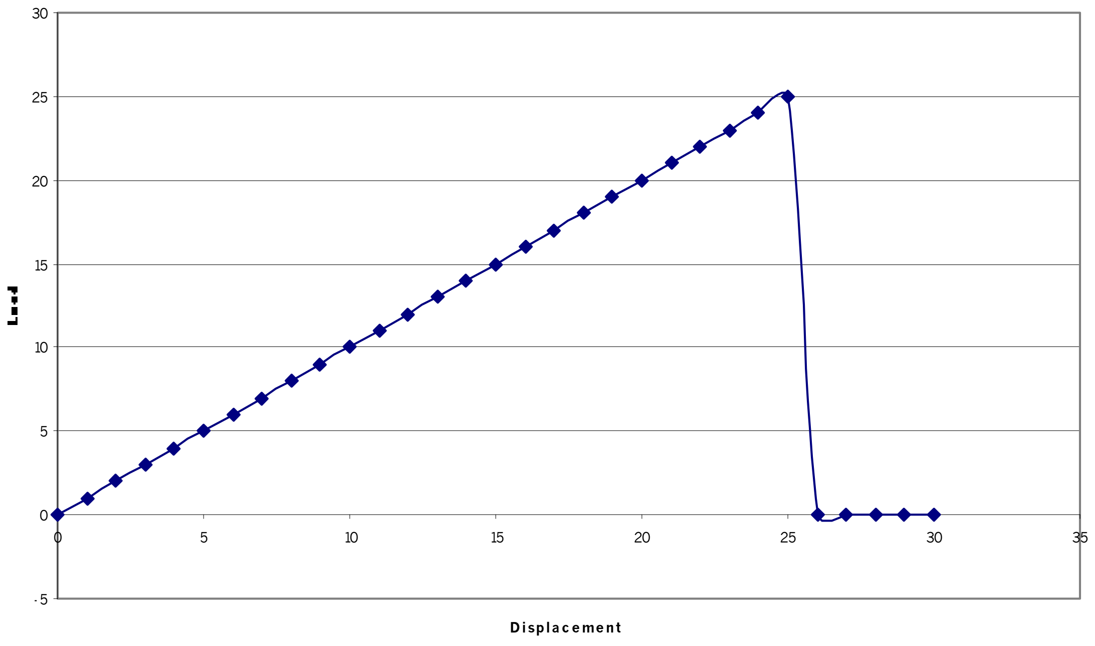


Figure 74. An idealized load vs. displacement graph for a fiber.

Stiff, high performance fibers are typically brittle materials. They load elastically until some critical load, when they fail catastrophically. Theoretical and experimental energies for this fiber failure are shown in figure 75.

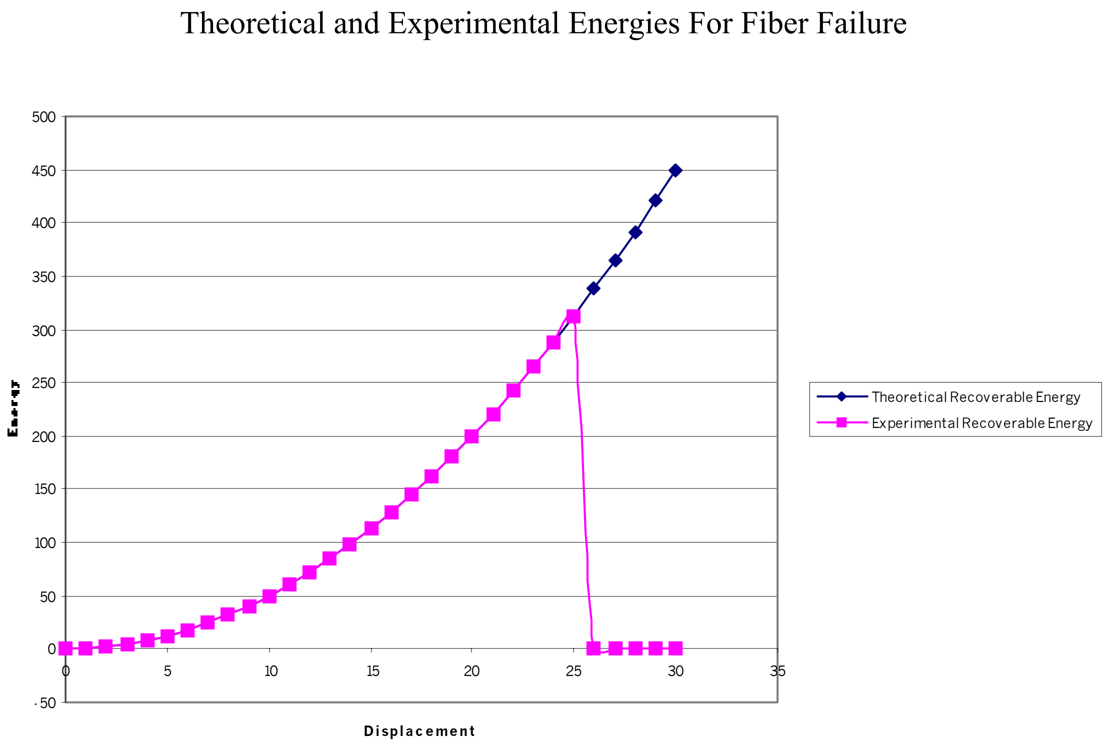


Figure 75. Theoretical and experimental recoverable energies for fiber failure.

The NDRE for the fiber failure test is shown in figure 76.

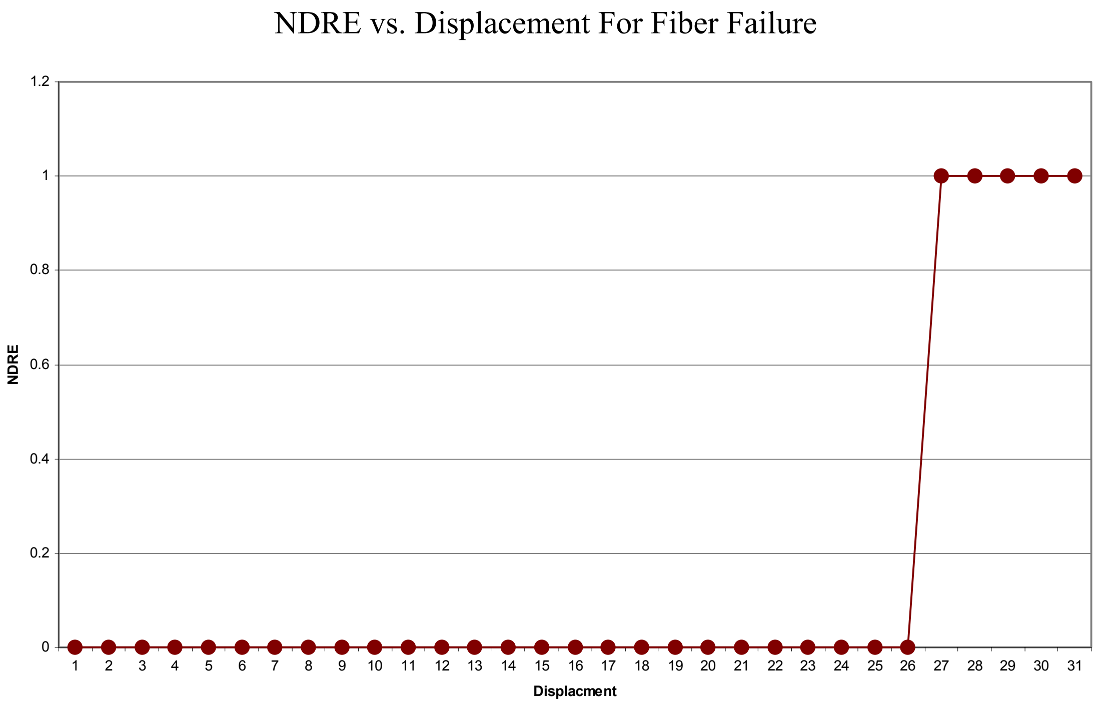


Figure 76. NDRE for fiber failure test.

In the case of a fiber breakage test the NDRE function becomes a binary function, almost like a switch indicator. It is difficult to model this phenomenon with the given interpolative approach. With the interpolative approach the function will want to transition smoothly between values of zero and one, rather than just jump between. Types of failures where damage to resin is predominantly absorbing the available energy have are more easily modeled as these are less catastrophic. A future approach could be to have a way to switch between different methods of modeling these phenomena, whether the method of damage is predominantly fibrous damage or resin damage to increase the accuracy of the approximation.

CHAPTER 13: CONCLUSIONS

Work documented in this paper has been on the theoretical development of the energy methods and their application and on the physical development of the IPL testing machine. These areas will be discussed separately below.

It has been shown that energy methods can be applied to studying the failure of composite materials. It has been shown that a dissipated energy density criteria was successful when modeling the behavior of compact tensions samples. This approach worked the best for compact tension samples that have less than 90% of the fibers in parallel to the crack path. This approach could be modified to include another method when more than 90% of the fibers are parallel to the crack path.

Dissipated energy density proved to not work as well with multi-axial displacements in the in plane loader. Another criterion, normalized difference in recoverable energy was developed and provided better results with IPL data. A normalized technique is needed because of the large differences in the amount of energy that damage absorbs with the different loading paths that the IPL can provide.

These energy techniques have demonstrated their ability to indicate the onset of damage as well as the ability to create load vs. displacement relationships for composite samples.

This paper has outlined the basic framework for an analytical approach that couples the laboratory and the design studio. Based on these results a possible design algorithm is as follows.

1) Designers are presented with a project and perform analysis to determine the required constitutive properties for a composite structure. This analysis could be done through analytical or numerical methods. The separate laminates in this design are then identified.

2) Coupons of the design lamina are created and tested in an IPL to determine an energy function. This function accurately describes how energy is absorbed through damage for any in-plane strain state.

3) The energy function is applied to a finite element model of the structure for any number of design displacements.

4) The analyst can then determine if a given design displacement a) creates a strain state that will cause damage to the structure to occur b) where this damage to the structure occurs and c) the amount of energy absorbed and the resulting load vs. displacement relationship after damage has occurred.

5) The structure can then be redesigned to avoid unwanted characteristics.

Work on the IPL began four years ago. It started as a ME 404 design project and was then completed by a number of undergraduate and graduate students. The timely completion and use of the IPL shows the high caliber of the MSU engineering students.

The IPL was designed and built entirely on the MSU campus with some of the machining performed by the CIM lab and technical services and the remaining machining performed by students in the student machine shop. The availability of the student machine shop has been integral to the completion of this project and has been extremely educational to the student engineers involved. It has shown the difference between designs that get the job done and designs that get the job done and are easy to manufacture.

The IPL was built with off the shelf components whenever possible. This has reduced the cost and time needed to complete the IPL as well as increased the reproducibility of the machine. It has also demonstrated that electric actuators perform this type of testing well while reducing the complexity and danger of the machine.

This project has demonstrated that multi-axial testing machines can be built at a cost competitive with the cost of a uni-axial testing machine, while providing much more utility. It has also been shown that data acquisition and control of this machine is a tractable problem for engineers.

CHAPTER 14: FUTURE WORK

IPL Mechanical Structure

One problem discovered with the IPL is with the problem of bearing backlash. Backlash is the amount that the bearing will displace perpendicular to the load. While a small amount of backlash with bearings is usually tolerable in mechanism testing machines require very accurate displacements to provide useful data. Load vs. Displacement data from the three actuators in an IPL test are shown in figure 77. In these figures it can be seen that loads are nearly zero up to a certain displacement where they begin to pick up. If there were no backlash in the system the loads should begin to increase linearly from zero displacement. This backlash displacement was removed mathematically in this study in order for the energies to be calculated. In an industrialized approach for developing the goals set forth in chapter one, it is convenient to use only actuator displacements for data collection. Resolving this backlash is also important for cyclical loading studies. Corrected data are shown in figure 78. Bearings with less backlash are available commercially and should be used with the IPL.

Load vs. Displacement Data For IPL Actuators

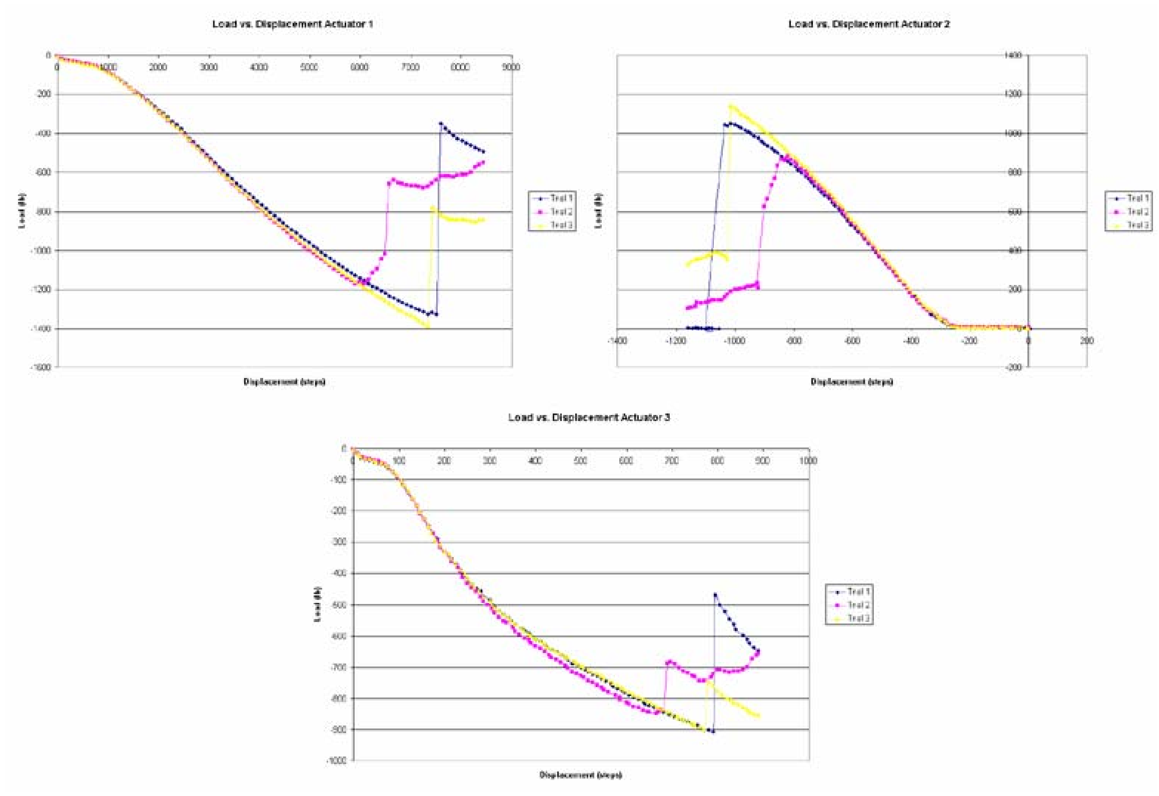


Figure 77. IPL raw actuator data.

Backlash Corrected Load vs. Displacement Data For IPL Actuators

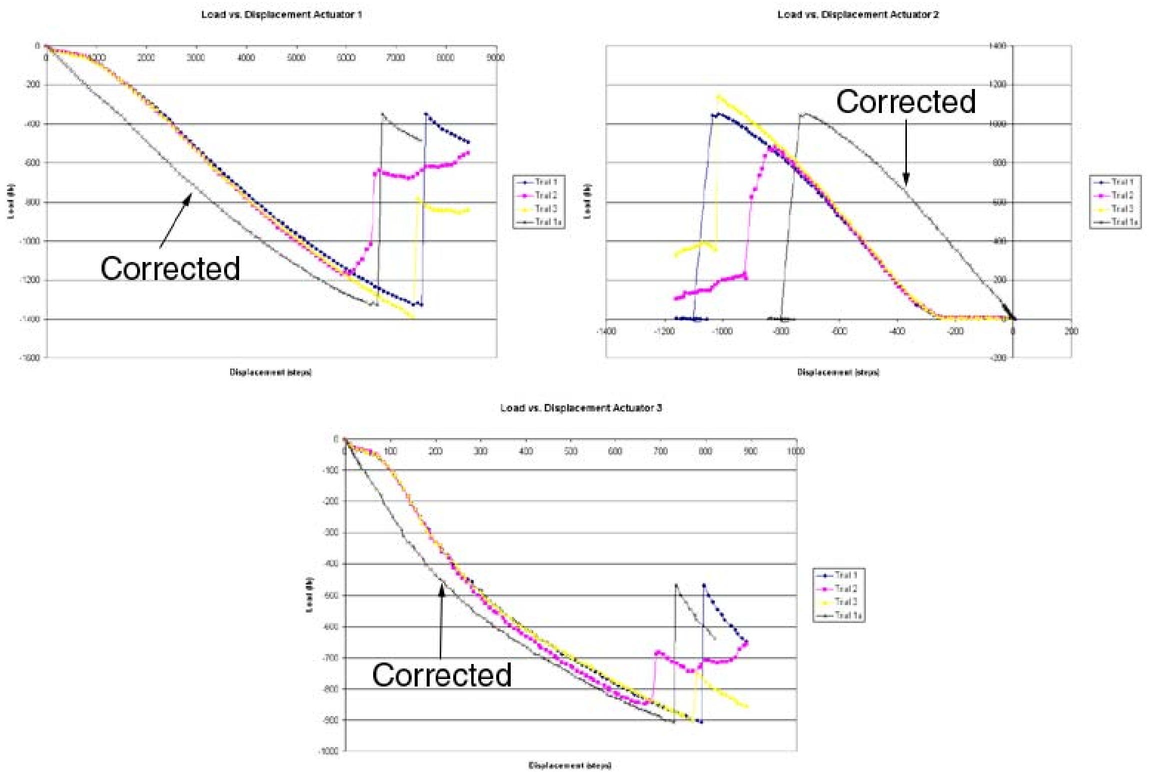


Figure 78. Actuator data with backlash removed.

Interpolation Mesh Shape

For the DED and NDRE functions in this paper an evenly spaced rectangular interpolation mesh was always used. The Tsai Wu quadratic interaction criterion discussed in chapter one demonstrates that composites typically have a failure surface that can be described as a hyper-ellipsoid. An example of this failure surface in three strain dimensions is shown in figure 79.

Tsai – Wu Failure Surface

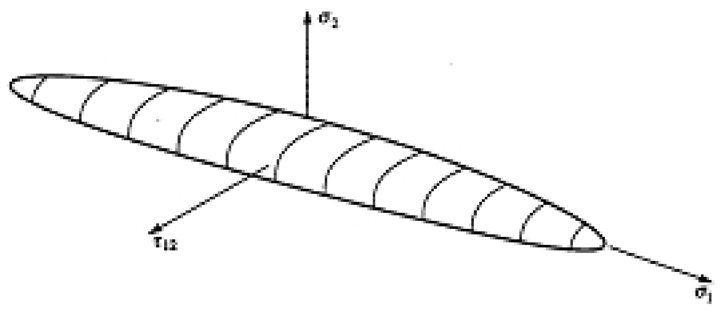


Figure 79. Tsai - Wu failure surface for composites.

One possible method for increasing the accuracy of the energy functions would be to switch to a different mesh in interpolation space. The central nodes in interpolation space could be moved to the edge of the failure ellipsoid. This would allow the function to very accurately describe the region of strain space where there will be no damage, i.e. NDRE and DED have a value of zero. The nodes outside of this surface would then be used to describe the NDRE and DED functions where their value is greater than zero.

Nonlinear Finite Element Modeling

As stated earlier, these energy methods are based on linear finite element modeling. The strain fields certainly change in a nonlinear fashion as damage is occurring within the coupons. The energy functions could be defined more accurately if the strains used to define them were also more accurate. This would also allow the energy functions to be defined over a smaller strain space, allowing for higher resolution at lower strains.

At some point with an accurate, nonlinear constitutive model, energy methods are a natural consequence. However, due to the complexity of fiber vs. resin damage this constitutive model has not been successfully developed over the entire strain space and may be path dependant. The use of these energy methods represents an approximation to this nonlinear constitutive model that is a tractable problem and requires only simple finite element analysis methods.

IPL Specimens and Loading paths

There are many issues to be addressed with the IPL specimens and loading paths. The ratio of plies to coupon thickness could be explored to ensure that the energy functions derived are macroscopic properties. In the work performed at NRL the coupons used had 20 or more plies where our samples had only six. Also NRL samples had two ply angles while ours have three. An important distinction between the NRL testing and that performed here is that the NRL samples are primarily controlled by matrix damage formation and not by fiber damage. A study is called for here to determine the effects of these parameters. Loading paths could be investigated as well to determine which paths provide the most unique strain fields for deriving the nodal values of the energy functions.

Extension to Structural Configurations

Armed with the database for a given material it is now necessary to extend this to other structural configurations to determine damage evolution and load vs. displacement response. These test cases will provide designer confidence in the application of unfamiliar material properties to primary composite structures.

CHAPTER 15: REFERENCES

[1] Cairns, D.S. and Skramstad, J. "Evaluation of Hand Lay-up and Resin Transfer Molding in Composite Wind Turbine Blade Manufacturing," Report SAND00-1425, Sandia National Laboratories, Albuquerque, NM (2000), www.coe.Montana.edu/composites

[2] Mandell, J.F., Samborsky, D.D., and Sutherland, H.J., "Effects of Materials Parameters and Design Details on the Fatigue of Composite Materials for Wind Turbine Blades," 1999 European Wind Energy Conference, 1-5 March 1999, Nice France, pp. 628-633, www.sandia.gov/Renewable_Energy/wind_energy/other/EWEC99-1.pdf

[3] Mandell, J.F., Samborsky, D.D., and Cairns, D.S. "Fatigue of Composite Materials and Substructures for Wind Turbine Blades" Contractor Report SAND 2002-0771, Sandia National Laboratories, Albuquerque, NM (March 2002), www.coe.Montana.edu/composites

[4] Samborsky, Daniel, D., "Fatigue of E-glass Fiber Reinforced Composite Materials and Structures," M.S. Thesis, Department of Civil Engineering, Montana State University, 1999.

[5] Griffith, A.A. "The phenomena of rupture and flow in solids," *Phil. Trans. Roy Soc of London*, A 221, (1921), pp. 163-197.

[6] Irwin, G.R., Fracture in "Handbuch der Physik," vol. V, Springer, New York, 1958.

[7] Hyer, M.W. “Stress analysis of fiber-reinforced composite materials” McGraw-Hill Columbus, Ohio 1998

[8] MIL HDBK 17, *The Composite Materials Handbook is the primary and authoritative source for statistically-based characterization data of current and emerging polymer matrix, metal matrix, and ceramic matrix composite materials, reflecting the best available data and technology for testing and analysis, and including data development and usage guidelines.* www.mil17.org

[9] Informal estimates from Boeing (St. Louis) and the Navy Air Systems Command (NAVAIR), Patuxent River, MD.

[10] Broek, David “Elementary Engineering Fracture Mechanics” Martinus Nijhoff Publishers, Boston 1986

[11] Smith, J.M. “Introduction to Chemical Engineering Thermodynamics” McGraw-Hill, New York 1996

[12] Mast, P.W., Nash, G.E., Michopoulos, J., Thomas, R.W, Badaliance, R., and Wolock, I., “Experimental Determination of Dissipated Energy Density as a Measure of Strain-Induced Damage in Composites,” Naval Research Laboratory report **NRD/FR/6383-92-9369**, April 17, 1992

[13] Kreyszig, Erwin “Advanced Engineering Mathematics” Wiley Text Books 1998 Hoboken, New Jersey 1998

[14] Burnett, David S. "Finite Element Analysis" Addison-Wesley Reading Massachusetts 1988.

[15] Matlab is mathematics software available from The Mathworks, 508-647-7000. www.mathworks.com

[16] Knytex fiberglass fabrics are available from the Owens Corning corporation, 800-GET-PINK www.owenscorning.com

[17] Derakane is vinylester resin available from Composites One LLC, 11917 Altamar Place Santa Fe Springs CA 90670, 800-237-0087.

[18] Ansys is finite element software available from Ansys Inc., 724-514-3304. www.ansys.com

[19] Agastra, Pancasatya, "Mixed Mode Delamination of Glass Fiber/Polymer Matrix Composite Materials," M.S. Thesis, Department of Chemical Engineering, Montana State University, 2003.

[20] Labview is measurement and automation software available from National Instruments. 512-683-8411.
www.ni.com

[21] Booth, Eric, Schaff, Mark, "IN-PLANE-LOADER, a multi axis composite testing machine," Senior Design Project Report, Department of Mechanical and Industrial Engineering, Montana State University, 2001.

[22] Rhinoceros is three dimensional CAD software available from Robert McNeel and associates. 206-545-7000.

www.rhino3d.com

[23] Pro/E Software is three dimensional CAD software available from the Parametric Technology Corporation. 888-782-3776

www.ptc.com

[24] Maple is symbolic mathematics software available from Waterloo Maple. 800-267-6583

www.maplesoft.com

[25] Nilson, James William "Electric Circuits" Addison-Wesley Publishing Massachusetts 1993

APPENDICES

APPENDIX A

MATLAB CODE

```
%The purpose of this program (readnfit) is to read in discrete data sets of an arbitrary
%number of points and create data sets with the same number of points through
%interpolation.  Thus if data set a has 44 points and data set b has 68 points this program
%can return a data sets based on a and b with 100 data points each.

cd C:\IPL
clear

%Set the on and off point for each test
on(1) = 1;
off(1) = 89;
on(2) = 1;
off(2) = 70;
on(3) =1;
off(3) = 73;
on(4) = 1;
off(4) = 44;
on(5) = 1;
off(5) = 42;
on(6) = 1;
off(6) = 52;
on(7) = 1;
off(7) = 20;

%Set the number of data points in the sets to be produced
npoints = 100;

for i=1:7;
   clear A de dx dy dr t t1 t2
   clear t3 de1 dx1 dy1 dr1 A1 x file

   %open each dissipated energy file and find the number of data points
   file = cat(2,'DER',num2str(i),'.txt');
   [dep,dxp,dyp,drp] = textread(file,'%f %f %f %f');
   de = dep(1:off(i));
   dx = dxp(1:off(i));
   dy = dyp(1:off(i));
   dr = drp(1:off(i));

   t = linspace(1,off(i),off(i));
   t1 = linspace(1,off(i),npoints);

   %Create data sets with (npoints) data points.  Int is a user defined function, it is
   %attached after this program
```

```
    dx1 = int(dx,npoints);
    dy1 = int(dy,npoints);
    dr1 = int(dr,npoints);
    de1 = int(de,npoints);

    %Create plots
    figure(1);
    plot(t,dx,t1,dx1);
    title('DX')
    figure(2)
    plot(t,dy,t1,dy1);
    title('DY')
    figure(3)
    plot(t,dr,t1,dr1);
    title('DR')
    figure(4)
    plot(t,de,t1,de1);
    title(i)
    pause

    %Write the data to a file
    out = cat(2,dx1',dy1',dr1',de1');
    size(out)
    file = cat(2,'data',num2str(i),'_',num2str(npoints),'r','.txt');
    dlmwrite(file,out,'\t');
    dirt = cat(2,'C:\IPL\test',num2str(i));
    cd(dirt)
    dlmwrite('data.txt',out,'\t');
    cd C:\IPL
end


%This function (int) interpolates a data set to return another data set with (npts) points

function bafv = int(ba,npts)

%Find the length of the source data
dimba = length(ba);

%Find the index vector for the source data
band = linspace(1,dimba,dimba);

%Find the index vector for the interpolate values
baf =  linspace(1,dimba,npts);
lowend = 0;
highend = 0;
```

```
j = 1;
low = 1;
high = 1;

for i = 1:npts
   if i == j
      low = high;
   end

   while baf(i) > j
      high = inc(high);
      j = inc(j);
   end

   while low + 1 < baf(i)
      low = inc(low);
   end

   if high == low
      bafv(i) = ba(j);
   else
      dv = (ba(high) - ba(low))/(high - low);
      bafv(i) = ba(low) + dv*(baf(i) - low);
   end
end
```

```
%This program (phi_b) creates the coefficient matrix (referred to as μ above) from the
%IPL data

clear

%Save the current directory
dir_prev = pwd;

%Define the number of data points in each data set
pts = 100;

%Set the thickness of each element. Areas in mm^2, thickness in mm, volume in mm^3
thick = 3.2/2;

%Pick which tests will have a coefficient matrix made from them
i = 1;
lamsipl(i) = 1;
i = inc(i);
lamsipl(i) = 2;
i = inc(i);
lamsipl(i) = 3;
i = inc(i);
lamsipl(i) = 4;
i = inc(i);
lamsipl(i) = 5;
i = inc(i);
lamsipl(i) = 6;
i = inc(i);
lamsipl(i) = 7;
sets = length(lamsipl);

%(expdata) is a user defined function, it is included after this program.  It's purpose is to
%load the needed data into memory.  These data include the strain fields from the FEA
%models and the experimental data.
expdata(lamsipl,sets);
load expdata.mat

%Set the range of elements used.  A fraction of the elements may be used for faster runs
%and debugging.
estart = 1;
estop = nelements;

%Define the total area and volume of the FEA model
atotal = sum(ars);
v = atotal*thick*6;
```

```
%Set the number of nodes in the solution space and the range of strains that it covers
e11max = .10;
e22max = .10;
e12max = .10;
nnum = 6;

%(elstart) is a user defined function, it is included after this program.  Its purpose is to
%define the locations of each node in solution space, determine which nodes define an
%element and the ranges in strain space covered by each element
[elinfo,ninfo] = elstart(-e11max,e11max,-e22max,e22max,-e12max,e12max,nnum);

%(nedger) is a user defined function, it is included after this program.  Its purpose is to
%determine which nodes are lying on the edge of the solution space.  This is used for
%elements whose strains are outside of the defined strain space.  They contribute to the
%coefficient of the closest edge node to them.
[nedge] = nedger(ninfo,e11max,e22max,e12max);

%Loop through each model in the set
for i = 1:sets
   sam = i;
   lam = cat(2,'test',num2str(lamsipl(i)));

   %switch directories to get the data
   d_string = cat(2,'C:\IPL\',lam)
   cd(d_string);

   %Define the strain amplification factors for superposition of element strains
   ax = alphax(startpoint(i):offpoint(i));
   ay = alphay(startpoint(i):offpoint(i));
   ar = alphar(startpoint(i):offpoint(i));
   slvD = de(startpoint(i):offpoint(i));

   %Initialize the coefficient matrix
   mat = 1;
   if mat == 1
      phi_prime = zeros(pts,nnum*nnum*nnum);
   end

   %Define the coefficient matrix for this problem
   rep = 0;
   jcnt = 0;
   for j = 1:100

   %(inc) is a user defined function, it is included after this program.  It simply adds 1 to
   %the variable it is given
      jcnt = inc(jcnt);
```

```
if jcnt == 10
      j
      jcnt = 0;
   end
   n = (sam-1)*100 + j;
   for e = estart:estop

      %Find the strains for each element through superposition
      e4511 = ax(j)*ex4511(e) + ay(j)*ey4511(e) + ar(j)*er4511(e);
      e4522 = ax(j)*ex4522(e) + ay(j)*ey4522(e) + ar(j)*er4522(e);
      e4512 = ax(j)*ex4512(e) + ay(j)*ey4512(e) + ar(j)*er4512(e);

      em4511 = ax(j)*exm4511(e) + ay(j)*eym4511(e) + ar(j)*erm4511(e);
      em4522 = ax(j)*exm4522(e) + ay(j)*eym4522(e) + ar(j)*erm4522(e);
      em4512 = ax(j)*exm4512(e) + ay(j)*eym4512(e) + ar(j)*erm4512(e);

      e9011 = ax(j)*ex9011(e) + ay(j)*ey9011(e) + ar(j)*er9011(e);
      e9022 = ax(j)*ex9022(e) + ay(j)*ey9022(e) + ar(j)*er9022(e);
      e9012 = ax(j)*ex9012(e) + ay(j)*ey9012(e) + ar(j)*er9012(e);

      %(elfind) is a user defined function, it is included after this program.  Its purpose
      %is to determine which solution element the FEA strain coordinate (exx11,
      %exx22, exx12) lies in.

      el90 = elfind(e9011,e9022,e9012,elinfo);
      el45 = elfind(e4511,e4522,e4512,elinfo);
      elm45 = elfind(em4511,em4522,em4512,elinfo);

      %(bint) is a user defined function, it is included after this program.  Its purpose is
      %to determine the values added to the nodal coefficients.
      %(bintwo) is a user defined function, it is included after this program.  Its purpose
      %is to determine the values added to the nodal coefficients for elements whose
      %strain coordinate lies outside the solution space.
      %(adds) is a user defined function, it is included after this program.  Its purpose is
      %to add the second number to the first number.

      if el45 ~= 0
         f45 = bint(elinfo(el45,:),e4511,e4522,e4512);
         for i = 1:8
            phi_prime(j,elinfo(el45,i)) =
            adds(phi_prime(j,elinfo(el45,i)),2*thick*ars(e)*f45(i)/v);
         end
      else
         c45 = bintwo(nedge,ninfo,e4511,e4522,e4512);
         phi_prime(j,c45) = adds(phi_prime(j,c45),2*thick*ars(e)/v);
```

```
            end
            if elm45 ~= 0
               fm45 = bint(elinfo(elm45,:),em4511,em4522,em4512);
               for i = 1:8
                  phi_prime(j,elinfo(elm45,i)) =
                  adds(phi_prime(j,elinfo(elm45,i)),2*thick*ars(e)*fm45(i)/v);
               end
            else
               cm45 = bintwo(nedge,ninfo,em4511,em4522,em4512);
               phi_prime(j,cm45) = adds(phi_prime(j,cm45),2*thick*ars(e)/v);
            end

            if el90 ~= 0
               f90 = bint(elinfo(el90,:),e9011,e9022,e9012);
               for k = 1:8
                  phi_prime(j,elinfo(el90,k)) =
                  adds(phi_prime(j,elinfo(el90,k)),2*thick*ars(e)*f90(k)/v);
               end
            else
               c90 = bintwo(nedge,ninfo,e9011,e9022,e9012);
               phi_prime(j,c90) = adds(phi_prime(j,c90),2*thick*ars(e)/v);
            end
         end
      end

      %Save the coefficient matrix and the experimental data
      pwd
      phi_p = phi_prime;
      save phirn18.mat phi_p
      slvDt = slvD;
      save slvDrn18.mat slvDt
   end

   %Return to the previous directory and beep to alert the user
   cd(dir_prev);
   beep
```

```
%This function (expdata) gathers the necessary data to create the coefficient matrix and
%solve for the C vector

function expdata(lams,sets)
clear de
cd C:\IPL

%Remember the current directory
pdr = pwd;

%Read the experimental data for each model
for i = 1:sets
   switch i
      case 1
         dr = cat(2,'C:\IPL\Test',num2str(lams(i)))
         cd(dr);
         [xd(:,i),yd(:,i),rt(:,i),dt(:,i)] = textread('data.txt','%f %f %f %f');
      case 2
         dr = cat(2,'C:\IPL\Test',num2str(lams(i)));
         cd(dr);
         [xd(:,i),yd(:,i),rt(:,i),dt(:,i)] = textread('data.txt','%f %f %f %f');
      case 3
         dr = cat(2,'C:\IPL\Test',num2str(lams(i)));
         cd(dr);
         [xd(:,i),yd(:,i),rt(:,i),dt(:,i)] = textread('data.txt','%f %f %f %f');
      case 4
         dr = cat(2,'C:\IPL\Test',num2str(lams(i)));
         cd(dr);
         [xd(:,i),yd(:,i),rt(:,i),dt(:,i)] = textread('data.txt','%f %f %f %f');
      case 5
         dr = cat(2,'C:\IPL\Test',num2str(lams(i)));
         cd(dr);
         [xd(:,i),yd(:,i),rt(:,i),dt(:,i)] = textread('data.txt','%f %f %f %f');
      case 6
         dr = cat(2,'C:\IPL\Test',num2str(lams(i)));
         cd(dr);
         [xd(:,i),yd(:,i),rt(:,i),dt(:,i)] = textread('data.txt','%f %f %f %f');
      case 7
         dr = cat(2,'C:\IPL\Test',num2str(lams(i)));
         cd(dr);
         [xd(:,i),yd(:,i),rt(:,i),dt(:,i)] = textread('data.txt','%f %f %f %f');
   end

   %Vertically concatenate the experimental data sets
   offpoint(i) = i*size(xd(:,i),1)
```

```
    if i == 1
        xdisp = xd(:,i);
        ydisp = yd(:,i);
        rot = rt(:,i);
        de = dt(:,i);
        startpoint(i) = 1
    else
        xdt = xdisp;
        ydt = ydisp;
        rtt = rot;
        dtt = de;
        x = xd(:,i);
        y = yd(:,i);
        r = rt(:,i);
        d = dt(:,i);
        xdisp = cat(1,xdt,x);
        ydisp = cat(1,ydt,y);
        rot = cat(1,rtt,r);
        de = cat(1,dtt,d);
        startpoint(i) = offpoint(i-1) + 1
    end
end

npoints = length(de);

%Create the amplification factors, these numbers come from the displacements used in
%the FEA model
for i = 1:npoints
    alphax(i) = xdisp(i)/.1;
    alphay(i) = ydisp(i)/.1016;
    alphar(i) = rot(i)/(.04433);
end

%Create plots of the amplification factors
plots = 1
if plots == 1
    figure(1)
    plot(alphax)
    title('alphax')
    figure(2)
    plot(alphay)
    title('alphay')
    figure(3)
    plot(alphar)
    title('alphar')
```

```
   figure(4)
   plot(dt)
   title('xd')
end

%The next section loads the strains from the FEA models
cd C:\IPL\strains

%Read in the strains from Ansys
[ex4511] = textread('ex4511.txt','%f');
[ex4522] = textread('ex4522.txt','%f');
[ex4512] = textread('ex4512.txt','%f');

[exm4511] = textread('exm4511.txt','%f');
[exm4522] = textread('exm4522.txt','%f');
[exm4512] = textread('exm4512.txt','%f');

[ex9011] = textread('ex9011.txt','%f');
[ex9022] = textread('ex9022.txt','%f');
[ex9012] = textread('ex9012.txt','%f');



[ey4511] = textread('ey4511.txt','%f');
[ey4522] = textread('ey4522.txt','%f');
[ey4512] = textread('ey4512.txt','%f');
[eym4511] = textread('eym4511.txt','%f');
[eym4522] = textread('eym4522.txt','%f');
[eym4512] = textread('eym4512.txt','%f');
[ey9011] = textread('ey9011.txt','%f');
[ey9022] = textread('ey9022.txt','%f');
[ey9012] = textread('ey9012.txt','%f');

eymass =
cat(1,ey4511,ey4522,ey4512,eym4511,eym4522,eym4512,ey9011,ey9022,ey9012);
eymax = max(eymass);
eymin = min(eymass);
aymax = max(alphay);
eymaxt = eymax*aymax;
eymint = eymin*aymax;

[er4511] = textread('er4511.txt','%f');
[er4522] = textread('er4522.txt','%f');
[er4512] = textread('er4512.txt','%f');
[erm4511] = textread('erm4511.txt','%f');
[erm4522] = textread('erm4522.txt','%f');
```

```
[erm4512] = textread('erm4512.txt','%f');
[er9011] = textread('er9011.txt','%f');
[er9022] = textread('er9022.txt','%f');
[er9012] = textread('er9012.txt','%f');

ermass =
cat(1,er4511,er4522,er4512,erm4511,erm4522,erm4512,er9011,er9022,er9012);
ermax = max(ermass);
ermin = min(ermass);
armax = max(alphar);
ermaxt = ermax*armax;
ermint = ermin*armax;

%read in the areas from Ansys
[ars] = textread('areas.txt','%f');
nelements = length(er9012);
cd(pdr)

%Save these data to a file
save expdata.mat


%This function (elstart) initializes the solution element mesh and finds all the necessary
%info about each element

function [elinfo,ninfo] = elstart(xmin,xmax,ymin,ymax,zmin,zmax,nnum)

%Define the ranges in strain space
xrange = xmax - xmin;
yrange = ymax - ymin;
zrange = zmax - zmin;

%Define the number of element divisions
elnum = nnum - 1;
n = 1;
rcnt = 1;
ccnt = 1;
lcnt = 1;

for i = 1:(elnum)*(elnum)*(elnum)
   %Number the corner nodes
   n = ((lcnt -1)*(elnum + 1)*(elnum + 1) + (ccnt - 1)*(elnum + 1) + rcnt);
   elinfo(i,1) = n;
   elinfo(i,2) = n + 1;
   elinfo(i,3) = n + nnum;
   elinfo(i,4) = n + nnum + 1;
```

```
    elinfo(i,5) = n + nnum*nnum;
    elinfo(i,6) = n + nnum*nnum + 1;
    elinfo(i,7) = n + nnum*nnum + nnum;
    elinfo(i,8) = n + nnum*nnum + nnum + 1;

    %Define the upper and lower strain bounds for each element
    elinfo(i,9) = xmin + (xrange)*(rcnt - 1)/elnum;
    elinfo(i,10) = xmin + (xrange)*(rcnt)/elnum;
    elinfo(i,11) = ymin + (yrange)*(ccnt - 1)/elnum;
    elinfo(i,12) = ymin + (yrange)*(ccnt)/elnum;
    elinfo(i,13) = zmin + (zrange)*(lcnt - 1)/elnum;
    elinfo(i,14) = zmin + (zrange)*(lcnt)/elnum;

    rcnt = inc(rcnt);
    if rcnt == elnum + 1
        rcnt = 1;
        ccnt = inc(ccnt);
        if ccnt == elnum + 1
            ccnt = 1;
            lcnt = inc(lcnt);
        end
    end
end

rcnt = 1;
ccnt = 1;
lcnt = 1;

for i = 1:(nnum)*(nnum)*(nnum)

    %Number the corner nodes
    n = ((lcnt -1)*(nnum)*(nnum) + (ccnt - 1)*(nnum) + rcnt);
    ninfo(i,1) = n;
    ninfo(i,2) = xmin + (xrange)*(rcnt - 1)/elnum;
    ninfo(i,3) = ymin + (yrange)*(ccnt - 1)/elnum;
    ninfo(i,4) = zmin + (zrange)*(lcnt - 1)/elnum;
    rcnt = inc(rcnt);

    if rcnt == nnum + 1
        rcnt = 1;
        ccnt = inc(ccnt);
        if ccnt == nnum + 1
            ccnt = 1;
            lcnt = inc(lcnt);
        end
```

```
   end
end


%This function (nedger) finds the nodes that are one the edges of the solution strain field
function [nedge] = nedger(ninfo,e11max,e22max,e12max)
nnodes = size(ninfo,1);

j = 1;
for i = 1:nnodes

   if (ninfo(i,2) == e11max)|(ninfo(i,2) == -e11max)|(ninfo(i,3) == e22max)|(ninfo(i,3)
   == -e22max)|(ninfo(i,4) == e12max)|(ninfo(i,4) == -e12max)
      nedge(j) = i;
      j = inc(j);
   end
end


%This function (inc) adds 1 to the given variable a
function x = inc(a)
x = a + 1;


%This function (elfind) determines the element containing the strain coordinate
%(e11,e22,e12)
function [el] = elfind(e11,e22,e12,elinfo)
el = 0;
nelements = size(elinfo,1);
for i = 1:nelements
   if (e11<=elinfo(i,10))&(e11>=elinfo(i,9))
   &(e22<=elinfo(i,12))&(e22>=elinfo(i,11))
   &(e12<=elinfo(i,14))&(e12>=elinfo(i,13))
      el = i;
   end
end


%This function (bint) calculates the 8 linear interpolation functions for a solution space
%element

function [f] = bint(elinfo,e11,e22,e12)
```

```
%Define the local element coordinates
x = (e11 - elinfo(9))/(elinfo(10) - elinfo(9));
y = (e22 - elinfo(11))/(elinfo(12) - elinfo(11));
z = (e12 - elinfo(13))/(elinfo(14) - elinfo(13));

%Define the interpolation functions
f(1) = (1-x)*(1-y)*(1-z);
f(2) = x*(1-y)*(1-z);
f(3) = (1-x)*y*(1-z);
f(4) = x*y*(1-z);
f(5) = (1-x)*(1-y)*z;
f(6) = x*(1-y)*z;
f(7) = (1-x)*y*z;
f(8) = x*y*z;


%This function (bintwo) finds the edge node closest to the strain coordinate
%(e11,e22,e12)
function c = bintwo(nedge,ninfo,e11,e22,e12)

nnodes = size(nedge,2);
dmin = 1E6;
for i = 1:nnodes
   d1 = e11 - ninfo(nedge(i),2);
   d2 = e22 - ninfo(nedge(i),3);
   d3 = e12 - ninfo(nedge(i),4);
   d = sqrt(d1^2 + d2^2 + d3^2);
   if d < dmin
      dmin = d;
      c = nedge(i);
   end
end
```

```
%This program (solver) combines the coefficient matrices and experimental data and
%solves for the C vector

clear

%The following section determines which data sets are included
i = 1;
%lams(i) = 1;
%i = inc(i);
%lams(i) = 2;
%i = inc(i);
%lams(i) = 4;
%i = inc(i);
%lams(i) = 6;
%i = inc(i);
%lams(i) = 7;
%i = inc(i);
%lams(i) = 8;
%i = inc(i);
%lams(i) = 9;
%i = inc(i);
%lams above 9 are for the IPL tests
%test 1
lams(i) = 10;
i = inc(i);
%test 2
lams(i) = 11;
i = inc(i);
%test 3
lams(i) = 12;
i = inc(i);
%test 4
lams(i) = 13;
i = inc(i);
%test 5
lams(i) = 14;
i = inc(i);
%test 6
lams(i) = 15;
i = inc(i);
%test7
lams(i) = 16;

%Find the number of cases to be used
sets = length(lams);
```

```
%Save the present working directory
dir_prev = pwd;

%Create the combined phi,slvD and slvcod
for i= 1:sets
   %switch directories to get the data, different files for CTS data
   lam = cat(2,'lam',num2str(lams(i)));
   if lams(i) > 9
      lam = cat(2,'test',num2str(lams(i) - 9));
   end
   d_string = cat(2,'C:\IPL\',lam)
   cd(d_string);
   load phirn14.mat
   load slvDrn14.mat

   if lams(i) < 10
      load slvcod6.mat
   end

   %Concatenate the coefficient matrices and experimental data
   if i == 1
      phi_prime = phi_p;
      slvD = slvDt;
      if lams(i) < 10
         slvcod = codt;
      end
   else
      phi_primet = cat(1,phi_prime,phi_p);
      phi_prime = phi_primet;
      slvDt = cat(1,slvD,slvDt);
      slvD = slvDt;
      if lams(i) < 10
         slvcodt = cat(1,slvcod,codt);
         slvcod = slvcodt;
      end
   end
end

%Create a directory for the solution
solname = 'solIPLrn18m7';
mkdir(solname);
cd(solname);
```

```
%Determine the number of data points and variables to solve for
[tot_points,max_combs] = size(phi_prime);

dir_prev = pwd;

%Find the columns that have a coefficient of zero and remove them from the system of
%equations.  Also remove the nodes from the central element from the system of
%equations.  This is done so the NDRE or DED for small strains has a value of zero.
j = 1;
l = 1;
for i = 1:max_combs
   s = sum(phi_prime(:,i));
   switch i
      case 87
         s = 0;
      case 88
         s = 0;
      case 93
         s = 0;
      case 94
         s = 0;
      case 123
         s = 0;
      case 124
         s = 0;
      case 129
         s = 0;
      case 130
         s = 0;
   end

   if s == 0
      zerocol(j) = i;
      j = inc(j);
   else
      col(l) = i;
      l = inc(l);
   end
end
ncol = l-1;
zerocol(j) = 0;

%Maker the initial guess at the C vector.  It can be set to values determined from a
%previous solve or to a user defined function.
mode = 0
```

```
if mode == 1
   amp = 40E13;
   e11max = .15;
   e22max = .15;
   e12max = .15;
   nnum = 6;
   [elinfo,ninfo] = elstart(-e11max,e11max,-e22max,e22max,-e12max,e12max,nnum);

   %(Cfaker) is a user defined function, it is attached after this program.  Its purpose is to
   %define initial nodal values proportional to their distance from the origin.
   Cfake = Cfaker(ninfo,amp);
else
   cd C:\IPL\test7\solIPLrn14
   load C.mat
   Cfake = C;
   %Cfake = 1000*ones(216,1);
end

%Create a coefficient matrix without columns set above

for i = 1:ncol
   if i == 1
      phi(:,1) = phi_prime(:,col(1));
      g(1) = Cfake(col(1));
      %g(1) = 1.9E3;
   else
      phi_prev = phi;
      phi_next = phi_prime(:,col(i));
      phi = [phi_prev,phi_next];
      g(i) = Cfake(col(i));
   end
end

%create the constraints on the coeffiecients
%x_l ensures that the coefficients are greater than zero
x_l = zeros(ncol,1);
gmax = g*1000000;

%Create an index for plotting
ip = linspace(1,tot_points,tot_points);
fake = phi*g';

%Create a plot of the initial guess for the C vector
figure(1)
plot(ip,fake,ip,slvD);
```

```
title('Seed');
pause

%Use Matlab's lsqlin to solve for the C vector
maxi = input('Maximum number of iterations? ');
opt = optimset('MaxIter',maxi,'TolFun',2E-40000);
%x = lsqlin(C,d,A,b,Aeq,beq,lb,ub,x0,options,p1,p2,...)
[Cs,buh,muh,fuh] = lsqlin(phi,slvD,[],[],[],[],x_l,[],g,opt);
C = Cs;
size(C);

%Create the dissipated energy approximation and plot
dap = phi*C;
figure(2)
plot(ip,dap,ip,slvD);
title('opto');

%Find the rms error between the data and the approximation
for i = 1:tot_points
   errs(i) = (sqrt((dap(i) - slvD(i))^2));
end

err = sum(errs)

%Assign the average value of the populated rows in the C vector to the
%unpopulated values
j = 1;
k = 1;
Cvg = sum(C)/size(C,1);
for i = 1:max_combs
   if zerocol(j) == i
      Ctot(i,1) = 0;
      j = inc(j);
   else
      Ctot(i,1) = C(k,1);
      k = inc(k);
   end
end

%Save the C vector
C = Ctot;
size(Ctot)

%Save the C vector and beep
save C.mat C
cd(dir_prev);
```

```
%This function (Cfaker) creates initial guess for the C vector based on the distance a
%node is from the origin.
function Cfake = Cfaker(ninfo,amp);

nnodes = size(ninfo,1)
nxpamp = 1;
nxnamp = 1;
nypamp = 1;
nynamp = 1;
nzpamp = 1;
nznamp = 1;

for i = 1:nnodes
   if ninfo(i,2) > 0
      nxp = ninfo(i,2);
      nxn = 0;
   else
      nxp = 0;
      nxn = ninfo(i,2);
   end
   if ninfo(i,3) > 0
      nyp = ninfo(i,3);
      nyn = 0;
   else
      nyp = 0;
      nyn = ninfo(i,3);
   end
   if ninfo(i,4) > 0
      nzp = ninfo(i,4);
      nzn = 0;
   else
      nzp = 0;
      nzn = ninfo(i,4);
   end

   %Calculate the distance from the origin to the node
   d = sqrt(nxpamp*nxp^2 + nxnamp*nxn^2 + nypamp*nyp^2 + nynamp*nyn^2 +
        nzpamp*nzp^2 + nznamp*nzn^2);

   if d <= .03
      Cfake(i) = 0;
   else
      Cfake(i) = amp*(d)^(12);
```

```
    end
end
```

<u>APPENDIX B</u>

ANSYS CODE

```
/COM *** Perform IPL Study ***
/COM Filename: /usr/people/wjritter/SAMPLE_FEA/coupon.sub
/COM ***                    IPL Sample                    ***
/COM This file creates the FEA model for the IPL

/COM create the coupon geometry and material model
/FILNAME,IPL60x254
filename = 'mesh_10esr'

/PREP7
n_div = 60                         ! Number of element divisions on the left edge

/COM IPL_geom is a user defined function, it is attached after this program.  The
/COMpurpose of this function is to create the mesh for the IPL sample
/INP,IPL_geom,sub

/COM create the displacements
x_disp = 2.54
y_disp = 0
rot = 0

/COM IPL_disp is a user defined function, it is attached after this program.  The
/COM purpose of this function is to impose displacements and constraints on the model
/INP,IPL_disp,sub

/COM solve the model
/SOL
allsel,all
SOLVE

!/POST1
/COM IPL_strn is a user defined function.  Its purpose is to output the strain fields of the
/COM model to be used by MATLAB

!/INP,IPL_strn,txt
save
!FINISH
!/CLEAR


/COM *** Create the coupon mesh ***
/COM ***                    IPL_geom                    ***
/COM This file creates the mesh for an IPL coupon
/COM *** Geometric Parameters of composite plate (mm) ***
```

```
length = 25.4/1000              ! Length of test section
width = 24.68/1000              ! Width of test section
depth = 11.78/1000              ! Depth of cut
d_cut = 6.35/1000               ! Diameter of cut

/COM *** Create the material and element types
/COM IPL_mat is a user defined function, it is attached after this function.  Its purpose is
/COM to define the material properties of the composite sample
/INP,IPL_mat,sub

square_frac = .7                ! Determines the size of the corner areas
x_cent = width - depth + d_cut/2! Center of curvature of notch
y_cent = length/2
hyp = (x_cent*x_cent + y_cent*y_cent)**(1/2)
yp = (y_cent*d_cut)/(2*hyp)
xp = (x_cent*d_cut)/(2*hyp)
corner = (square_frac*((x_cent**2 + y_cent**2)**(1/2) - d_cut/2))/(2**(1/2))
mid_curve = ((x_cent**2 + y_cent**2)**(1/2) - d_cut/2)/(2**(1/2))
inv_corner = (mid_curve - corner)*(2**(1/2))
x_6 = width - depth - inv_corner
rad_out = x_cent - x_6
y_12 = length/2 - d_cut/2 - inv_corner
x_11 = x_cent – xp
y_11 = y_cent - yp

! *** Create Keypoints ***
k,1,0,0,0
k,2,0,corner,0
k,3,corner,corner,0
k,4,corner,0,0
k,5,0,length - corner,0
k,6,corner,length - corner,0
k,7,0,length,0
k,8,corner,length,0
k,9,x_cent,y_cent - rad_out,0
k,10,x_cent,0,0
k,11,x_11,y_11,0
k,12,x_cent,length/2 - d_cut/2,0
k,13,x_cent - xp,y_cent + yp,0
k,14,x_cent,y_cent + rad_out,0
k,15,x_cent,y_cent + d_cut/2,0
k,16,x_cent,length,0
k,17,width,y_cent - rad_out,0
k,18,width,0,0
k,19,width,y_cent - d_cut/2,0
```

```
k,20,width,y_cent + rad_out,0
k,21,width,y_cent + d_cut/2,0
k,22,width,length,0
k,23,x_cent,Y_cent,0

! *** Create Lines ***
L,1,2                                    ! 1
L,2,3                                    ! 2
L,3,4                                    ! 3
L,4,1                                    ! 4
L,2,5                                    ! 5
L,5,6                                    ! 6
LARC,3,6,23,rad_out                      ! 7
L,5,7                                    ! 8
L,7,8                                    ! 9
L,8,6                                    ! 10
LARC,9,3,23,rad_out                      ! 11
L,9,10                                   ! 12
L,10,4                                   ! 13
L,3,11                                   ! 14
LARC,12,11,23,d_cut/2                    ! 15
L,9,12                                   ! 16
L,6,13                                   ! 17
LARC,11,13,23,d_cut/2                    ! 18
LARC,6,14,23,rad_out                     ! 19
L,14,15                                  ! 20
LARC,13,15,23,d_cut/2                    ! 21
L,8,16                                   ! 22
L,14,16                                  ! 23
L,9,17                                   ! 24
L,17,18                                  ! 25
L,10,18                                  ! 26
L,12,19                                  ! 27
L,19,17                                  ! 28
L,14,20                                  ! 29
L,20,21                                  ! 30
L,15,21                                  ! 31
L,16,22                                  ! 32
L,22,20                                  ! 33

! *** Find the length of each line, these lengths are used to ensure that lines on opposite
! sides of an area have the same number of elements ***
*DIM,long,,33
KDIST,1,2
long(1) = _RETURN
```

```
KDIST,2,3
long(2) = long(1)
long(3) = long(1)
long(4) = long(1)
long(5) = _RETURN
KDIST,5,6
long(6) = _RETURN
long(7) = long(5)
KDIST,5,7
long(8) = long(1)
long(9) = long(1)
long(10) = long(1)
KDIST,4,10
long(11) = _RETURN
long(12) = long(3)
long(13) = long(11)
KDIST,3,11
long(14) = _RETURN
long(15) = long(11)
long(16) = long(14)
long(17) = long(14)
long(18) = long(7)
long(19) = long(11)
long(20) = long(14)
long(21) = long(15)
long(22) = long(21)
long(23) = long(12)
KDIST,9,17
long(24) = _RETURN
long(25) = long(12)
long(26) = long(24)
long(27) = long(24)
long(28) = long(16)
long(29) = long(27)
long(30) = long(20)
long(31) = long(27)
long(32) = long(31)
long(33) = long(23)

! *** Create the element sizes on each line ***
*DO,i,1,33
   LESIZE,i,,,n_div*long(i)/length
*ENDDO
```

```
! *** Create areas ***
AL,1,2,3,4                                        ! 1
AL,5,6,7,2                                        ! 2
AL,6,8,9,10                                       ! 3
AL,13,3,11,12                                     ! 4
AL,11,14,15,16                                    ! 5
AL,7,17,18,14                                     ! 6
AL,17,19,20,21                                    ! 7
AL,10,22,23,19                                    ! 8
AL,26,12,24,25                                    ! 9
AL,24,16,27,28                                    ! 10
AL,31,20,29,30                                    ! 11
AL,29,23,32,33                                    ! 12
ANORM,1

! *** Mesh the areas ***
AMAP,1,1,2,3,4
AMAP,2,2,3,6,5
AMAP,3,5,6,8,7
AMAP,4,4,10,9,3
AMAP,5,3,11,12,9
AMAP,6,3,6,13,11
AMAP,7,6,14,15,13
AMAP,8,6,8,16,14
AMAP,9,10,9,17,18
AMAP,10,9,12,19,17
AMAP,11,14,20,21,15
AMAP,12,14,16,22,20

NUMMRG,NODE,,,,LOW

! *** Count the number of elements and nodes
*GET,NELEMENTS,ELEM,,COUNT       ! Finds the number of elements
*GET,NNODES,NODE,,COUNT          ! Finds the number of nodes
!ENORM,nelements                 ! Reorient all of the elements to the same direction

! *** Keep track of the node numbers for each element, and get the area
*DIM,num_nod,,NELEMENTS,4
*DIM,area,,NELEMENTS
*DO,i,1,NELEMENTS
   *GET,num_nod(i,1),ELEM,i,NODE,1
   *GET,num_nod(i,2),ELEM,i,NODE,2
   *GET,num_nod(i,3),ELEM,i,NODE,3
   *GET,num_nod(i,4),ELEM,i,NODE,4
   *GET,area(i),ELEM,i,AREA
```

```
*ENDDO
SAVE


/COM *** Create the material and elements ***
/COM ***                         IPL_mat                         ***
/COM This file creates the material and element type

/COM *** Reference Parameters ***
ET_LAYER1 = 1                    ! Reference Element type # for Beam
MP_LAYER1 = 1                     ! Reference Material type # for layer
MP_LAYER2 = 2
MP_GRIP = 3

R_LAYER1 = 1                     ! Reference # for Set of PLATE Real constants
R_LAYER2 = 2

THICKNESS1 = 3.27/6000           ! Thickness of layer 1

! *** Set the angular orientation of each ply ***
*DIM,theta,,6
theta(1) = 0 + 90
theta(2) = 45 + 90
theta(3) = -45 + 90
theta(4) = -45 + 90
theta(5) = 45 + 90
theta(6) = 0 + 90

NLAYERS = 6
!*DO,i,1,NLAYERS
!    theta(i) = 90
!*ENDDO

/COM *** Elements for layer 1 ***
ET,ET_LAYER1,SHEll91,NLAYERS,1,0,0,1,0        ! Element Type, Ref#, name of El.
KEYOPT,ET_LAYER1,8,1
type,NL,1

/COM *** Material Properties for unbroken layer ***
E_AXIAL = 10.47E9          ! Axial modulus
!E_axial = 38.68E9
E_TRANSVERSE = 38.68E9     ! Transverse modulus
!E_TRANSVERSE = 10.47E9
PR_IN_PLANE = .3           ! Inplane poisson's ratio
PR_TRANSVERSE = .3          ! Transverse poisson's ratio
GAMMA = 4.36E9             ! Shear Modulus
```

```
mp,ex,MP_LAYER1,E_AXIAL  ! Mat. Prop, Young's Mod., Mat#, Magnitude of EX
mp,ey,MP_LAYER1,E_TRANSVERSE  ! Mat. Prop, Young's Mod., Mat#, Magnitude
! of EY
mp,ez,MP_LAYER1,E_TRANSVERSE  ! Mat. Prop, Young's Mod., Mat#, Magnitude
!of EZ
mp,prxy,MP_LAYER1,PR_IN_PLANE ! Mat. Prop, Poisson's Ratio, Mat#, Mag.
mp,pryz,MP_LAYER1,PR_TRANSVERSE   ! Mat. Prop, Poisson's Ratio, Mat#, Mag.
mp,prxz,MP_LAYER1,PR_TRANSVERSE   ! Mat. Prop, Poisson's Ratio, Mat#, Mag.
mp,gxy,MP_LAYER1,GAMMA           ! Mat. Prop, Shear Modulus, Mat#, Mag.
mp,gyz,MP_LAYER1,GAMMA           ! Mat. Prop, Shear Modulus, Mat#, Mag.
mp,gxz,MP_LAYER1,GAMMA           ! Mat. Prop, Shear Modulus, Mat#, Mag.

mp,ex,MP_LAYER2,E_TRANSVERSE   ! Mat. Prop, Young's Mod., Mat#, Magnitude
! of EX
mp,ey,MP_LAYER2,E_AXIAL  ! Mat. Prop, Young's Mod., Mat#, Magnitude of EY
mp,ez,MP_LAYER2,E_TRANSVERSE  ! Mat. Prop, Young's Mod., Mat#, Magnitude
! of EZ
mp,prxy,MP_LAYER2,PR_IN_PLANE ! Mat. Prop, Poisson's Ratio, Mat#, Mag.
mp,pryz,MP_LAYER2,PR_TRANSVERSE   ! Mat. Prop, Poisson's Ratio, Mat#, Mag.
mp,prxz,MP_LAYER2,PR_TRANSVERSE   ! Mat. Prop, Poisson's Ratio, Mat#, Mag.
mp,gxy,MP_LAYER2,GAMMA           ! Mat. Prop, Shear Modulus, Mat#, Mag.
mp,gyz,MP_LAYER2,GAMMA           ! Mat. Prop, Shear Modulus, Mat#, Mag.
mp,gxz,MP_LAYER2,GAMMA           ! Mat. Prop, Shear Modulus, Mat#, Mag.

maggrp = 10
mp,ex,MP_GRIP,maggrp*E_AXIAL  ! Mat. Prop, Young's Mod., Mat#, Magnitude of
! EX
mp,ey,MP_GRIP,maggrp*E_TRANSVERSE  ! Mat. Prop, Young's Mod., Mat#,
! Magnitude of EY
mp,ez,MP_GRIP,maggrp*E_TRANSVERSE  ! Mat. Prop, Young's Mod., Mat#,
! Magnitude of EZ
mp,prxy,MP_GRIP,PR_IN_PLANE       ! Mat. Prop, Poisson's Ratio, Mat#, Mag.
mp,pryz,MP_GRIP,PR_TRANSVERSE         ! Mat. Prop, Poisson's Ratio, Mat#, Mag.
mp,prxz,MP_GRIP,PR_TRANSVERSE         ! Mat. Prop, Poisson's Ratio, Mat#, Mag.
mp,gxy,MP_GRIP,maggrp*GAMMA          ! Mat. Prop, Shear Modulus, Mat#, Mag.
mp,gyz,MP_GRIP,maggrp*GAMMA          ! Mat. Prop, Shear Modulus, Mat#, Mag.
mp,gxz,MP_GRIP,maggrp*GAMMA          ! Mat. Prop, Shear Modulus, Mat#, Mag.

! *** Input for Shell91 real constants ***
! NL,LSYM,,,,ADMSUA,,,,,,,MAT,THETA,TK(i),TK(j),TK(k),TK(l)
! NL = number of layers.  LSYM is layer symmetry = 1 (true) 0 (false).
! ADMSUA = added mass per unit area
! MAT = material reference number.  Theta = angle orientation.  Tk(i) =
! thickness of node i
```

```
r,1,NLAYERS,0,,,,
rmore,0,,,,,,,
*DO,i,1,NLAYERS
   rmore,MP_LAYER1,theta(i),THICKNESS1,,,,
     !*IF,theta(i),EQ,90,THEN
   !    rmore,MP_LAYER2,0,THICKNESS1,,,,
   !*ELSE
   !    rmore,MP_LAYER1,theta(i),THICKNESS1,,,,
   !*ENDIF
*ENDDO

MAT,LAYER1
REAL,1
TYPE,1
ESYS,0


/COM Filename: /usr/people/wjritter/SAMPLE_FEA/1_21_03
/COM ***                      IPL_disp                      ***
/COM This program creates the desired upper and lower jaw displacements
/COM *** Rotate into the IPL coordinate system
LOCAL,11,0,0,length,0,0,180,0
/COM *** Create the lower jaw displacement ***
nsel,s,loc,y,0                ! Select all the bottom nodes and fix them
d,all,UX,0
d,all,UY,0
d,all,UZ,0

/COM *** Create the upper jaw displacement ***
*IF,rot,gt,0,then
   rot1 = rot
   rot = -rot
   nsel,s,loc,y,length
   d,all,UZ,0
   x_mid = width/2
   y_mid = length/2
   nsel,s,loc,y,length
   *GET,num_nodes,node,0,count                    ! Get the number of selected nodes
   *GET,node_min,node,0,NUM,MIN! Get the smallest node number
   nod = node_min
   Pi = 3*acos(1/2)
   conv = Pi/180
   c = cos(rot*conv)
   s = sin(rot*conv)
   *DO,i,1,num_nodes
```

```
    *GET,nod_x,NODE,nod,LOC,x! Get the node's x location
    *GET,nod_y,NODE,nod,LOC,y! Get the node's y location
    r_x = nod_x - x_mid                          ! Get the node's distance vector
    r_y = nod_y - y_mid                          ! components from the midpoint
    r_x_prime = r_x*c + r_y*s! Find the rotated components
    r_y_prime = r_y*c - r_x*s
    del_x = -r_x + r_x_prime                     ! Find the displacements
    del_y = r_y - r_y_prime
    d,nod,UX,del_x + x_disp                              ! Apply the displacements
    d,nod,UY,del_y - y_disp
    *GET,next,NODE,nod,NXTH                      ! Get the next node
    nod = next
   *ENDDO
*ELSEIF,y_disp,gt,0,then
   nsel,s,loc,y,length
   d,all,UY,-y_disp
   d,all,UZ,0
   *IF,x_disp,eq,0,then
    d,all,UX,0
   *ENDIF
*ELSEIF,x_disp,gt,0,then
   nsel,s,loc,y,length
   d,all,UX,x_disp
   d,all,UZ,0
   *IF,y_disp,eq,0,then
     d,all,UY,0
   *ENDIF
*ENDIF

! *** Increase the stiffness of the elements on the grip edges to reduce their strains
!nsel,s,loc,y,length
!esln,s,0
!mpchg,MP_GRIP,all
!nsel,s,loc,y,0
!esln,s,0
!mpchg,MP_GRIP,all

! *** Remove the displacements on the corner nodes to reduce the stress ***
!NSEL,S,LOC,X,width,2*width
!DDELE,ALL,UX
!DDELE,ALL,UY
!NSEL,S,LOC,X,-width,width
!DDELE,ALL,UX
!DDELE,ALL,UY
```

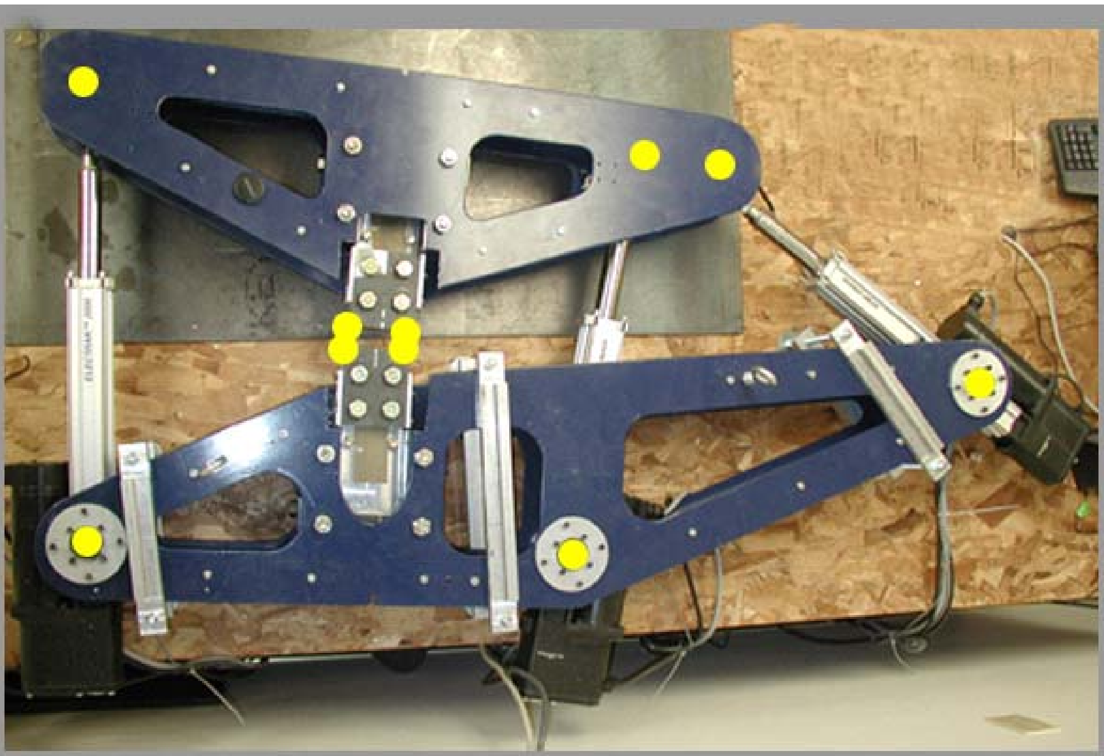
```
/COM ***                    IPL_strn                    ***
/COM This program outputs the strains at the query point.  Used in convergence study
nodx = width – depth
nody = length/2
KSEL,S,KP,,11
NSLK,S
*GET,nodefr,NODE,0,NUM,MIN
*GET,strnfrx,NODE,nodefr,EPTO,X
*GET,strnfry,NODE,nodefr,EPTO,Y
*GET,strnfrxy,NODE,nodefr,EPTO,XY
*CFOPEN,check,txt,,APPEND
*DIM,jobstat,string,128
jobstat(1) = 'Jobname: '
*VWRITE,filename
%s
*VWRITE,strnfrx
%g
*VWRITE,strnfry
%g
*VWRITE,strnfrxy
%g


/COM ***                  Strainout
/COM *** This program outputs the average strains in each element, to be used by
/COM MATLAB ***

CSYS,1
NSEL,S,EPTO,EQV,0,2E31
NSEL,A,EPTO,EQV,-2E31,0

! Go through all elements
*DIM,stx,,nelements
*DIM,sty,,nelements
*DIM,stxy,,nelements

! Find the max and min strains
stxmin = 0
stxmax = 0
stymin = 0
stymax = 0
stxymin = 0
stxymax = 0
```

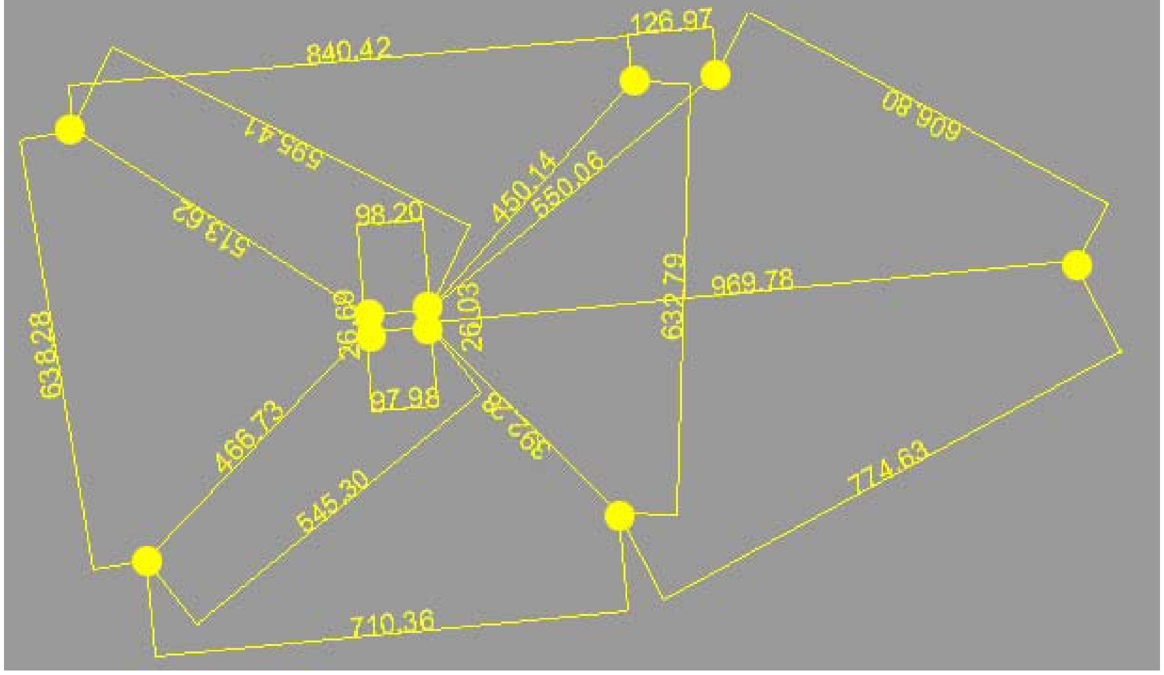
```
*DO,i,1,NELEMENTS
   ! Find the nodes at each corner
   *GET,node1,ELEM,i,NODE,1
   *GET,node2,ELEM,i,NODE,2
   *GET,node3,ELEM,i,NODE,3
   *GET,node4,ELEM,i,NODE,4

   ! Get the strains at each of these nodes
   *GET,stx1,NODE,node1,EPEL,X
   *GET,stx2,NODE,node2,EPEL,X
   *GET,stx3,NODE,node3,EPEL,X
   *GET,stx4,NODE,node4,EPEL,X

   *GET,sty1,NODE,node1,EPEL,Y
   *GET,sty2,NODE,node2,EPEL,Y
   *GET,sty3,NODE,node3,EPEL,Y
   *GET,sty4,NODE,node4,EPEL,Y

   *GET,stxy1,NODE,node1,EPEL,XY
   *GET,stxy2,NODE,node2,EPEL,XY
   *GET,stxy3,NODE,node3,EPEL,XY
   *GET,stxy4,NODE,node4,EPEL,XY

   !Average the four corner node strains
   stx(i) = (stx1 + stx2 + stx3 + stx4) / 4
   sty(i) = (sty1 + sty2 + sty3 + sty4) / 4
   stxy(i) = (stxy1 + stxy2 + stxy3 + stxy4) / 4

   *IF,stx(i),lt,stxmin,then
      stxmin = stx(i)
   *ENDIF

   *IF,sty(i),lt,stymin,then
      stymin = sty(i)
   *ENDIF

   *IF,stxy(i),lt,stxymin,then
      stxymin = stxy(i)
   *ENDIF

   *IF,stx(i),gt,stxmax,then
      stxmax = stx(i)
   *ENDIF
   *IF,sty(i),gt,stymax,then
      stymax = sty(i)
   *ENDIF

   *IF,stxy(i),gt,stxymax,then
      stxymax = stxy(i)
   *ENDIF

*ENDDO
*CFOPEN,strainyx,out
*DO,i,1,NELEMENTS
   stemp = stx(i)
*VWRITE,stemp
%g
*ENDDO
*CFCLOS

*CFOPEN,strainyy,out
*DO,i,1,NELEMENTS
   stemp = sty(i)
*VWRITE,stemp
%g
*ENDDO
*CFCLOS

*CFOPEN,strainyxy,out
*DO,i,1,NELEMENTS
   stemp = stxy(i)
*VWRITE,stemp
%g
*ENDDO
*CFCLOS

*CFOPEN,strminimax,out
*VWRITE,stxmin
%g
*VWRITE,stxmax
%g
*VWRITE,stymin
%g
*VWRITE,stymax
%g
*VWRITE,stxymin
%g
```

```
*VWRITE,stxymax
%g
*CFCLOS
```

APPENDIX C

DIMENSIONS FROM DIGITIZATION





Dimensions from digitization of the IPL. Dimensions shown are in mm. The IPL was digitized with a gauge length of one inch, zero x displacement and zero rotation.

APPENDIX D

MAPLE CODE

This program (dcm) finds the actuator lengths and angles for a desired array of deformations.

```
> restart:with(linalg):
```

Enter the coordinates of the pivot points:

```
> p1:=[-40.1084,-.4288]: p2:=[-11.9890,-12.2342]: p3:=[15.9810,-12.3611]:
> p4:=[-19.9685,12.4717]: p5:=[-14.9695,12.4959]: p6:=[18.1170,12.6797]:
```

Create the axis vectors

```
> a1:=p4-p1: a2:=p5-p2: a3:=p6-p3:
```

Find the length of each vector

```
> a10:=norm(a1,frobenius): a20:=norm(a2,frobenius): a30:=norm(a3,frobenius):
```

Create the displacement vectors

```
> npoints:=101:
> dxmin:=0; dxmax:=.2:
> dymin:=0; dymax:=.12:
> drmin:=0; drmax:=9:
> dx:=array(1..npoints):
> dy:=array(1..npoints):
> dr:=array(1..npoints):
> for i to npoints do dx[i]:=dxmin + (i-1)*(dxmax - dxmin)/(npoints - 1) od:
> for i to npoints do dy[i]:=dymin + (i-1)*(dymax - dymin)/(npoints - 1) od:
> for i to npoints do dr[i]:=drmin + (i-1)*(drmax - drmin)/(npoints - 1) od:
```

Create the vector of axis lengths for a set of displacements. Also determine the actuator angles for each displacement. Export this data to a file.

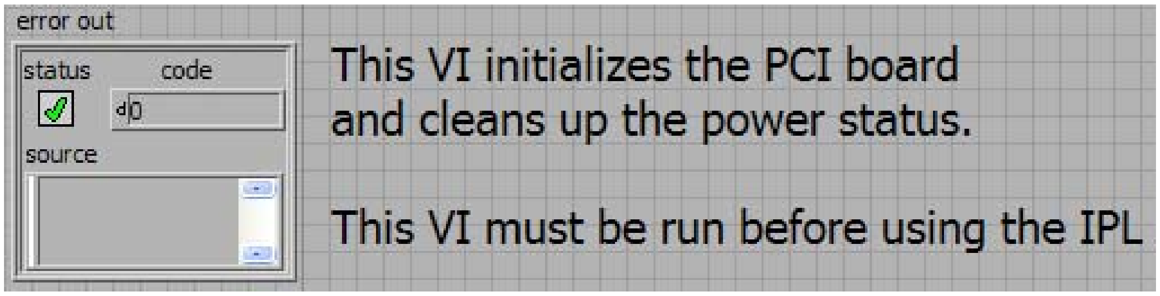
```
> datout:=array(1..npoints,1..12):
> for i to npoints do
>    p4ta:=[dx[i],dy[i]]+p4: p5ta:=[dx[i],dy[i]]+p5: p6ta:=[dx[i],dy[i]]+p6:
>    r4:=norm(p4ta,frobenius): r5:=norm(p5ta,frobenius): r6:=norm(p6ta,frobenius):
>    a4:=(angle(p4ta,[0,1])): a5:=(angle(p5ta,[0,1])): a6:=(angle(p6ta,[0,1])):
>    a4t:=evalf(a4+dr[i]*(Pi/180)): a5t:=evalf(a5+dr[i]*(Pi/180)):
>    a6t:=evalf(a6-dr[i]*(Pi/180)):
>    p4t:=[-r4*sin(a4t),r4*cos(a4t)]: p5t:=[-r5*sin(a5t),r5*cos(a5t)]:
>    p6t:=[r6*sin(a6t),r6*cos(a6t)]:
>    a1t:=p4t-p1: a2t:=p5t-p2: a3t:=p6t-p3:
>    a1:=norm(a1t,frobenius): a2:=norm(a2t,frobenius): a3:=norm(a3t,frobenius):
>    da1:=a1-a10: da2:=a2-a20: da3:=a3-a30:
>    st1:=round(50000*da1): st2:=round(50000*da2): st3:=round(50000*da3):
```

```
>    aa1:=evalf(angle(a1t,[1,0])): aa2:=evalf(angle(a2t,[1,0])):
>    aa3:=evalf(angle(a3t,[1,0])):
>    c1:=cos(aa1): s1:=sin(aa1): c2:=cos(aa2): s2:=sin(aa2): c3:=cos(aa3): s3:=sin(aa3):
>    datout[i,1]:=st1: datout[i,2]:=st2: datout[i,3]:=st3:
>    datout[i,4]:=dx[i]: datout[i,5]:=dy[i]: datout[i,6]:=dr[i]:
>    datout[i,7]:=c1: datout[i,8]:=s1: datout[i,9]:=c2:
>    datout[i,10]:=s2: datout[i,11]:=c3: datout[i,12]:=s3:
> od:
> fd:=fopen(dat,WRITE,TEXT):
> writedata(fd,datout):
> fclose(fd):
```

APPENDIX E

LABVIEW CODE

The initialize program is used to get the IPL ready to move. This program has to be run before any testing is done.



error out

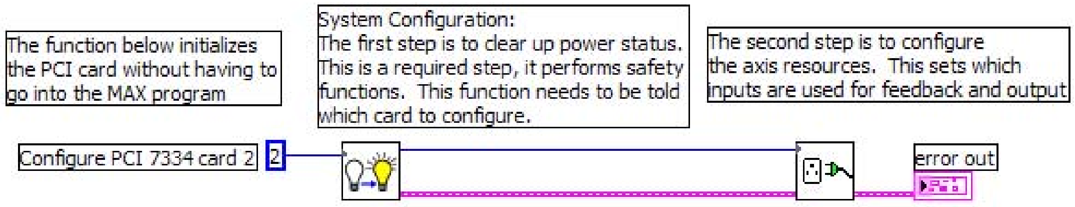
status code

d0

source

This VI initializes the PCI board and cleans up the power status.

This VI must be run before using the IPL



The function below initializes the PCI card without having to go into the MAX program

System Configuration:
The first step is to clear up power status. This is a required step, it performs safety functions. This function needs to be told which card to configure.

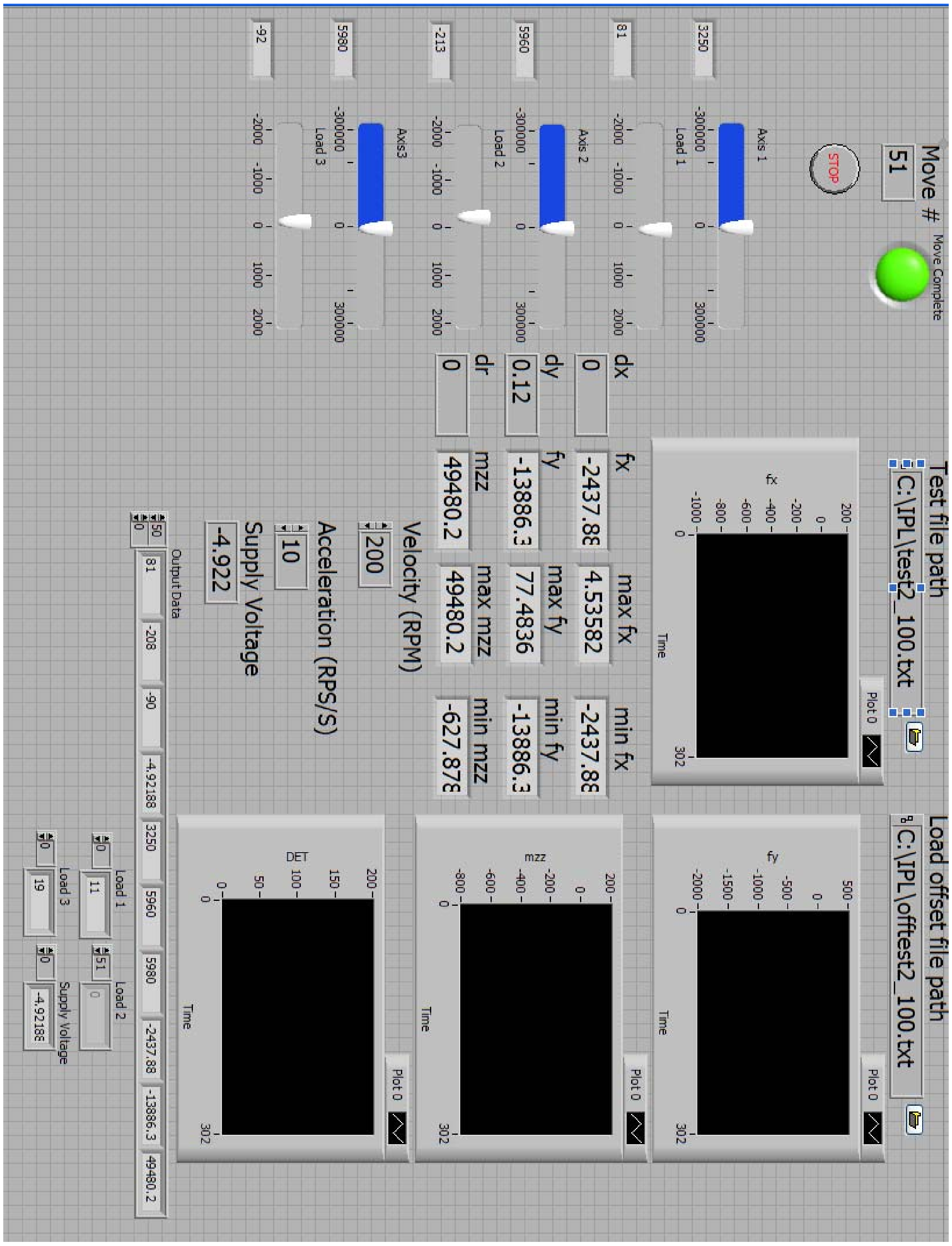
The second step is to configure the axis resources. This sets which inputs are used for feedback and output

Configure PCI 7334 card 2

2

error out

The following interface and code are for running a test in the IPL. All subVI's will be documented after this program.


Move # 51

Move Complete

Test file path C:\IPL\test2_100.txt

Load offset file path C:\IPL\offtest2_100.txt

Axis 1 3250

Load 1 81

Axis 2 5960

Load 2 -213

Axis3 5980

Load 3 -92

dx 0

dy 0.12

dr 0

fx -2437.88

fy -13886.3

mzz 49480.2

max fx 4.53582

max fy 77.4836

max mzz 49480.2

min fx -2437.88

min fy -13886.3

min mzz -627.878

Velocity (RPM) 200

Acceleration (RPS/S) 10

Supply Voltage -4.922

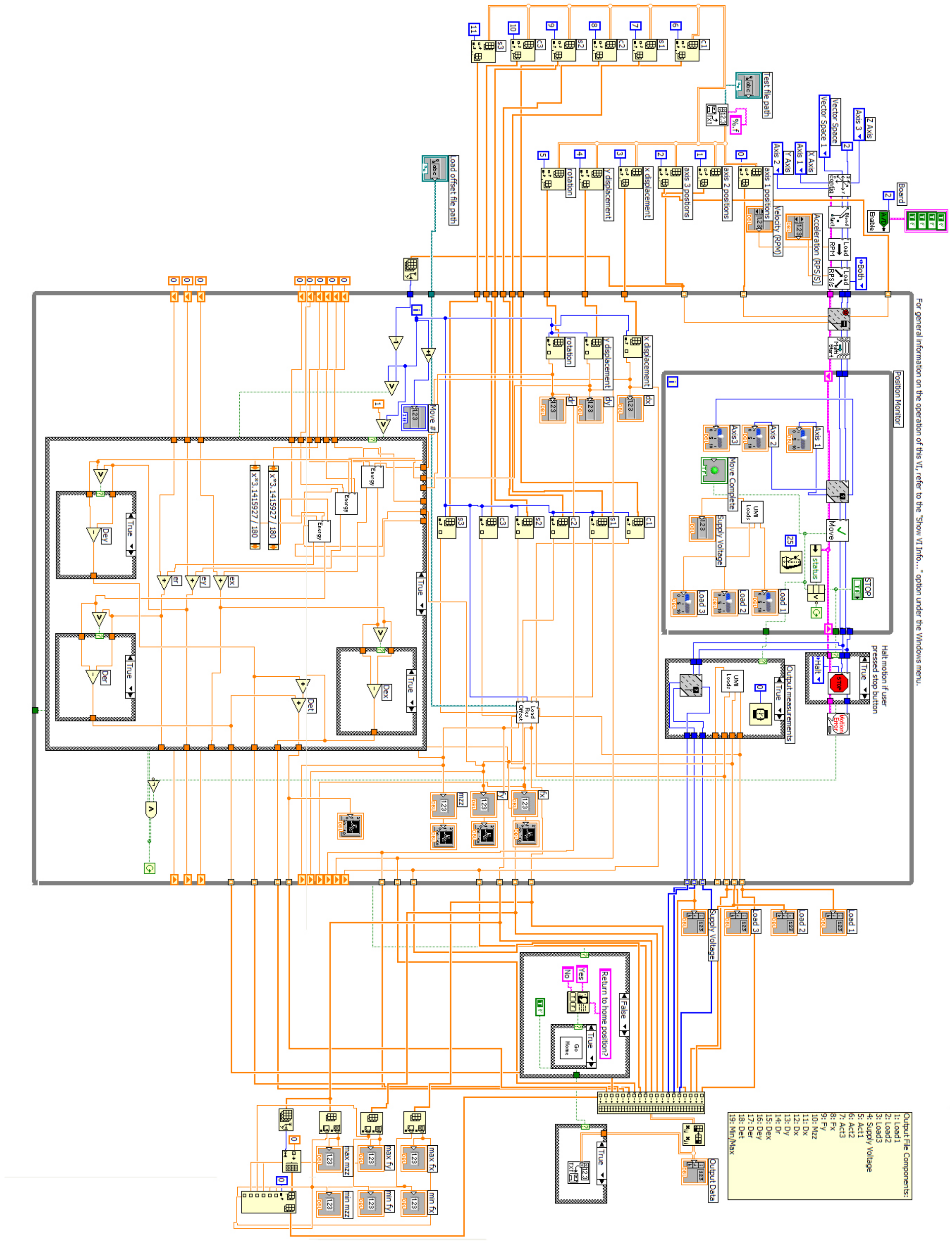
Output Data 50 0: 81, -208, -90, -4.92188, 3250, 5960, 5980, -2437.88, -13886.3, 49480.2

Load 1 0: 11

Load 2 51: 0

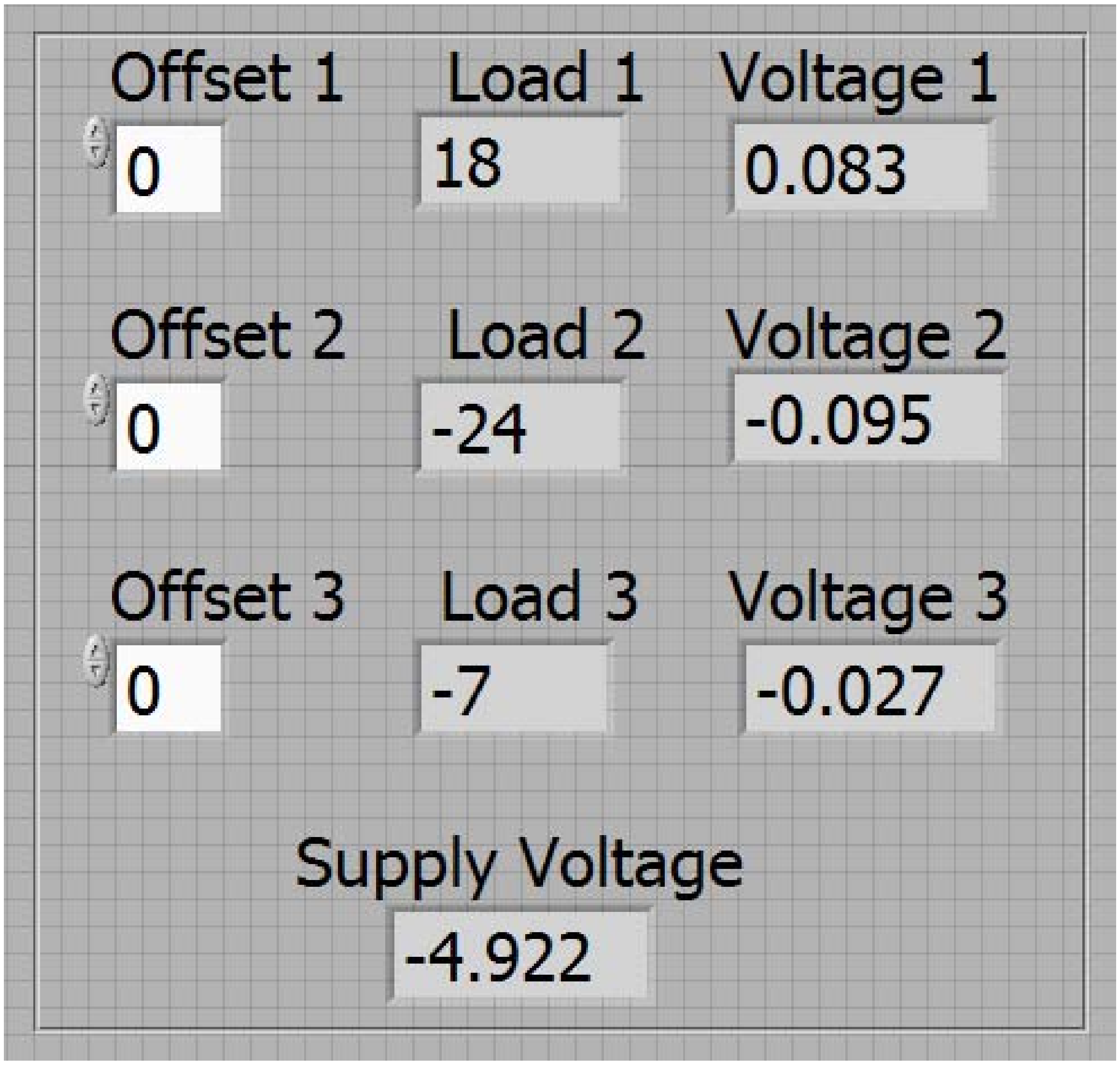
Load 3 0: 19

Supply Voltage 0: -4.92188



Test file path
Load offset file path
Vector Space 1
Vector Space
Axis 1
Axis 2
Axis 3
X Axis
Y Axis
Z Axis
Board
Enable
Both
axis 1 positions
axis 2 positions
axis 3 positions
x displacement
y displacement
rotation
Velocity (RPM)
Acceleration (RPS/S)
c1
c2
c3
s1
s2
s3
dx
dy
dr
Move #
Position Monitor
Move Complete
Supply Voltage
UMI Loads
Load 1
Load 2
Load 3
Move
status
STOP
Halt motion if user pressed stop button
Halt
Output measurements
True
False
Dex
Dey
Der
Det
ex
ey
er
Energy
x*3.1415927/180
Load Rss Offset
Fx
Fy
mzz
Return to home position?
Yes
No
Go Home
max fx
max fy
max mzz
min fx
min fy
min mzz
Output Data
Output File Components:
1: Load1
2: Load2
3: Load3
4: Supply Voltage
5: Act1
6: Act2
7: Act3
8: Fx
9: Fy
10: Mzz
11: Dx
12: Dx
13: Dy
14: Dr
15: Dex
16: Dey
17: Der
18: Det
19: Min/Max
For general information on the operation of this VI, refer to the "Show VI Info..." option under the Windows menu.

The following subVI is shown as UMI loads on the test VI above. It reads the voltages from the load cells and converts them into load values. The interface is shown first and the block diagram is shown second.



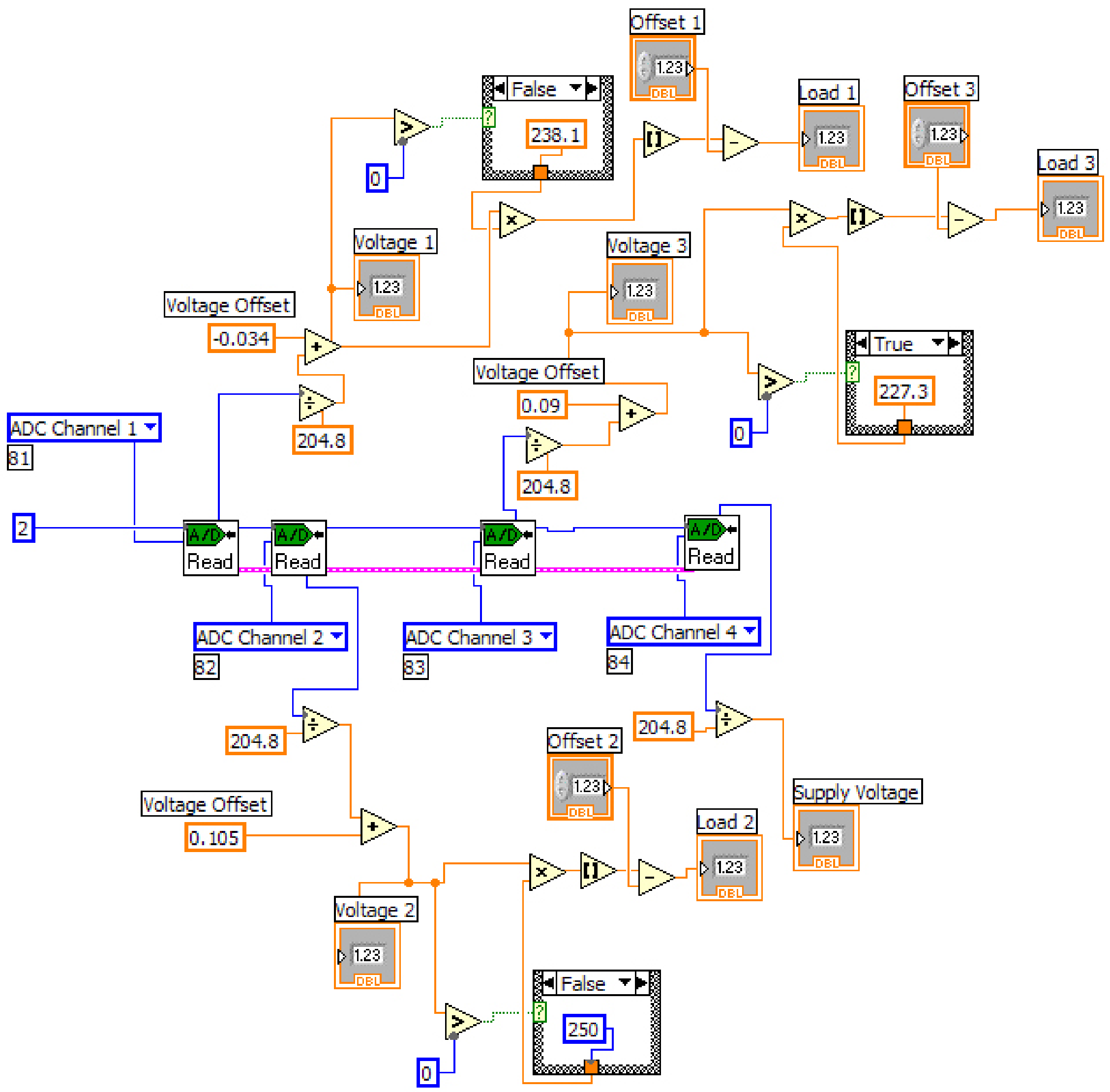
Offset 1	Load 1	Voltage 1
0	18	0.083

Offset 2	Load 2	Voltage 2
0	-24	-0.095

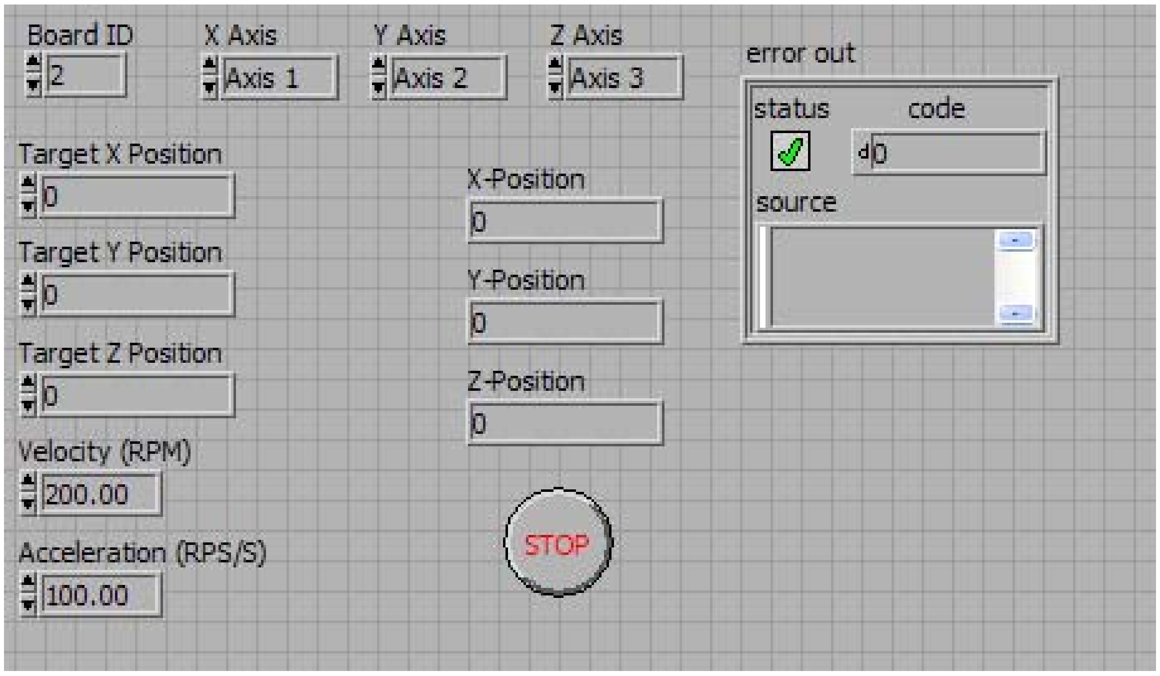
Offset 3	Load 3	Voltage 3
0	-7	-0.027

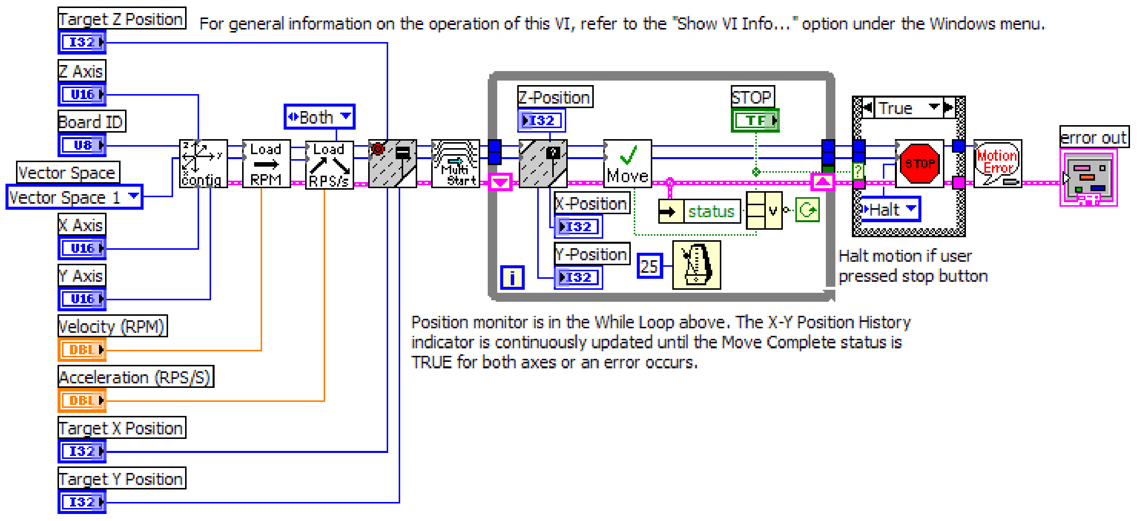
Supply Voltage

-4.922


Offset 1
1.23
DBL
Load 1
1.23
DBL
Offset 3
1.23
DBL
Load 3
1.23
DBL
False
238.1
0
Voltage 1
1.23
DBL
Voltage 3
1.23
DBL
Voltage Offset
-0.034
True
227.3
Voltage Offset
0.09
0
204.8
204.8
ADC Channel 1
81
2
A/D
Read
A/D
Read
A/D
Read
A/D
Read
ADC Channel 2
82
ADC Channel 3
83
ADC Channel 4
84
204.8
204.8
Offset 2
1.23
DBL
Supply Voltage
1.23
DBL
Voltage Offset
0.105
Load 2
1.23
DBL
Voltage 2
1.23
DBL
False
250
0

The program go home is used to make the IPL return to the home position, a gauge length of 1 inch.



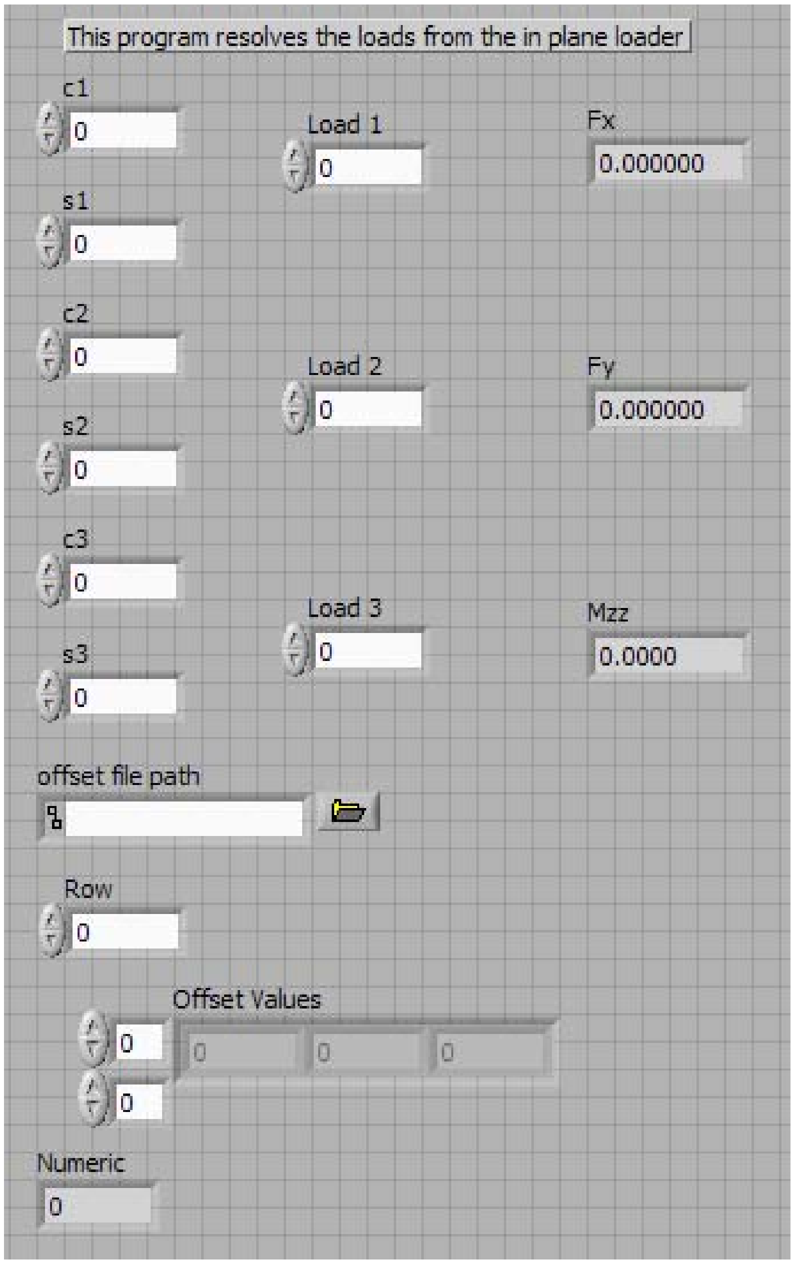


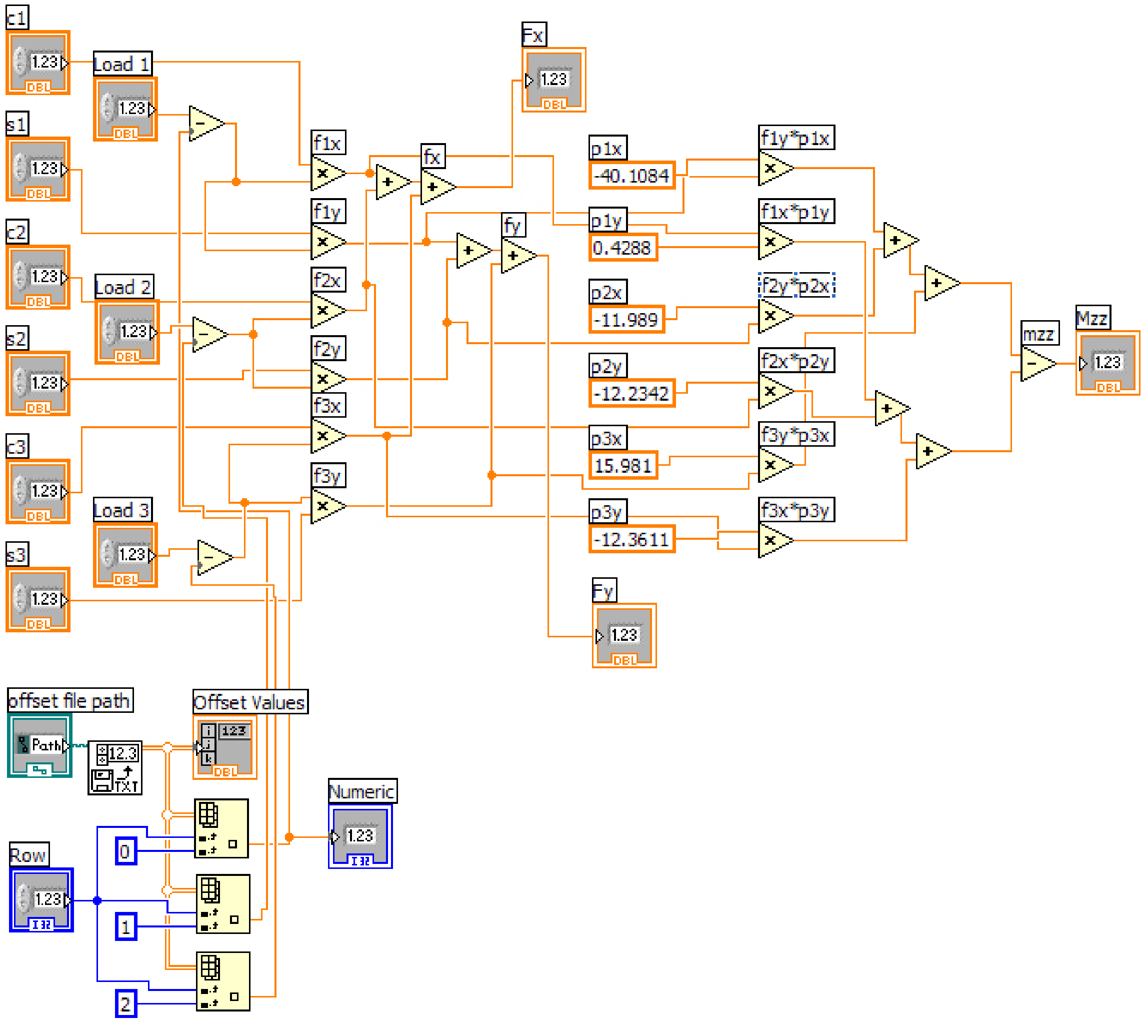
For general information on the operation of this VI, refer to the "Show VI Info..." option under the Windows menu.

Halt motion if user pressed stop button

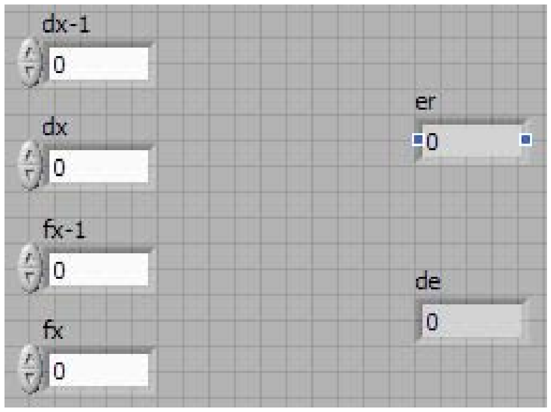
Position monitor is in the While Loop above. The X-Y Position History indicator is continuously updated until the Move Complete status is TRUE for both axes or an error occurs.

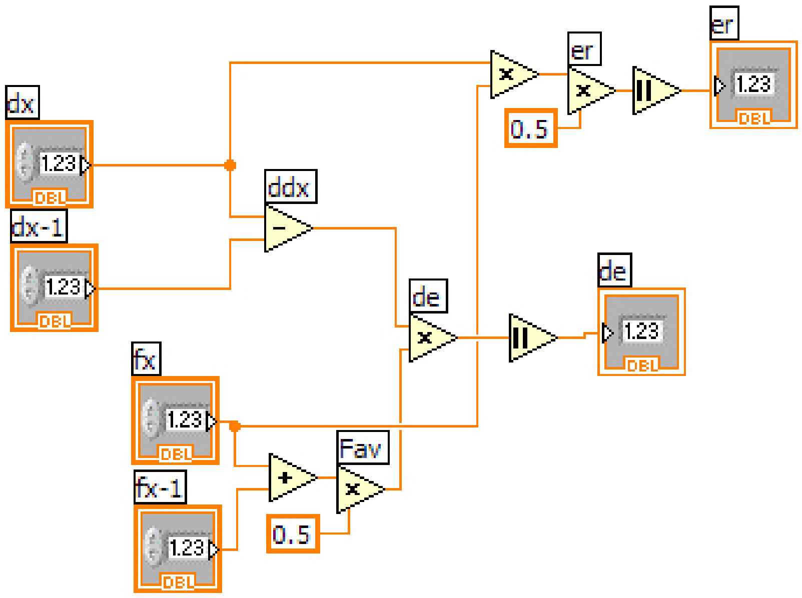
Load res off is used to resolve the loads into the x, y and r components.




c1
s1
c2
s2
c3
s3
Load 1
Load 2
Load 3
f1x
f1y
f2x
f2y
f3x
f3y
fx
fy
Fx
Fy
p1x
-40.1084
p1y
0.4288
p2x
-11.989
p2y
-12.2342
p3x
15.981
p3y
-12.3611
f1y*p1x
f1x*p1y
f2y*p2x
f2x*p2y
f3y*p3x
f3x*p3y
mzz
Mzz
offset file path
Path
Offset Values
Numeric
Row
0
1
2

The energy program was used to calculate the dissipated energy in each dimension as the test was running.


dx-1
0
dx
0
fx-1
0
fx
0
er
0
de
0


dx
1.23
DBL
dx-1
1.23
DBL
ddx
fx
1.23
DBL
fx-1
1.23
DBL
Fav
0.5
de
er
0.5
er
1.23
DBL
de
1.23
DBL