Formulation of a Classical Laminate Theory (CLT) Based Finite Element Analysis (FEA) Suite

ESC499 Undergraduate Thesis Final Report

Author: Linxi (Gary) Li

Supervisor: Dr. Craig Steeves

April 16, 2021

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

Abstract

A key building block to the development of fiber optimization and design methodologies is having a rigorous laminate analysis suite using methods such as Finite Element Analysis (FEA). In the existing literature, there is a scarcity in the detailed derivation of low order model based laminate FEA, which are easy to understand intuitively and to implement. Also, there is a lack of assessment of performance when different element types are used. The work documented in this manuscript contributes to both mentioned topics. A Classical Laminate Theory based FEA model was thoroughly derived. Two shell elements are incorporated in the derivation, formulated by combining a plane-stress element with a conforming or nonconforming C^1 plate bending element. The FEA was implemented into MATLAB and was verified using rigid body mode analyses as well as comparing its output with ABAQUS, a commercial FEA software, using archetypal problems of: edge bounded square plate subjected to distributed load and cantilever bending. Results from square plate case studies showed that the ability of the MATLAB FEA to predict transverse deformation neglecting transverse shear contribution is independent to the variation of laminate length to thickness ratio. The cantilever test case showed that the MATLAB FEA can predict, with high accuracy and consistency, the deformation of laminates whose layouts does not result in significant membrane-bending coupling. The performance comparison between the two shell elements were also elucidated using the mentioned case studies. In general, the accuracy of both elements is extremely similar for all the analyzed cases, hence suggesting that they have similar capabilities. However, the nonconforming related shell element requires less computational resource, hence making its overall performance more optimal compared to the conforming related shell element.

Acknowledgements

Just as it takes a village to raise a child, it takes a dedicated and supportive group of professors, friends and family members to complete a research project. Hereby I would like to pay my sincere appreciation to everyone who has contributed to my undergraduate thesis journey. I am really grateful for all the love and support I have received for the past year of studies. I would like to especially thank my supervisor, Dr. Craig Steeves, for granting me this opportunity to work in-depth on an exiting and fruitful project. Dr.Steeves' was always welling to shear his wealth of engineering knowledge with me. His insights, guidance, as well as timely nudges to help me over difficulties were crucial to the advancement of this project. A special thanks also goes out to my parents, my aunt and my uncle. They have been super supportive to my education and my well-being, which ensured that I was able to smoothly complete my final undergraduate year, albeit the awkward online learning environment.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Objectives

Fiber-reinforced laminate composites have historically been a sought-after material in the aerospace industry due to their high stiffness and strength to weight ratios, making them suitable for high-performance and weight-sensitive applications. Since the application of composite materials to the Airbus A380 wingbox, fiber composites have begun seeing extensive usage as primary structural components in passenger airlines. This trend necessitated advanced design and analysis techniques to fully understand the behaviour of fiber composites in various application scenarios [10, 13, 24, 27, 31, 34, 38, 45]. Consequentially, structural optimization problems involving fiber laminate composites have been in the forefront of research for the aerospace industry. Typically in these optimization problems, a selected material property of the fiber laminate, such as the mass or the compliance of the structure, is optimized subject to constraints, such as the geometry or manufacturing limitations, in order to obtain the composite layout with the optimal performance characteristics when subjected to a desired set of boundary conditions, such as external applied loads [5, 15, 19, 37, 42, 47]. Due to the construction of the fiber laminate, the optimization problems tend to be characterized by a potentially large number of design variables, including ply geometry, matrix material selection and distribution, fiber angle orientation, as well as stacking sequence of the plies [1, 15, 42]. In the practical aspect of manufacturing, the recent advance of technologies such as automated fiber placement enables the production of laminates that have curvilinear fiber patterns [18, 46, 48]. This development spurred increasing research interests in the design of tailored fiber configurations, which are laminates with specific fiber strand orientations for desired applications, hence an optimization problem with fiber angle as the main design variable.

This project hopes to further advance the research on tailored composite by developing and investigating a novel methodology for designing fiber angle layout through structural optimization. The overarching approach proposed for the methodology involves decomposing the structural requirements for a fiber laminate from a given design problem, so that each ply of the laminate is constructed individually, and their structural performance is subtracted from the design problem after construction. This process is carried out sequentially until the combined laminate satisfy all the problem requirements. To represent the fiber angle layouts in the optimization algorithm efficiently, the fiber patterns at each layer is parameterized by level set functions constructed using different functional basis. Specifically, the choices under investigation include radial basis functions used as compact support basis and Fourier series used as globally supported basis. The optimization process is based on gradient descent algorithms, which will be used to update key design variables, thus tuning the functional representation of the fiber layout.

Since this project is in the early stage, the investigation for this thesis will first focus on rigorous construction of a laminate Finite Element Analysis (FEA) suite that will be used in the future to support optimization analysis of fiber layouts. Hence, the main purposes of this manuscript include selecting of an appropriate theory modeling fiber laminate composites. Subsequently, the modeling theory will be used as the basis for the FEA suite, as a thorough and detailed mathematical derivation will be provided to formulate the necessary equations for implementation of the FEA suite in a computer program. The accuracy and capability of the FEA module will then validated using simple archetypal problems, such as membrane deflection and cantilever bending problems. In addition, these benchmarking case studies are expected to provide a performance comparison between a couple of FEA element choices. This in turn will allow the users to make more informed element selection based on application scenarios.

Chapter 2

Background

2.1 Lamina and Laminate in Composite Materials

Composites are materials that consist of two or more constituent material in different physical or chemical phases, and are engineered so that the individual components interact in a complementary manner to enhance the chemical and physical properties of the overall material [10, 29, 40]. Fiber metal laminates, ceramics, polymers, and sandwich structures are all composites. This work will be focused primarily on modeling fiber reinforced polymer composites (FRP). FRP composites, as shown in Figure 2.1, are made from combining, at the macroscopic scale, a polymer-based matrix with man-made or nature fibrous strands. The polymer matrix is load transferring medium and bonds fibers together, while the fibers provide the majority of the strength and stiffness. One could further distinguish between different types of FRPs by looking at the distribution of the fibers. In general, there are four types of fiber configurations as illustrated in Figure 2.2.

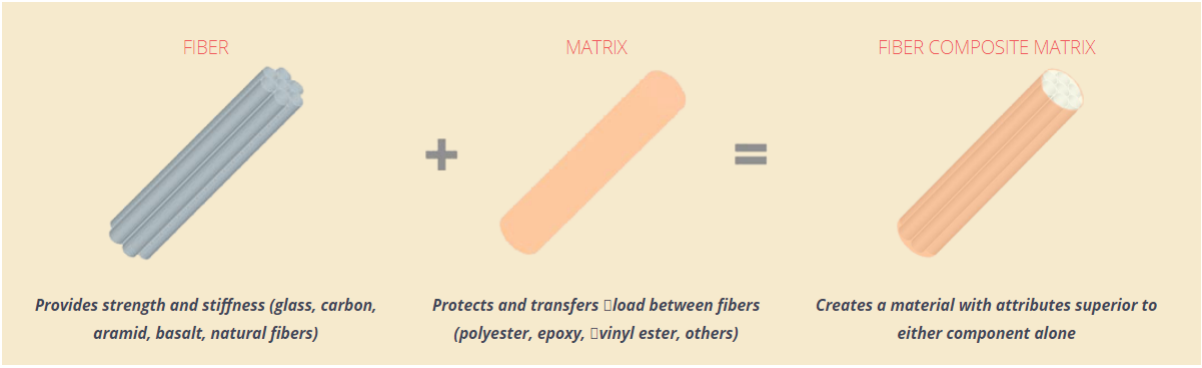

Figure 2.1: Macrscopic constituent of Fiber Reinforced Polymer composites [44]

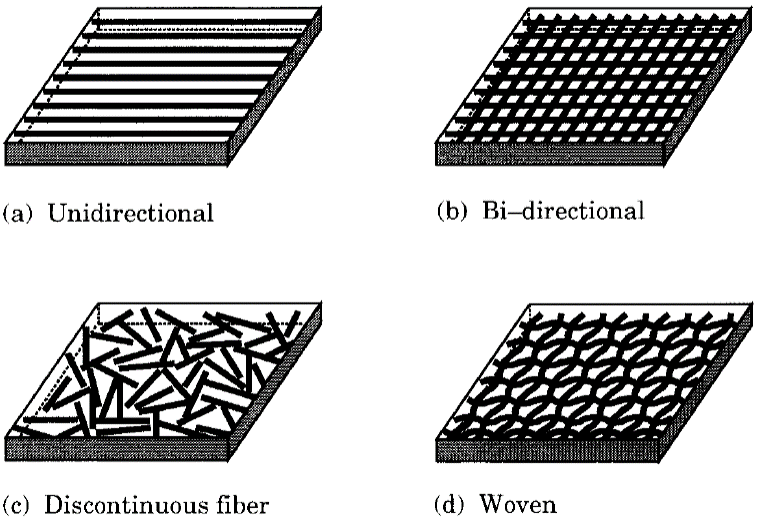

Figure 2.2: Different varieties of fiber configurations. [29]

The building block of an FRP structure is called a lamina or ply, which is a sheet of fiber strands embedded in polymer matrix. Laminate refers to a stack of individual lamina, typically bonded together using the same polymer matrix material. For the case of unidirectional fiber distribution, each lamina layer will have a specific fiber angle orientation (as shown in Figure 2.3), and the overall laminate structural properties will be directly related to the specific sequence of lamina layering. Equipped with the definition of lamina and laminate, it is very important to note that there is a substantial difference between a lamina constitutive model and laminate material theories. Lamina constitutive model refers to the modeling of the mechanical stress and strain behaviour of a single layer or ply with a specific fiber orientation using the generalized Hooke's Law. Laminate material theories on the other hand define both the strain-displacement and the force-stress relationships, thus specifying the interactions between adjacent lamina layers. In other words, lamina constitutive model, when applied with specific assumptions, can be incorporated into multiple different laminate material theories, and the material theories will be distinguished by their unique definition of strains as functions of displacement and rotation degrees of freedom.

For the purpose of formulating a simple laminate analysis suite for preliminary level study, the material modeling will only consider laminate plate formed by unidirectional fiber ply layers.

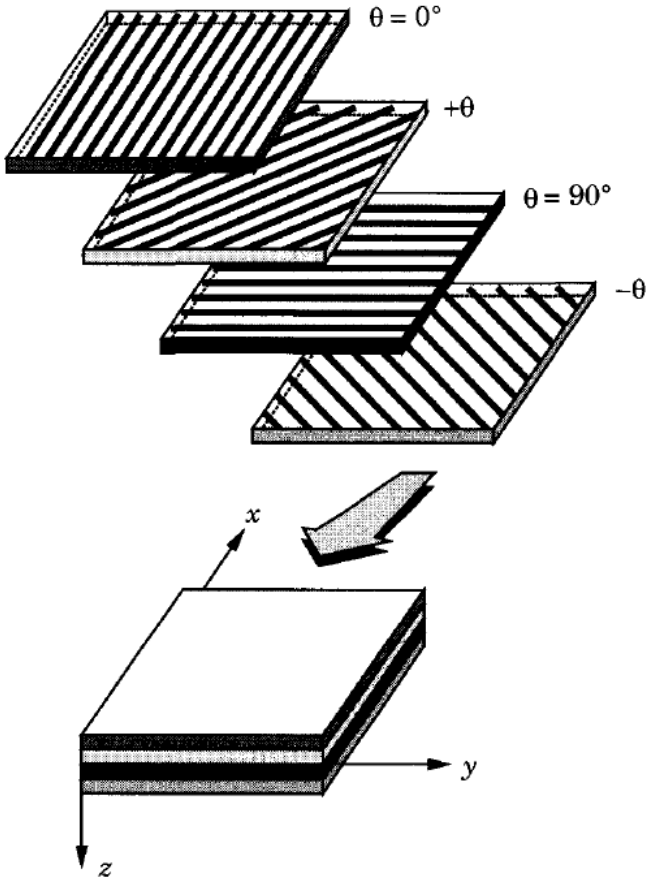

Figure 2.3: Example of a laminate plate established by layering of unidirectional lamina with specific fiber angle orientations. [29]

2.1.1 Lamina Constitutive Model

A lamina constitutive model is generated using the generalized Hooke's Law, thus relating stress with strain using material properties. Stress and strain are tensorial properties that are defined for an infinitesimal element in a material. When a particular coordinate system is specified, the strain and stress tensor components can be expressed using notations of ϵ_{ij} and σ_{ij} respectively, where the i subscript denotes the coordinate axis parallel to the normal of the plane the stress/strain is acting on and the j index describes the positive direction for stress/strain acting on that surface. In the 3D Cartesian coordinate context, stress and strain tensors will have nine component each, and their distribution is as illustrated in Figure 2.4. Thus, the ϵ_{ii} and σ_{ii} are denoted as the axial strain and stress terms, while other ϵ and σ terms are the shear strain and shear stress terms. Without the loss of generality, the rest of lamina constitutive model derivation will be done in a Cartesian coordinate system. Equations in other coordinate systems, such as curvilinear ones, can always be obtained by

applying appropriate coordinate transformation procedures.

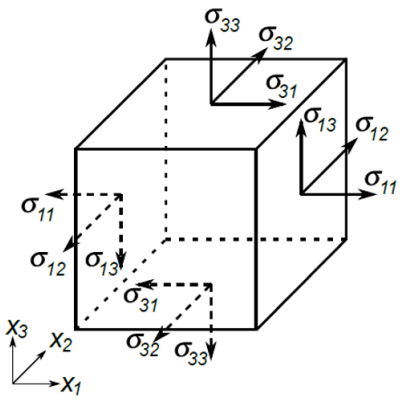

Figure 2.4: Stress tensor components acting on an infinitesimal material cube defined in a 3D Cartesian coordinate system.[40]

The fundamental assumption of generalized Hooke's Law is that the material behaves in a linear elastic manner. Thus, the stress tensor can be expressed as the linear product between strain tensor and a fourth order tensor (C_{ijkl}), as shown in Equation 2.1.

$$\sigma_{ij} = C_{ijkl}\epsilon_{kl} = \sum_{k=1}^{3}\sum_{l=1}^{3} C_{ijkl}\epsilon_{kl} \tag{2.1}$$

The fourth order tensor contains 81 constants related to material properties, however it can be reduced when certain assumptions are incorporated. Fiber laminates are typically modeled as orthotropic materials, meaning that the material inherently has three orthogonal planes of symmetry, where the material properties stays constant along the normal direction of the planes. For fiber laminates, the three orthogonal planes, as demonstrated in Figure 2.5, are: one plane with normal vector parallel to the direction of fiber, and two planes with normal vector orthogonal to the direction of fiber and also orthogonal to each other. Assuming the existence of strain energy density (U') of the form $U' = \frac{1}{2}\sigma_{ij}\epsilon_{kl}$, for orthotropic materials in a Cartesian coordinate system, the strain and stress relationship can be simplified into matrix products of the form:

$$\begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ \gamma_{23} \\ \gamma_{13} \\ \gamma_{12} \end{bmatrix} = \begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{21}}{E_2} & -\frac{\nu_{31}}{E_3} & 0 & 0 & 0 \\ -\frac{\nu_{12}}{E_1} & \frac{1}{E_2} & -\frac{\nu_{32}}{E_3} & 0 & 0 & 0 \\ -\frac{\nu_{13}}{E_1} & -\frac{\nu_{23}}{E_2} & \frac{1}{E_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{23}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{13}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{12}} \end{bmatrix} \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{bmatrix} \tag{2.2}$$

In Equation 2.2, the 6 by 6 matrix is known as the compliance matrix, which constitutes from various material constants. Specifically, E_i is the Young's modulus along the i axis, ν_{ij} is the Poisson's ratio describing the contraction in direction j when stress is applied in direction i, and G_{ij} is the shear modulus describing the effect of shear stress σ_{ij}. Also $\gamma_{ij} = 2\epsilon_{ij}$ are defined as the engineering shear strains. For unidirectional lamina, one of the common assumptions applied is that the distribution of fibers in both of the transverse directions (perpendicular to the fiber orientation) are similar in packing density. As a result, the material responses in those directions are approximately equivalent. The key result of this assumption is that the compliance matrix in Equation 2.2 is symmetric, thus reducing the amount of independent material constants.

For the sake of clarity, the 1 direction will correspond to the principal fiber orientation, the 2 direction will refer to the in-plane perpendicular direction, and the 3 direction will be the out-of-plane perpendicular direction, which corresponds to the lamina stacking direction. Equation 2.2 is the key lamina constitutive equation which will be used as the basis for formulating laminate material models.

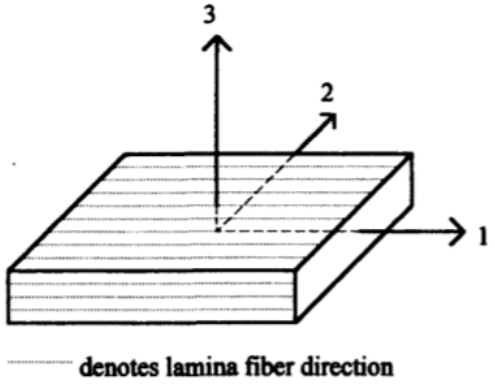

Figure 2.5: Vectors denoting the normal direction of the three planes of material symmetry for fiber lamina.[26]

2.2 Fiber Laminate Analysis Suite

As an integral part of the design procedure, a fiber laminate analysis tool is required to analyze the static structural response of the laminate, specifically the displacement and strain response, given a specified layout sequence and a set of imposed boundary conditions. In this research, Finite Element Analysis (FEA) will be used to accomplish the said task.

Two of the key ingredients in formulating FEA for laminates are the selection of the material theory and lamina constitutive model, as well as the choice of element types. Both factors effects the ability of the model to capture the physics of the material response. In the open literature, there is a rich collection of derivations for each aspect individually [27, 31]. However, as the methods of analysis became increasingly sophisticated, there is a deficiency in guidelines for supporting the selection of model based on the desired task [27, 31]. To exacerbate the issue, model validation with experimental methods is typically difficult or unfeasible with fiber laminate structures. These factors necessitate a detailed investigation to construct rigorous and appropriate finite element models to support the fiber laminate analysis.

Due to the heterogeneous and orthotropic nature of fiber laminates [10, 13, 45], models (in terms of both constitutive model and element type) with a wide range of complexity have been formulated by various researchers [27, 31]. Hence, the development of a comprehensive guide for model selection is deemed outside of the scope of the project. In hindsight, since the design method proposed by this project is still within the preliminary phase, choosing an overly complex FEA model would impose difficulties in terms of understanding the underlying physical implications, model implementation,and also computational efficiency. Thus, selection of appropriate FEA models for this project was based on metrics of: ease in understanding the physical principles, readiness to integrate numerically with code, and computational effort necessary to reach preliminary level of accuracy.

2.2.1 Laminate FEM Literature Review and Model Selection Decision

This section presents a brief summary of the available literature regarding FEA formulation for fiber laminates, in both aspect of material model and element types, thus laying out the groundwork for subsequent determination of optimal parametrisation using the metrics described in the previous section.

In general, the two major categories of fiber laminate material models are the equivalent single layer (ESL) theory and the layerwise theories [21, 27, 31]. The ESL theory is

based on the assumption that the displacement field of the lamina is at least C^1 continuous through the thickness. As a result, the lamina layout can be treated as statically equivalent to a single plane with averaged structural characteristics. The main assumption applied to the lamina constitutive model is that the transverse direction principal stress and strain are both zero [9, 14, 17, 21, 27, 31]. Hence, the analysis will be two dimensional (2D) and based on the modulus-weighted mid-plane of the laminate. The variation of the displacement field through the thickness of the laminate is accounted for by incorporating a function that is dependent on the derivatives of the transverse displacement of the mid-plane and the through-thickness position of each layer [9, 14, 17, 21, 27, 31]. The two lowest order theories in the ESL hierarchy are the classical laminate theory (CLT) and first order shear deformation theory (FSDT) [9, 14, 17, 21, 27, 31]. CLT was formulated based on Kirchoff thin plate theory, which is an 2D extension of the Bernoulli beam theory, as the through-thickness displacement field variation is linearly proportional to the first derivatives of the mid-plane through-thickness displacement, which approximates the out-of-plane rotations. The formulation of CLT implies that it only has 3 axial displacements as independent degrees of freedom, thus it only considers the in-plane displacements and out-of-plane bending as the deformation modes for the structure. However, it neglects the transverse shearing effect as the accuracy of the model deteriorates for thick laminates [14, 17, 21, 27, 31]. The FSDT on the other hand relaxes the assumption imposed by the thin plate theory by decoupling the out-of-plane rotations as independent degree of freedoms, resulting in 5 total degrees of freedoms. This is based on the Mindlin thick plate theory, which is an 2D extension of the Timoshenko beam theory, and comparing to CLT it accounts for the additional contribution of transverse shear to the structural deformation. However, the FSDT requires a shear correction factor, which is generally unknown for arbitrary fiber laminate structures, to fully account for the transverse shear [9, 14, 17, 21, 27, 31]. The other drawback of FSDT is related to the numerical implementation process, as it encounters shear-locking problem when dealing with relatively thin laminate plates. It is interesting to note that higher order shear deformation theories presented in literature are typically derived either by extending the displacement field dependencies to include products of high order through-thickness position terms with rotational degree of freedoms, or by incorporating additional assumptions to approximate in-plane displacements [9, 14, 17, 21, 27, 31]. For instance, Reddy formulated a third-order theory by assuming the transverse shear distribution vanishes on the outer surfaces of the laminate [28]. The resultant displacement field equations present an increase in accuracy compared to FSDT in addition to avoiding the need for the shear correction factor. Idlbi et al. on the other hand included a sinusoidal term in the in-plane displacement approximation equation [16]. In general, the supplement of higher order terms results in

higher model accuracy, thus also allowing the equations to be tailored towards specific structural problems. Nevertheless, they also increase the complexity of the material displacement equation, consequentially leading to more degrees of freedom which are less intuitive to interpret, higher shape function continuity constraints when incorporated into FEA, and also significant increase in computational cost [17, 27, 31]. These are all shortcomings of higher order ESL models.

Layerwise theory alleviates the transverse continuity constraint of displacement field imposed by the ESL down to C^0, thus allowing each lamina to be considered individually [17, 27, 31, 35]. This class of models performs better (compared to the ESL theories) in terms of capturing the discrete layer transverse strain effects by emphasizing the discontinuous nature of the material behaviours through the layers. To maintain realistic kinematics for the interaction of layers, this class of theory still requires the displacement field to be continuous throughout the layers. This requirement is achieved by enforcing compatibility equations at layer interfaces and also equilibrium conditions within the layer itself [17, 27, 31, 35]. Hence, the resultant displacement field equations are formed by a linear combination of products of in-plane coordinate dependent function with through-thickness coordinate dependent function. The method of compatibility enforcement distinguishes layerwise theories from each other, and hence it is still an intensive research topic in the field of fiber laminate modeling [17]. In general, most layerwise theories falls into two categories. One category of methods invokes the compatibility equation laboriously and attempts to solve the geometric property of each layer individually. These methods results in high number of displacement functional degrees of freedom which scales with the number of layers. Hence the computational complexity in solving these equation tends to be very high [17, 27, 31, 35]. However, they do have the ability to model laminate failure behaviours. For instance, disregarding the continuity constraint at specific interface that exhibits delamination. The other category of methods additionally applies a-priori assumptions to the geometry and stress distributions of the laminate through the thickness, thus regarding the displacement functional degrees of freedom as independent from the number of layers, effectively reducing the total number [17, 35]. However, the additional assumptions will result in higher continuity requirements on the shape function when incorporated into FEA analysis.

As evident from the brief overview of the various laminate model types, layerwise theories are more sophisticated and thus offers a better modeling accuracy compared to ESL theories. Layerwise theory captures the transverse behaviour better and some are capable of modeling delamination failure. However, the trade-offs of having high model resolution are that the compositions of the laminate displacement equations are significantly less intuitive, the model's 3D nature or added functional degree of freedom necessitates high dimension or

high order elements when integrated into FEA, as well as the requiring increased computational effort. For the purpose of a preliminary analysis, the modeling accuracy only needs to be sufficient in the elastic domain, hence trading off numerical economy as well as the ease of implementation and interpretation is not worth the increase of solution accuracy achieved with complex models.

On the other hand, the intuitiveness of low order ESL theories has made them easily applicable to initial stages of design and research projects. Their usefulness can be illustrated through various composite literature. For instance, in an exploratory study to consider the load bearing capacity of fused filament fabricated fiber reinforced thermoplastic, Choi and Kortschot [7] used CLT to validate the experimental measurements of the mechanical properties of the material. The success of the validation prompted the authors to consider advanced analysis by incorporating CLT with FEA. Also, Cichosz [8] formulated a CLT-based framework for macroscopic modeling of bi-axial braided composites, to address the issue of the lack of established models for prediction of braided composite mechanical properties.

Furthermore, since the overarching goal of the research is to create and assess an optimization based fiber design method, eventually the laminate analysis suite will be expected to handle curvilinear continuous fiber orientations. Conveniently, there are existing methods documented in literature that are pertinent. For instance, Ribeiro et al. [30] demonstrated the idea of re-deriving the constitutive matrices to be functions of in-plane coordinates, thus allowing the material model to continuously follow the fiber curvature. Ferreira et al. [12] addressed this problem in a FEA framework by taking the fiber angle in each element to be a constant calculated from weighted sum of the fiber path slopes at the element nodes. Haystead [15] also illustrated the usefulness of considering the fiber angles as piecewise continuous functions of the in-plane coordinate to support optimization analysis.

Ultimately, the low order ESL class of models, specifically the CLT, was selected as the preferred laminate modeling method in this project. The simple nature of CLT means that is optimal for preliminary research stage implementation, in addition to having the potential of adapting to curvilinear fiber layout analysis.

In terms of the choices of apposite element type for composite analysis, typically two types, namely shell elements and 3D continuum/solid elements, are frequently considered in literature and in commercial FEA packages [2, 9, 27, 29, 34, 45]. Shell elements can also be split into two groups: conventional shell and continuum shell elements. The continuum shell elements is similar to a 3D solid element as both only have translational degree of freedom at their nodal points. Thus, both of these elements necessitate 3D discretization of the structure in order to perform analysis [2, 9, 27, 29, 34, 45]. Nevertheless, since the laminate model of interest in this research are the ESL models, 2D element types are sufficient for

analysis. Hence, conventional shell elements are preferred, as they have both translational and rotational degrees of freedom at the nodes, thus only requiring 2D meshing. Regarding to the element components, for this project, the four-noded quadrilateral (Q4) elements will be used in combination with shape functions of appropriate continuity. The Q4 elements have a simple rectangular geometry, thus making the meshing process and the shape function definitions straightforward. Thus the simple nature of Q4 elements is in accordance with the goal of the preliminary study.

2.2.2 Identification of Knowledge Gaps

Substantial bodies of knowledge have been built in the subject of detailed derivation of laminate theories [14, 17, 21, 27, 31, 35] and formulation of FEA shape functions to satisfy certain imposed continuity constraints. However, through brief review of literature, it is evident that majority of the articles are scattered in the extremities of the spectrum, resulting in a scarcity of manuscripts that present the entire process starting from integrating laminate theories into the FEA weak form and ending with implementation and comparison of possible shape functions. This insufficiency is especially exacerbated for low order ESL models, as the limited amount of literature that actually illustrates the detailed FEA integration process tends to focus on high order and esoteric models. Hence, this research hopes to contribute in addressing the said scarcity by first demonstrating a comprehensive and rigorous process of integrating both the material model and simple shape functions into the CLT based weak form of the laminate equilibrium equations. Subsequently, the performance of the FEA suite will be assessed against commercial FEA software result, which is treated as the pseudo-ground truth, using archetypal problems of plate bending and cantilever bending. This procedure, alongside with rigid body modes analyses will act as verification process for the developed FEA, hence adding confidence to the correctness of the implement. Also, the comparison analyses will be able to shed light on the optimal selection of elements in different loading scenarios. Not only will these analyses lay the groundwork for a solid laminate analysis suite to support future optimization studies, but also it will benefit researchers who are in the realm of preliminary design or introductory composite material modeling.

Chapter 3

Mathematical Derivation of CLT-Based Laminate FEA

3.1 Laminate Material Theories

The laminate material model of interest is the CLT in the class of ESL theories. In this section, the main assumptions and the strain-displacement relationship for each theory will be discussed,

3.1.1 Classical Laminate Theory (CLT)

CLT: Main Assumptions and Lamina Stress-Strain Relationship

CLT is a composite plate theory developed based on Kirchhoff thin plate theory [9, 29, 32, 40]. The key kinematic assumptions enforced by the theory are as listed below:

1. Planes perpendicular to the mid-plane of the laminate will deform (rotate) in a specific manner that cause them to maintain their orthogonality with the mid-plane while also remaining straight. This is analogous to the Euler's assumption imposed for Euler-Bernoulli beams.

2. Lines in out-of-plane perpendicular direction, orthogonal to the mid-plane do not experience elongation or compression. This implies that the laminate does not experience a change of thickness during deformation.

3. Infinitesimal strains and displacements are presumed

4. The in-plane displacement field varies as a linear function with respect to the out-of-plane (the lamina stacking direction) coordinate.

The implication of these assumptions is that the out-of-plane normal stress and shear stress are neglected, thus rendering the laminate in a state of plane stress in addition to zero through-thickness strain. Hence, the lamina stress and strain constitutive model reduces to Equation 3.1 shown below. This approximation is appropriate when the laminate is sufficiently thin, as the accuracy of the assumption breaks down for thick laminates. Also note that the planes perpendicular to the out-of-plane direction will have the same transverse displacement field, and hence purely a function of the in-plane coordinates.

$$\begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \gamma_{12} \end{bmatrix} = \begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{21}}{E_2} & 0 \\ -\frac{\nu_{12}}{E_1} & \frac{1}{E_2} & 0 \\ 0 & 0 & \frac{1}{G_{12}} \end{bmatrix} \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix} = S \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix} \tag{3.1}$$

In Equation 3.1, S is the reduced lamina compliance matrix. Recall the assumption stated in section 2.1.1 that S is symmetrical. In addition, the inverse of S maps the strains to the stresses, and is called the lamina stiffness matrix, which will be denoted as K. Thus:

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix} = S^{-1} \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \gamma_{12} \end{bmatrix} = K \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \gamma_{12} \end{bmatrix} \tag{3.2}$$

CLT: Strain-Displacement Relationship

In the global laminate coordinates, denote the 1 direction as the x axis, the 2 direction as the y axis, and the 3 direction as the z axis, and set the mid-plane of the laminate to correspond to the x-y plane of the Cartesian coordinate. The other important implication from the assumptions is that the in-plane displacements induced from bending and twisting motions are directly associated with the transverse displacement field of the mid-plane. Specifically, angle of rotation of a material point in the laminate can be approximated by the in-plane derivatives of transverse displacement field, as demonstrated in Figure 3.1. This gives rise to the displacement field equations for any material point in the laminate, as shown in Equation

3.3.

$$
\begin{aligned}
u(x,y,z) &= u_0(x,y) - z\frac{\partial w_0}{\partial x} \\
v(x,y,z) &= v_0(x,y) - z\frac{\partial w_0}{\partial y} \\
w(x,y,z) &= w_0(x,y)
\end{aligned}
\tag{3.3}
$$

Note that u_0, v_0, and w_0 are the mid-plane displacement field corresponding to displacements in (x,y,z) axes respectively. The mid-plane corresponds to the x-y plane through the origin.

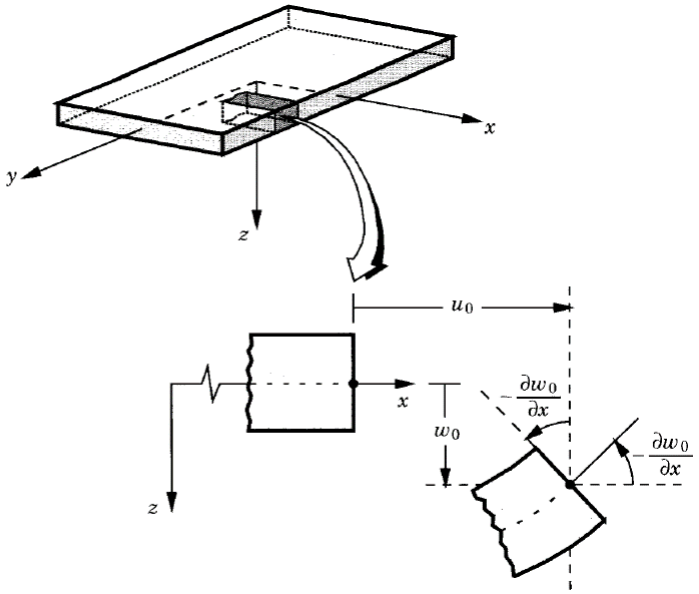

Figure 3.1: Rotation induced displacement in x axis direction.[29]

Applying the infinitesimal strain assumption, the strain-displacement relationship for CLT can be established and is shown in Equation 3.4. Note that only three independent terms u, v, and w are required to define the displacement field at any location, thus the total displacement functional degree of freedom for CLT is three.

$$
\begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} \frac{\partial u(x,y,z)}{\partial x} \\ \frac{\partial v(x,y,z)}{\partial y} \\ \frac{\partial u(x,y,z)}{\partial y} + \frac{\partial v(x,y,z)}{\partial x} \end{bmatrix} = \begin{bmatrix} \frac{\partial u_0}{\partial x} - z\frac{\partial^2 w_0}{\partial x^2} \\ \frac{\partial v_0}{\partial y} - z\frac{\partial^2 w_0}{\partial y^2} \\ \frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x} - 2z\frac{\partial^2 w_0}{\partial x \partial y} \end{bmatrix}
\tag{3.4}
$$

CLT: Coordinate Transformed Equations

For a general laminate constituting of multiple layers of lamina, the fiber orientations of the laminae need not to be the same, as demonstrated in Figure 2.3. Consequently, since only one set of global coordinates are defined when analyzing a laminate, but the stress-strain and strain-displacement relationship of each ply layer is defined based on the local coordinate with respect to the fiber orientation, thus a transformation is required to relate the local lamina equations to global laminate coordinates. Note that the ESL theories are 2D models, so the transformation matrices only need to consider rotation of the plane perpendicular to the stacking direction of plies, as the transverse axis is shared between local and global systems.

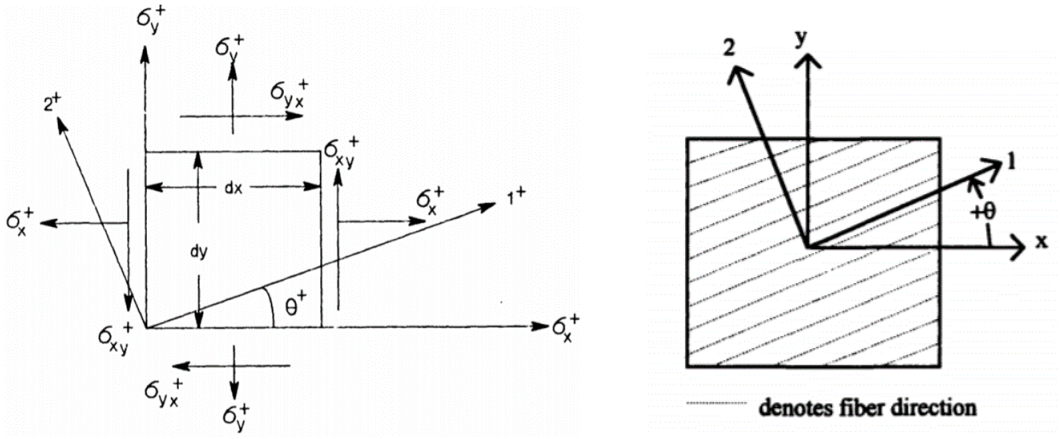

Figure 3.2: Lamina local stress state (1,2) transformed into local coordinate system (x,y) [21]

To clearly distinguish between the two coordinate systems, for rest of the section, define the local in-plane coordinate system axes as (1,2) and denote the global in-plate coordinate system axes as (x,y). Consider the stress state shown in Figure 3.2. The matrix formulation shown below can be used to transform the lamina stress state into components along global (x,y) coordinates:

$$\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = T^{-1} \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix} \tag{3.5}$$

Here, the T matrix is the transformation matrix formulated using the angle θ between the x-axis and the 1-axis:

$$T = \begin{bmatrix} \cos^2(\theta) & \sin^2(\theta) & 2\cos(\theta)\sin(\theta) \\ \sin^2(\theta) & \cos^2(\theta) & -2\cos(\theta)\sin(\theta) \\ -\cos(\theta)\sin(\theta) & \cos(\theta)\sin(\theta) & \cos^2(\theta) - \sin^2(\theta) \end{bmatrix} \tag{3.6}$$

The same transformation can also be used to convert strains in the principal fiber direction (1) to the strain at the global x-axis at an arbitrary angle θ with respect to the 1-axis. The only change required is the application of Reuter's Matrix (R), as defined below, which accounts for the usage of engineering shear strains.

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix} \tag{3.7}$$

So the strain transformation would take the form of:

$$\begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \gamma_{xy} \end{bmatrix} = RT^{-1}R^{-1} \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \gamma_{12} \end{bmatrix} \tag{3.8}$$

It is important to note that the elastic Hooke's law in Equation 3.1 and Equation 3.2 were derived for the angled lamina. Hence, the stress-strain relationship based on local lamina frame in Equation 3.1 need to be reformed into expressions in global (x,y) coordinate system:

$$\begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \gamma_{xy} \end{bmatrix} = RT^{-1}R^{-1} \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \gamma_{12} \end{bmatrix} = RT^{-1}R^{-1}S \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix} = RT^{-1}R^{-1}ST \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} \tag{3.9}$$

In the above formulation (Equation 3.9), the $RT^{-1}R^{-1}ST$ matrix is called the transformed compliance matrix, and will be denoted as $\bar{S}$. The inverse of the transformed compliance matrix will be denoted as $\bar{K} = \bar{S}^{-1}$, and it is called the transformed stiffness matrix. $\bar{K}$ can

be used to map strains in global coordinates to stresses in global coordinate for a angled lamina, and the equality can naturally be arrived by repeating the derivation shown in Equation 3.9 but instead for stress values based on Equation 3.2. The final result will be:

$$\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \bar{K} \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \gamma_{xy} \end{bmatrix} \tag{3.10}$$

CLT: Laminate Force-Strain Relationship

Although the strain and displacement field are derived to be continuous through the thickness of the laminate, however, the stress field is only continuous within the thickness of one lamina, and could vary from ply to ply, thus appearing discontinuous along the laminate thickness. This behaviour is due to the fact that the material constant in the compliance matrix can be different for each lamina, thus inducing jump discontinuities in the stress field between different lamina layers. This also implies that Equations 3.2 and Equation 3.1 are not applicable for a general laminate. Instead, it is more appropriate to relate strains/displacement field to generalized force distributions for the case of a laminate. By applying the assumption that the lamina layers in a laminate are perfectly bonded, then the stresses in each lamina will combine to produce equivalent laminate level axial loads per unit width (N) and bending moments per unit width (M). The expression of the two generalized forces can be derived through lamina-wise integration of the stresses. Assuming that a laminate is made from n layers of plies:

$$\begin{aligned} \begin{bmatrix} N_{xx} \\ N_{yy} \\ N_{xy} \end{bmatrix} &= \sum_{k=1}^{n} \int_{z_k}^{z_{k+1}} \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} dz = \sum_{k=1}^{n} \int_{z_k}^{z_{k+1}} \bar{K} \begin{bmatrix} \frac{\partial u_0}{\partial x} - z\frac{\partial^2 w_0}{\partial x^2} \\ \frac{\partial v_0}{\partial y} - z\frac{\partial^2 w_0}{\partial y^2} \\ \frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x} - 2z\frac{\partial^2 w_0}{\partial x \partial y} \end{bmatrix} dz \\ &= \sum_{k=1}^{n} \bar{K}(z_{k+1} - z_k) \begin{bmatrix} \frac{\partial u_0}{\partial x} \\ \frac{\partial v_0}{\partial y} \\ \frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x} \end{bmatrix} + \sum_{k=1}^{n} \frac{1}{2} \bar{K}(z_{k+1}^2 - z_k^2) \begin{bmatrix} -\frac{\partial^2 w_0}{\partial x^2} \\ -\frac{\partial^2 w_0}{\partial y^2} \\ -2\frac{\partial^2 w_0}{\partial x \partial y} \end{bmatrix} \\ &= A \begin{bmatrix} \frac{\partial u_0}{\partial x} \\ \frac{\partial v_0}{\partial y} \\ \frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x} \end{bmatrix} + B \begin{bmatrix} -\frac{\partial^2 w_0}{\partial x^2} \\ -\frac{\partial^2 w_0}{\partial y^2} \\ -2\frac{\partial^2 w_0}{\partial x \partial y} \end{bmatrix} \end{aligned} \tag{3.11}$$

Equation 3.11 relates the mid-plane displacement field u_0, v_0, and w_0, which are purely functions of x and y, with the laminate axial forces. The matrix $A = \sum_{k=1}^{n} \bar{K}(z_{k+1} - z_k)$ is known as the extensional stiffness matrix and the $B = \frac{1}{2}\sum_{k=1}^{n} \bar{K}(z_{k+1}^2 - z_k^2)$ matrix is known as the bending-extensional coupling stiffness matrix. On the other hand:

$$\begin{aligned}
\begin{bmatrix} M_{xx} \\ M_{yy} \\ M_{xy} \end{bmatrix} &= \sum_{k=1}^{n} \int_{z_k}^{z_{k+1}} z \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} dz = \sum_{k=1}^{n} \int_{z_k}^{z_{k+1}} z\bar{K} \begin{bmatrix} \frac{\partial u_0}{\partial x} - z\frac{\partial^2 w_0}{\partial x^2} \\ \frac{\partial v_0}{\partial y} - z\frac{\partial^2 w_0}{\partial y^2} \\ \frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x} - 2z\frac{\partial^2 w_0}{\partial x \partial y} \end{bmatrix} dz \qquad (3.12) \\
&= \sum_{k=1}^{n} \frac{1}{2}\bar{K}(z_{k+1}^2 - z_k^2) \begin{bmatrix} \frac{\partial u_0}{\partial x} \\ \frac{\partial v_0}{\partial y} \\ \frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x} \end{bmatrix} + \sum_{k=1}^{n} \frac{1}{3}\bar{K}(z_{k+1}^3 - z_k^3) \begin{bmatrix} -\frac{\partial^2 w_0}{\partial x^2} \\ -\frac{\partial^2 w_0}{\partial y^2} \\ -2\frac{\partial^2 w_0}{\partial x \partial y} \end{bmatrix} \\
&= B \begin{bmatrix} \frac{\partial u_0}{\partial x} \\ \frac{\partial v_0}{\partial y} \\ \frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x} \end{bmatrix} + D \begin{bmatrix} -\frac{\partial^2 w_0}{\partial x^2} \\ -\frac{\partial^2 w_0}{\partial y^2} \\ -2\frac{\partial^2 w_0}{\partial x \partial y} \end{bmatrix}
\end{aligned}$$

Equation 3.12 relates the mid-plane displacement field to the laminate moment. The additional matrix $D = \frac{1}{3}\sum_{k=1}^{n} \bar{K}(z_{k+1}^3 - z_k^3)$ used is known as the bending stiffness matrix. Finally, note that since the transformed stiffness matrix $\bar{K}$ is symmetrical, as a result, A, B, and D matrices are all symmetrical as well.

3.2 Derivation of Strong and Weak Form Equations for CLT Based Fiber Laminate FEA

In the previous section, the strain-displacement and stress-strain relationships were derived for different laminate material model. These equations can now be incorporated to derive the both the strong form of the governing equation and the weak form for the FEA model.

3.2.1 Strong Form Derivation Using Static Principle of Virtual Work

Since the preliminary stage of this project will based on simple static archetypal problems, the strong form derivation will be done using the energy principle for static structures, specifically the theorem of minimum total potential energy [29, 43].

The theorem of minimum total potential energy is a special case of the principle of virtual displacements. For a general structure in static equilibrium, the total potential energy (L)

for the structure can be written as the sum of the total elastic strain energy stored in the structure (U) and the total potential energy associated with the external forces (V), i.e.:

$$L = U + V \tag{3.13}$$

Also note that in deriving the lamina constitutive model, one of the assumptions made was the existence of strain energy density ($U^{'}$) of the form $U^{'} = \frac{1}{2}\sigma_{ij}\epsilon_{ij}$. Thus, the total strain energy (U) can be found by integrating the strain energy density over the domain of the structure. For an elastic body, the theorem of minimum total potential energy states that the set of admissible displacements which satisfy the static equilibrium condition also minimizes the total potential energy. Using variational notations, this statement can be written as:

$$\delta L = \delta U + \delta V = 0 \tag{3.14}$$

The result of minimum potential energy will lead to the static equilibrium equations of the laminate, hence the relationships between generalized forces (surface forces and boundary forces) and generalized displacements.

Recall that for CLT, the reduced form of lamina stress-strain model was derived in Equation 3.1. The lamina equation can be used to establish the total strain energy of the laminate. Define the volume domain of the laminate to be Ω_0 such that:

$$\Omega_0 = \Omega \cup \ \{z : -h \leq z \leq h\}$$

where the in-plane domain is given as:

$$\Omega = \{x, y : a \leq x \leq b; c \leq y \leq d\}$$

The [a,b] and [c,d] here defines the arbitrary boundary of the laminate, and the [-h,h] here defines the arbitrary thickness of the laminate. Within the domain, the total strain energy (U) can be written as:

$$U = \frac{1}{2}\int_{\Omega_0} (\sigma_{xx}\epsilon_{xx} + \sigma_{yy}\epsilon_{yy} + \sigma_{xy}\gamma_{xy})\, d\Omega_0 \tag{3.15}$$

Substituting in the strain-displacement relationship:

$$U = \frac{1}{2}\int_{-h}^{h}\int_{\Omega_0} (\sigma_{xx}(\frac{\partial u_0}{\partial x} - z\frac{\partial^2 w_0}{\partial x^2}) + \sigma_{yy}(\frac{\partial v_0}{\partial y} - z\frac{\partial^2 w_0}{\partial y^2}) + \sigma_{xy}(\frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x} - 2z\frac{\partial^2 w_0}{\partial x \partial x})\, d\Omega_0\, dz \tag{3.16}$$

The mid-plane displacement field u_0, v_0, and w_0 are functions independent of the transverse

coordinate z. As a result, the integration over z can be done independently on the stress terms (σ_{ij} or $\sigma_{ij}z$). Applying Equation 3.12 and Equation 3.11 and expanding the matrix product term by term, then Equation 3.16 becomes:

$$
\begin{aligned}
U = \frac{1}{2}\int_{\Omega_0} [A_{11}\frac{\partial u_0}{\partial x}^2 - 2B_{11}\frac{\partial u_0}{\partial x}\frac{\partial^2 w_0}{\partial x^2}^2 + D_{11}\frac{\partial^2 w_0}{\partial x^2}^2 + A_{22}\frac{\partial v_0}{\partial y}^2 - 2B_{22}\frac{\partial v_0}{\partial y}\frac{\partial^2 w_0}{\partial y^2}^2 \\
+D_{22}\frac{\partial^2 w_0}{\partial y^2}^2 + A_{33}(\frac{\partial u_0}{\partial y}\frac{\partial v_0}{\partial x})^2 - 4B_{33}(\frac{\partial u_0}{\partial y}\frac{\partial v_0}{\partial x})\frac{\partial^2 w_0}{\partial y \partial x} + 2D_{33}\frac{\partial^2 w_0}{\partial y \partial x}^2 \\
+2A_{12}\frac{\partial u_0}{\partial x}\frac{\partial v_0}{\partial y} - B_{12}(\frac{\partial u_0}{\partial x}\frac{\partial^2 w_0}{\partial x^2} + \frac{\partial v_0}{\partial y})\frac{\partial^2 w_0}{\partial y^2} + D_{12}\frac{\partial^2 w_0}{\partial x^2}\frac{\partial^2 w_0}{\partial y^2} \\
+2A_{13}\frac{\partial u_0}{\partial x}(\frac{\partial u_0}{\partial y}\frac{\partial v_0}{\partial x}) - 2B_{13}(2\frac{\partial u_0}{\partial x}\frac{\partial^2 w_0}{\partial x \partial y} + \frac{\partial^2 w_0}{\partial x^2}(\frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x})) \\
+4D_{13}\frac{\partial^2 w_0}{\partial x^2}\frac{\partial^2 w_0}{\partial x \partial y} + 2A_{23}\frac{\partial v_0}{\partial y}(\frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x}) - 2B_{23}(2\frac{\partial v_0}{\partial y}\frac{\partial^2 w_0}{\partial x \partial y} + \frac{\partial^2 w_0}{\partial y^2}(\frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x})) \\
+4D_{23}\frac{\partial^2 w_0}{\partial y^2}\frac{\partial^2 w_0}{\partial x \partial y}]\ d\Omega_0
\end{aligned}
\tag{3.17}
$$

Note that (A_{ij},B_{ij},D_{ij}) corresponds to the ith row and jth column entry of the (A, B, D) matrices (introduced in section 3.1.1) respectively. Using the full expression of the strain energy, the variation can be taken with respect to the functional displacement degree of freedoms (u_0, v_0, w_0):

$$
\begin{aligned}
\delta U = \int_{\Omega_0} \{[A_{11}\frac{\partial u_0}{\partial x} + A_{12}\frac{\partial v_0}{\partial y} + A_{13}(\frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x}) - B_{11}\frac{\partial^2 w_0}{\partial x^2} - B_{12}\frac{\partial^2 w_0}{\partial y^2} - 2B_{13}\frac{\partial^2 w_0}{\partial x \partial y}]\frac{\partial \delta u_0}{\partial x} \\
+[A_{12}\frac{\partial u_0}{\partial x} + A_{22}\frac{\partial v_0}{\partial y} + A_{23}(\frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x}) - B_{12}\frac{\partial^2 w_0}{\partial x^2} - B_{22}\frac{\partial^2 w_0}{\partial y^2} - 2B_{23}\frac{\partial^2 w_0}{\partial x \partial y}]\frac{\partial \delta v_0}{\partial y} \\
+[A_{13}\frac{\partial u_0}{\partial x} + A_{23}\frac{\partial v_0}{\partial y} + A_{33}(\frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x}) - B_{13}\frac{\partial^2 w_0}{\partial x^2} - B_{23}\frac{\partial^2 w_0}{\partial y^2} - 2B_{33}\frac{\partial^2 w_0}{\partial x \partial y}](\frac{\partial \delta v_0}{\partial x} + \frac{\partial \delta u_0}{\partial y}) \\
+[-B_{11}\frac{\partial u_0}{\partial x} - B_{12}\frac{\partial v_0}{\partial y} - B_{13}(\frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x}) + D_{11}\frac{\partial^2 w_0}{\partial x^2} + D_{12}\frac{\partial^2 w_0}{\partial y^2} + 2D_{13}\frac{\partial^2 w_0}{\partial x \partial y}]\frac{\partial^2 \delta w_0}{\partial x^2} \\
+[-B_{12}\frac{\partial u_0}{\partial x} - B_{22}\frac{\partial v_0}{\partial y} - B_{23}(\frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x}) + D_{12}\frac{\partial^2 w_0}{\partial x^2} + D_{22}\frac{\partial^2 w_0}{\partial y^2} + 2D_{23}\frac{\partial^2 w_0}{\partial x \partial y}]\frac{\partial^2 \delta w_0}{\partial y^2} \\
+[-B_{13}\frac{\partial u_0}{\partial x} - B_{23}\frac{\partial v_0}{\partial y} - B_{33}(\frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x}) + D_{13}\frac{\partial^2 w_0}{\partial x^2} + D_{23}\frac{\partial^2 w_0}{\partial y^2} + 2D_{33}\frac{\partial^2 w_0}{\partial x \partial y}](2\frac{\partial^2 \delta w_0}{\partial y \partial x})\}\ d\Omega_0
\end{aligned}
\tag{3.18}
$$

Equation 3.18 can be simplified in terms of notations by considering Equation 3.12 and Equation 3.11, which provide the relationships between the axial load (N) as well as the (M)

moment of a laminate and the product of A, B, D matrices with the appropriate derivatives of mid-plane displacement field. As a result, Equation 3.18 can be alternatively expressed as:

$$\delta U = \int_{\Omega_0} (N_{xx}\frac{\partial \delta u_0}{\partial x} + N_{xy}\frac{\partial \delta u_0}{\partial y} + N_{yy}\frac{\partial \delta v_0}{\partial y} + N_{xy}\frac{\partial \delta v_0}{\partial x} - M_{xx}\frac{\partial^2 \delta w_0}{\partial x^2} - M_{yy}\frac{\partial^2 \delta w_0}{\partial y^2} - 2M_{xy}\frac{\partial^2 \delta w_0}{\partial x \partial y})\ d\Omega_0 \tag{3.19}$$

In order to present δU in a form such that integration into the theorem of minimum energy will immediately produce the static equilibrium equations, the derivative on the virtual displacements variations $(\delta u_0, \delta v_0, \delta w_0)$ would need to be reduced. This will be done by applying integration by parts alongside the Green's Theorem on Equation 3.19. Green's Theorem relates a line integral along an oriented simple closed curve (C) to an area integral with in the area bounded by the simple closed curve (Ω_0). Green's Theorem is applicable here if the stacking direction of the plies is defined as the positive z-axis direction, and each ply of the laminate is assumed to have the same in-plane dimensions with a continuous boundary. In that case, the simple closed curve refers to the boundary of laminate mid-plane in the xy-plane. The formula for the Green's Theorem is given in Equation 3.20 below:

$$\int_C (P\hat{e_y} - Q\hat{e_x}) \cdot \hat{e_s} ds = \int_{\Omega_0} \frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y} d\Omega_0 \tag{3.20}$$

Note that in Equation 3.20, the $\hat{e_x}$ and $\hat{e_y}$ represent the unit vector in x-axis direction and y-axis direction respectively, and the $\hat{e_s}$ corresponds to the unit vector in the tangential direction with respect to the simple closed curve (C). The Green's Theorem can generally be used during integration by parts using the following procedure:

$$\int_{\Omega_0} (F_1\frac{\partial \delta u}{\partial x} + F_2\frac{\partial \delta u}{\partial y}) d\Omega_0 = -\int_{\Omega_0} (\frac{\partial F_1}{\partial x} + \frac{\partial F_2}{\partial y})\delta u d\Omega_0 + \int_{\Omega_0} (\frac{\partial}{\partial x}(F_1 \delta u) + \frac{\partial}{\partial y}(F_2 \delta u) d\Omega_0 \tag{3.21}$$

Applying the Green's Theorem:

$$\int_{\Omega_0} (\frac{\partial}{\partial x}(F_1 \delta u) + \frac{\partial}{\partial y}(F_2 \delta u) d\Omega_0 = \int_C (F_1 \delta u \hat{e_y} - F_2 \delta u \hat{e_x}) \cdot \hat{e_s} ds$$

Using the above procedure, Equation 3.19 becomes:

$$\delta U = -(\int_{\Omega_0} \{(\frac{\partial N_{xx}}{\partial x} + \frac{\partial N_{xy}}{\partial y})\delta u_0 + (\frac{\partial N_{yy}}{\partial y} + \frac{\partial N_{xy}}{\partial x})\delta v_0 \tag{3.22}$$
$$+(\frac{\partial^2 M_{xx}}{\partial x^2} + \frac{\partial^2 M_{yy}}{\partial y^2} + 2\frac{\partial^2 M_{xy}}{\partial x \partial y})\delta w_0\}\, d\Omega_0)$$
$$+ \int_C \{(N_{xx}\delta u_0 - M_{xx}\frac{\partial \delta w_0}{\partial x} + \frac{\partial M_{xx}}{\partial x}\delta w_0 - M_{xy}\frac{\partial \delta w_0}{\partial y} + \frac{\partial M_{xy}}{\partial y}\delta w_0 + N_{xy}\delta v_0)\hat{e_y}$$
$$-(N_{yy}\delta v_0 + M_{yy}\frac{\partial \delta w_0}{\partial y} - \frac{\partial M_{yy}}{\partial y}\delta w_0 + M_{xy}\frac{\partial \delta w_0}{\partial x} - \frac{\partial M_{xy}}{\partial x}\delta w_0 + N_{xy}\delta u_0)\hat{e_x}\} \cdot \hat{e_s} ds$$

Before formulating the equation for the total potential energy associated with the external forces (V), it is important to recognize all the relevant surface forces and boundary forces applied to a laminate. The illustration of the forces are shown in Figure 3.3 and Figure 3.4. In the case of surface force, the only applicable term is the area distributed transverse load/pressure (q), which has a unit of $\frac{N}{m^2}$, and it is in general a function of (x,y). On the other hand, the set laminate boundary forces consist of in-plane tensile and shear forces (N_{nn}, N_{ns}) with units of $\frac{N}{m}$, in-plane bending and twisting moments (M_{nn}, M_{ns}) with units of $\frac{Nm}{m}$, and also transverse shear force (Q_n) with a unit of $\frac{N}{m}$. For a laminate with an arbitrary in-plane boundary geometry, these forces act either along the tangential direction of the boundary (s-direction) or the normal direction of the boundary (n-direction). For instance in Figure 3.4, when the tangential direction of the boundary is aligned with the y-axis, then the N_{ns} force becomes N_{xy} and N_{nn} force becomes N_{xx} [39]. The boundary forces can also be defined using their stress constituents through a lamina-wise integration:

$$\begin{bmatrix} N_{nn} \\ N_{ns} \end{bmatrix} = \int_{-h}^{h} \begin{bmatrix} \sigma_{nn} \\ \sigma_{ns} \end{bmatrix} dz \qquad \begin{bmatrix} M_{nn} \\ M_{ns} \end{bmatrix} = \int_{-h}^{h} z \begin{bmatrix} \sigma_{nn} \\ \sigma_{ns} \end{bmatrix} dz \qquad Q_n = \int_{-h}^{h} \sigma_{nz} dz \tag{3.23}$$

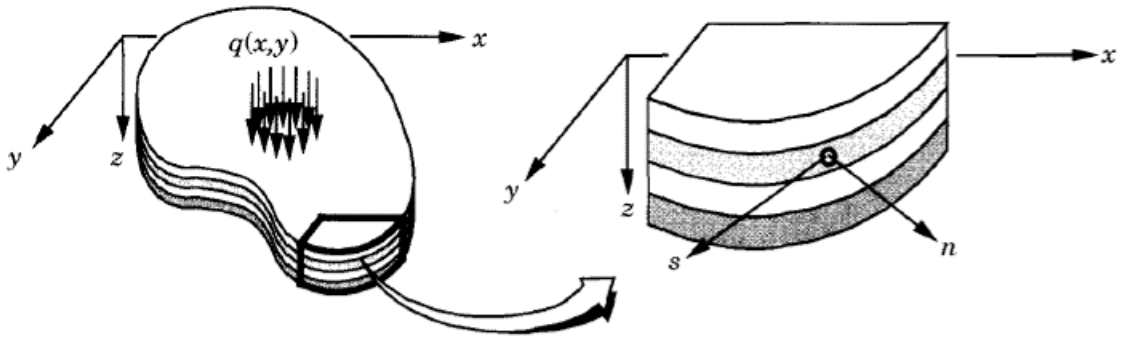

Figure 3.3: **left**: Illustration of the laminate surface force - area distributed transverse load. **right**: Boundary coordinate vectors (s,n) in comparison to the global laminate coordinate system (x,y). [29]

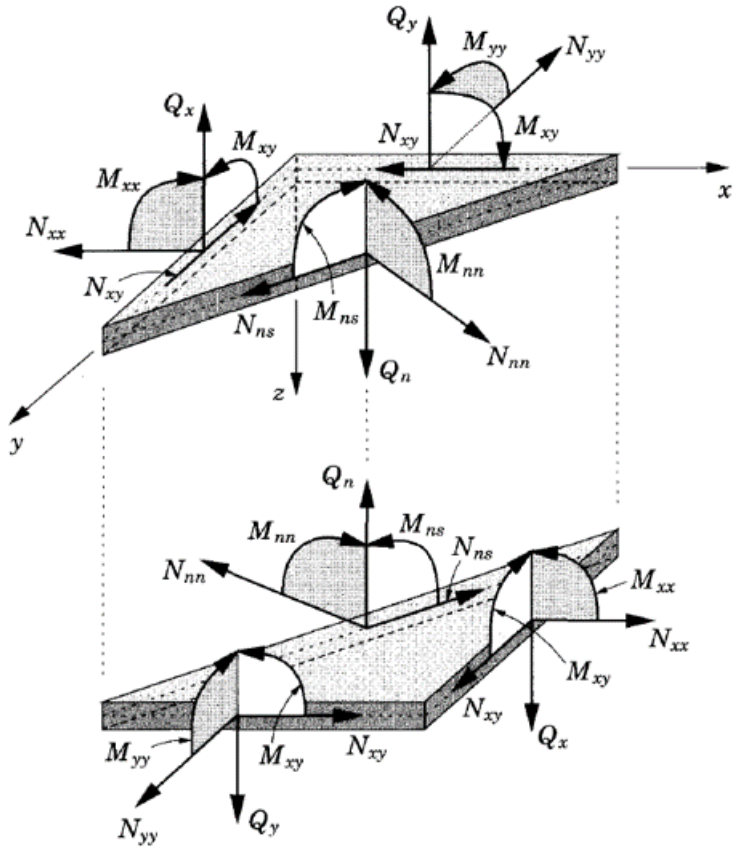

Figure 3.4: Boundary forces applicable to a laminate, shown with respect to global coordinate system (x,y) and also with respect to an arbitrary boundary surface's local coordinates (s,n).[29]

Due to the arbitrariness of the laminate boundary geometry, it is necessary to relate the local boundary coordinate system (n and s directions) to the global laminate coordinate

system (x and y axes). Let the positive z direction correspond to the stacking direction of the laminate. Assume that the in-plane boundary of the laminate is a simple closed curve (C). The "simple" part means that the boundary does not cross itself while the "closed" part means that the boundary encloses a finite area. Also define the positive orientation of the curve as the counterclockwise traversal direction when looking down at the curve from the positive z-axis (in other words, while traversing along the positive orientation of the curve, the enclosed finite area will always be on the left hand side). Then, the tangential coordinate (s) at a boundary point refers to the direction tangent to the positive orientation of the curve, and is equipped with the unit vector $\hat{e_s}$. The normal coordinate (n) at a boundary point is orthogonal to the tangential direction while also pointing away from the enclosed region, and is equipped with the unit vector $\hat{e_n}$. An illustration of the two directions defined is shown in Figure 3.3. Hence, if the normal direction at a boundary is at an angle θ counterclockwise from the positive x-axis direction, then the in-plane transformation between the global (x,y) system and boundary (n,s) system is given by:

$$\begin{aligned} \hat{e_x} &= \cos\theta \hat{e_n} - \sin\theta \hat{e_s} \\ \hat{e_y} &= \sin\theta \hat{e_n} + \cos\theta \hat{e_s} \end{aligned} \tag{3.24}$$

Equivalently, the inverse transformation is given by:

$$\begin{aligned} \hat{e_n} &= \cos\theta \hat{e_x} + \sin\theta \hat{e_y} \\ \hat{e_s} &= -\sin\theta \hat{e_x} + \cos\theta \hat{e_y} \end{aligned} \tag{3.25}$$

Note that the transformation given in Equation 3.24 and Equation 3.25 can be used to transform any variables from the (x,y) coordinates to the (n,s) coordinates, and vice versa. For example, they can employed to calculate the dot product between different unit vectors present in the line integral in Equation 3.22, i.e.:$\hat{e_x} \cdot \hat{e_s} = -\sin\theta$ and $\hat{e_y} \cdot \hat{e_s} = -\cos\theta$. At this point, the equation for the total potential energy associated with the external forces (V) can be formulated incorporating the surface force and boundary forces. To get the work associated with boundary forces or surface distributed forces, they would need to be multiplied with the corresponding displacements, and then integrated over the boundary or area. While for boundary moments, they would need to be multiplied with rotations, which for CLT would be derivatives of virtual transverse displacements, and integrated over the boundary. Hence:

$$V = \int_{\Omega_0} -qw_0 \, d\Omega_0 + \int_C (\bar{N_{nn}} u_{0,n} + \bar{N_{ns}} u_{0,s} - \bar{M_{nn}} \frac{\partial w_0}{\partial n} - \bar{M_{ns}} \frac{\partial w_0}{\partial s} + \bar{Q_n} w_0) \, ds \tag{3.26}$$

Note that the bar on top of the generalized force terms is to distinguish them clearly as external applied boundary forces. In the above equation, they are multiplied with displacements or rotations corresponding to the tangential (e.x. $u_{0,s}$) or normal direction (e.x. $u_{0,n}$) of the boundary. In order to transform those displacements into components (e.x. u_0 and v_0)along the global (x,y) orientation, Equation 3.25 can be applied. Subsequently, the first variation of V with respect to the displacement degree of freedoms (u_0, v_0, w_0) can be taken:

$$\delta V = \int_{\Omega_0} -q\delta w_0 \, d\Omega_0 + \int_C \{\bar{N_{nn}}(\cos\theta^{-1}\delta u_0 + \sin\theta^{-1}\delta v_0) + \bar{N_{ns}}(-\sin\theta^{-1}\delta u_0 + \cos\theta^{-1}\delta v_0) \tag{3.27}$$
$$-\bar{M_{nn}}(\frac{\partial \delta w_0}{\partial x}\cos\theta^{-1} + \frac{\partial \delta w_0}{\partial y}\sin\theta^{-1}) + \bar{M_{ns}}(\frac{\partial \delta w_0}{\partial x}\sin\theta^{-1} + \frac{\partial \delta w_0}{\partial y}\cos\theta^{-1}) + \bar{Q_n}\delta w_0\} \, ds$$

The expressions for δU in Equation 3.22 and δV in Equation 3.27, can now be gathered into the principle of minimum potential energy shown in Equation 3.14:

$$\begin{aligned}
\delta U + \delta V = -\langle \int_{\Omega_0} \{&(\frac{\partial N_{xx}}{\partial x} + \frac{\partial N_{xy}}{\partial y})\delta u_0 + (\frac{\partial N_{yy}}{\partial y} + \frac{\partial N_{xy}}{\partial x})\delta v_0 \\
&+(\frac{\partial^2 M_{xx}}{\partial x^2} + \frac{\partial^2 M_{yy}}{\partial y^2} + 2\frac{\partial^2 M_{xy}}{\partial x \partial y} + q)\delta w_0\} \, d\Omega_0)\rangle \\
+\int_C \{&(\bar{N_{nn}}\cos\theta^{-1} - N_{xx}\cos\theta + N_{xy}\sin\theta - \bar{N_{ns}}\sin\theta^{-1})\delta u_0 \\
&+(-N_{xy}\cos\theta + N_{yy}\sin\theta + \bar{N_{nn}}\sin\theta^{-1} + \bar{N_{ns}}\cos\theta^{-1})\delta v_0 \\
+(-\frac{\partial M_{xx}}{\partial x}\cos(\theta) - &\frac{\partial M_{xy}}{\partial y}\cos(\theta) - \frac{\partial M_{yy}}{\partial y}\sin\theta - \frac{\partial M_{xy}}{\partial x}\sin\theta + \bar{Q_n})\delta w_0 \\
&+(M_{xx}\cos\theta + M_{xy}\sin\theta - \bar{M_{nn}}\cos\theta^{-1} + \bar{M_{ns}}\sin\theta^{-1})\frac{\partial \delta w_0}{\partial x} \\
+(M_{xy}\cos\theta + M_{yy}\sin\theta - &\bar{M_{nn}}\sin\theta^{-1}) + \bar{M_{ns}}\cos\theta^{-1})\frac{\partial \delta w_0}{\partial y}\} \, ds
\end{aligned} \tag{3.28}$$

In order for $\delta U + \delta V$ to be equivalent to zero (minimum potential energy), the terms inside the domain area integral and boundary line integral must be functionally equivalent to the zero function. In addition, since the expressions in Equation 3.28 are factored with respect to the displacement variations, thus the coefficients of all the variations must be zero as well. Thus, from the domain area integral, three equivalencies are identified for within the domain

Ω_0:

$$\begin{aligned}
\delta u_0: &\quad \frac{\partial N_{xx}}{\partial x} + \frac{\partial N_{xy}}{\partial y} = 0 \\
\delta v_0: &\quad \frac{\partial N_{yy}}{\partial y} + \frac{\partial N_{xy}}{\partial x} = 0 \\
\delta w_0: &\quad \frac{\partial^2 M_{xx}}{\partial x^2} + \frac{\partial^2 M_{yy}}{\partial y^2} + 2\frac{\partial^2 M_{xy}}{\partial x \partial y} + q = 0
\end{aligned} \tag{3.29}$$

This system of equations (Equation 3.29) constitutes the strong form of static equilibrium equations for the laminate structure. From the boundary line integral, three matrix equations can be formulated:

$$\delta u_0 \; and \; \delta v_0: \quad \begin{bmatrix} \bar{N_{nn}} \\ \bar{N_{ns}} \end{bmatrix} = \begin{bmatrix} \cos^2_\theta & \sin^2_\theta & 2\cos_\theta \sin_\theta \\ -\cos_\theta \sin_\theta & \cos_\theta \sin_\theta & \cos^2_\theta - \sin^2_\theta \end{bmatrix} \begin{bmatrix} N_{xx} \\ N_{yy} \\ N_{xy} \end{bmatrix} \tag{3.30}$$

$$\frac{\partial \delta w_0}{\partial x} \; and \; \frac{\partial \delta w_0}{\partial y}: \quad \begin{bmatrix} \bar{M_{nn}} \\ \bar{M_{ns}} \end{bmatrix} = \begin{bmatrix} \cos^2_\theta & \sin^2_\theta & 2\cos_\theta \sin_\theta \\ -\cos_\theta \sin_\theta & \cos_\theta \sin_\theta & \cos^2_\theta - \sin^2_\theta \end{bmatrix} \begin{bmatrix} M_{xx} \\ M_{yy} \\ M_{xy} \end{bmatrix}$$

$$\delta w_0: \quad -\frac{\partial M_{xx}}{\partial x}\cos(\theta) - \frac{\partial M_{xy}}{\partial y}\cos(\theta) - \frac{\partial M_{yy}}{\partial y}\sin\theta - \frac{\partial M_{xy}}{\partial x}\sin\theta + \bar{Q_n} = 0$$

The matrix equality shown in Equation 3.30 are the admissible natural/Neumann boundary conditions that the laminate must satisfy. Equations 3.29 and 3.30 fully defines the strong form of the static laminate problem that the FEA will be attempting to analyze.

Recall in Equation 3.11 and Equation 3.12, the N and M terms are expressed as functions of spacial derivatives of the displacement field variables u_0, v_0, and w_0. Hence, the solution of the strong form equation would be a set of continuous functions over the problem domain, representing these displacement field variables. The strong form imposes additional spacial derivatives to the displacement fields. Hence, the regularity conditions related to strong form require the u_0 and v_0 functional solutions to be continuous and C^3 differentiable with respect to x and y in the domain, while the the w_0 functional solution needs to be C^4 differentiable in the domain. Additionally, they also need to satisfy the problem dependent Dirichlet/essential boundary conditions.

3.2.2 Weak Form Derivation Using Galerkin Method

Following the strong form equations, the Galerkin method was employed to derive the weak form equations for static equilibrium. The Galerkin method is a type of Weighted Residual

Method, which itself is a class of approximative methods for solving differential equations [6, 22]. The idea of approximation methods is that instead of finding the exact and continuous functional solution required for the strong form differential equation, try finding an estimate of the solution by satisfying some form of equality constraints over certain regions of the problem domain. Specifically for the case of Galerkin method, weight functions are multiplied to the strong form equation and boundary force conditions [6, 22]. The imposed constraint is that the approximated solution should result in the integral difference between the strong form and the applied boundary forces to be zero over both the domain and the boundary. In other words, Galerkin method restructures the equilibrium equations to one where the functional L^2 norm of the weight function and difference function, calculated as the difference between strong form equations and boundary forces, needs to be zero over the problem domain and the boundary. The weight functions hence must belong to the L^2 functional space defined with respect to the domain of the problem. Note that L^2 functional space means that the functions must be square integrable over the problem domain.

The advantage of the Galerkin method is that the highest order of differentiation is reduced compared to the strong form problem, and the equilibrium conditions are not as constringent since they are satisfied in an integral weighted error sense. Both factors also lead to reduction of the continuity requirements for the displacement variables, which will be demonstrated later on.

Since the strong form presented in Equation 3.29 is a system of equations, thus to apply the Galerkin method, denote three weight functions that belong to the L^2 functional space: δu_0, δv_0, and δw_0. Multiply each weight function to the respective strong form equation and integrate over the problem domain to obtain:

$$\begin{aligned}
\int_{\Omega_0} \delta u_0 \left[\frac{\partial N_{xx}}{\partial x} + \frac{\partial N_{xy}}{\partial y} \right] d\Omega_0 = 0 \\
\int_{\Omega_0} \delta v_0 \left[\frac{\partial N_{yy}}{\partial y} + \frac{\partial N_{xy}}{\partial x} \right] d\Omega_0 = 0 \\
\int_{\Omega_0} \delta w_0 \left[\frac{\partial^2 M_{xx}}{\partial x^2} + \frac{\partial^2 M_{yy}}{\partial y^2} + 2\frac{\partial^2 M_{xy}}{\partial x \partial y} + q \right] d\Omega_0 = 0
\end{aligned} \tag{3.31}$$

The equation set in 3.31 is the Weighted Residual Form (WRF) of the strong form obtained from Galerkin method. Note that since the choice of the weight functions can be arbitrary, Equation 3.31 must be satisfied for all weight functions belonging to the L^2 functional space over the domain. Subsequently, integration by part, using Green's Theorem (Equation 3.20) and Equation 3.25, can be applied to the WRF to reduce the order of spatial

derivatives and relate the equations to boundary forces:

$$\int_{\Omega_0} \left[\frac{\partial \delta u_0}{\partial x} N_{xx} + \frac{\partial \delta u_0}{\partial y} N_{xy} \right] d\Omega_0 = \int_C \delta u_0 \left(\bar{N_{xx}} cos(\theta) + \bar{N_{xy}} sin(\theta) \right) dS \tag{3.32}$$

$$\int_{\Omega_0} \left[\frac{\partial \delta v_0}{\partial y} N_{yy} + \frac{\partial \delta v_0}{\partial x} N_{xy} \right] d\Omega_0 = \int_C \delta v_0 \left(\bar{N_{yy}} sin(\theta) + \bar{N_{xy}} cos(\theta) \right) dS$$

$$\int_{\Omega_0} \left[\frac{\partial^2 \delta w_0}{\partial x^2} M_{xx} + \frac{\partial^2 \delta w_0}{\partial y^2} M_{yy} + \frac{\partial^2 \delta w_0}{\partial x \partial y} M_{xy} \right] d\Omega_0 = \int_{\Omega_0} -\delta w_0 q d\Omega_0$$

$$+ \int_C \left\{ - \left[\frac{\partial \delta w_0}{\partial x} (\bar{M_{xx}} cos(\theta) + \bar{M_{xy}} sin(\theta)) + \frac{\partial \delta w_0}{\partial y} (\bar{M_{yy}} sin(\theta) + \bar{M_{xy}} cos(\theta)) \right] + \delta w_0 \bar{Q_n} \right\} dS$$

Equation 3.32 is the weak form of the static laminate equilibrium equations. The terms with a bar on top are the boundary forces applied in the problem. As mentioned, the reduction of spatial derivatives in weak form is due to both integration by parts and also the application of integration. Thus, admissible displacement solutions and the weight functions only need to ensure that the integrals exist, in addition to satisfying the essential boundary conditions. In mathematical terms, this can be captured by defining special Sobolev spaces over the domain:

$$H^1(\Omega_0) \equiv \{ w : (w, \frac{\partial w}{\partial x}, \frac{\partial w}{\partial y}) \in L^2(\Omega_0) \} \tag{3.33}$$
$$H^1_E(\Omega_0) \equiv \{ w : w \in H^1(\Omega_0), \bar{w}(C_E) \}$$

Also:

$$H^2(\Omega_0) \equiv \{ w : (w, \frac{\partial w}{\partial x}, \frac{\partial w}{\partial y}, \frac{\partial^2 w}{\partial x^2}, \frac{\partial^2 w}{\partial y^2}, \frac{\partial^2 w}{\partial x \partial y},) \in L^2(\Omega_0) \} \tag{3.34}$$
$$H^2_E(\Omega_0) \equiv \{ w : w \in H^2(\Omega_0), w(C_E) = \bar{w}(C_E) \}$$

Essentially, the Sobolev spaces defined in Equation 3.33 and Equation 3.34 say that for a function belonging to this special functional space, itself and its appropriate spatial derivatives need to be square integrable.. In addition, the function needs to satisfy the imposed Dirichlet/essential boundary conditions ($\bar{w}$) imposed over the essential boundary portion

(C_E). Hence, the weak form suggests that the in-plane displacement solutions (u_0, v_0) and their corresponding weight functions (δu_0, δv_0) belong to the H^1_E functional space, which implies that they need to be C^0 continuous over the domain. The transverse displacement solution (w_0) and the weight function (δw_0) belong to the H^2_E functional space and thus, need to be C^1 continuous over the domain. These regularity conditions are much more relaxed compared to the ones demanded by the strong form equations.

Chapter 4

FEA Implementation

4.1 Domain Discretization and Element Integration

A key component of FEA is discretizing the large problem domain into small element domains and solving the problem using numerical methods. This procedure allows the weak equation to be applied locally, which enables discrete numerical approximations with reasonable accuracy. Instead of solving for a functional as required by weak form, the discrete approximation converts the weak form problem into a matrix product problem, and the goal is to solve for specific degrees of freedom at grid points of interest.

The discretization process, or better known as meshing, is directly related to the choice of element used in FEA implementation. The usage of an element is essentially a way to approximate the displacement field in the meshed domain. As mentioned in section 2.2.1, the conventional quadrilateral thin-shell element will be considered for this study. This thin-shell element is rectangular in shape with four nodes and four edges.

Element application typically consists of three criteria: identifying the applicable variables in the weak form, selecting the interpolation functions needed for approximation based on continuity requirements, as well as the knowing the type and number of nodal degrees of freedom needed to uniquely define the interpolation scheme. The shell element is a type of hybrid element, as the approximation scheme for the in-plane displacement variables (u_0, v_0) is different compared to that of the out-of-plane displacement variable (w_0). The in-plane variables are approximated by the C^0 continuous plane stress membrane element. Two different C^1 continuous elements will be considered for the transverse component: the conforming element and nonconforming element.

4.1.1 Plane Stress Element

The plane stress element belongs to a general group of elements called the isoparametric elements. The plane stress element is applicable to the u_0 and v_0 variables and requires four degrees of freedom (d.o.f.), specifically the displacement values at the four corner nodes, to uniquely define its spatial approximation scheme. The interpolation (shape) function used to approximate the two-dimensional displacement field using the nodal d.o.f. is constructed using polynomial terms, and is given as:

$$\Phi(x, y) = a + bx + cy + dxy \tag{4.1}$$

Note there will be four different $\Phi(x, y)$ functions, each with a unique set of (a, b, c, d) constants, and when multiplied with the four nodal d.o.f. will generate the element-wise displacement approximation scheme. Those constants are obtained by requiring the product between the interpolation functions and the nodal d.o.f. to exactly reproduce the nodal d.o.f. value at the four nodes. In addition, notice how the shape functions are linear with respect to both x and y individually. For this reason, the plane stress element also belongs to the bilinear quadrilateral element family. Note that typically in literature, the full set of shape functions are usually defined with respect to natural coordinates (η,ν) instead of (x,y) to better generalize the expressions. The natural coordinate system for this rectangular shell element is defined in Figure 4.1. The η and ν variables span the range of [-1,1] as the origin is defined at the centroid of the rectangle. In addition, the four nodes (N_i) in the element are also uniquely numbered by 1, 2, 3, and 4 going counterclockwise.

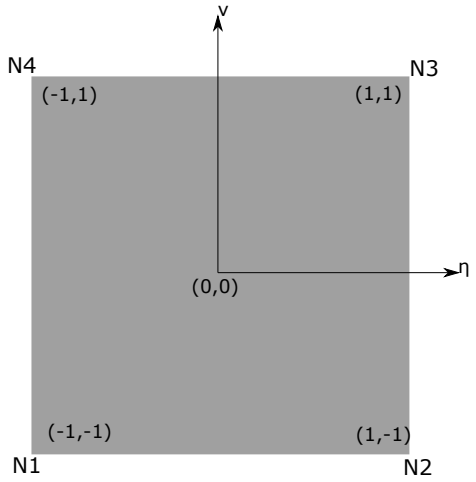

Figure 4.1: Illustration of the natural coordinate system for shape function definition and element node nomenclature.

In element domain, the natural coordinates (η,ν) can be related to the global in-plane coordinates (x,y) through a set of transformation equations:

$$x(\eta) = x(N1)\frac{1-\eta}{2} + x(N2)\frac{1+\eta}{2} \tag{4.2}$$

$$\eta(x) = \frac{2x}{x(N2) - x(N1)} - \frac{x(N1) + x(N2)}{x(N2) - x(N1)}$$

In Equation 4.2, $x(Ni)$ is the global x-coordinates for "node i" of the element. Equation 4.2 formulas global x-coordinate as a function of η, and vice versa. The identical relationship can be used to relate y-coordinate with ν:

$$y(\nu) = y(N1)\frac{1-\nu}{2} + y(N4)\frac{1+\nu}{2} \tag{4.3}$$

$$\nu(y) = \frac{2y}{y(N4) - y(N1)} - \frac{y(N1) + y(N4)}{y(N4) - y(N1)}$$

Using Equation 4.2 and 4.3 the unique shape functions (Equation 4.1) corresponding to the four element nodes can be expressed in the natural coordinate system.

$$\Phi_i(\eta, \nu) = \frac{1}{4}(1 + \eta(Ni)\eta)(1 + \nu(Ni)\nu); \quad i = 1, 2, 3, 4 \tag{4.4}$$

In Equation 4.4, $\eta(Ni)$ and $\nu(Ni)$ correspond to the natural coordinate value at "node i" of the element (see Figure 4.1). Thus, the full two-dimensional in-plane displacement field ($\begin{bmatrix} u_e(\eta,\nu) \\ v_e(\eta,\nu) \end{bmatrix}$, a 2×1 vector) for an element can be formulated as a matrix product between the shape functions and the nodal displacement d.o.f.:

$$\begin{bmatrix} u_e(\eta,\nu) \\ v_e(\eta,\nu) \end{bmatrix} = \mathbf{G} \begin{bmatrix} u(Ni) \\ v(Ni) \end{bmatrix}_{\times 4, i=1,2,3,4} \tag{4.5}$$

$$\mathbf{G} = \begin{bmatrix} \Phi_1 & 0 & \Phi_2 & 0 & \Phi_3 & 0\Phi_4 & 0 \\ 0 & \Phi_1 & 0 & \Phi_2 & 0 & \Phi_3 & 0\Phi_4 \end{bmatrix}$$

In Equation 4.5, **G** is a 2×8 matrix containing all the shape functions and $\begin{bmatrix} u(Ni) & v(Ni) \end{bmatrix}^T_{\times 4, i=1,2,3,4}$ is a 8×1 vector containing the nodal in-plane displacement values in sequential order of i = 1,2,3,4.

4.1.2 Nonconforming Element

Historically, there are two kinds of frequently used plate bending elements that are both C^1 continuous and also only require four element nodes (no interior nodes) [4, 29, 36]. They are the conforming element and the nonconforming elements. They are applicable only to the w_0 displacement variable. The main distinguishing feature between these two element is that the nonconforming element is only C^1 continuous inside element domain. Its shape functions does not satisfy the inter-element continuity of slopes (derivative of transverse displacement) at the element boundaries. However, the nonconforming element was still really popular in historical perspective, since they have found to be super effective in practical applications and requires a lot less computational resource than the conforming counterpart [4, 9, 29]. Hence, both the conforming and nonconforming element will be considered for this study.

The nonconforming element considered here is called the Adini-Clough Element. Adini-Clough Element requires a total of twelve nodal d.o.f.. The transverse displacement (w_0) as well as the transverse slopes in x and y direction ($\frac{\partial w_0}{\partial x} = w_x$, $\frac{\partial w_0}{\partial y} = w_y$) will be required to uniquely define the two-dimensional element-wise spatial approximation scheme. Correspondingly, the interpolation function employed is formed by a mixture of twelve polynomial terms.

$$\Psi(x, y) = a + bx + cy + dx^2 + exy + fy^2 + gx^3 + hx^2y + ixy^2 + jy^3 + kx^3y + lxy^3 \quad (4.6)$$

Similar to the plane stress element, there are twelve of these Ψ functions, each with a unique set of constants (a to l), which are determined by enforcing the sum of product between the twelve shape functions and the twelve nodal d.o.f. to exactly reproduce the nodal d.o.f. values at the nodal (x,y) position. Note the difference here is that since the nodal d.o.f. involves slopes, thus differentiation is required for the above procedure.

It is also interesting to note that in Equation 4.6, although the highest power (combining x and y) on the individual terms is four, however the polynomial itself does not form a complete 4th order order polynomial, because it does not include all the 4th power terms on the Pascal's Triangle. Hence, shape functions like these belong to the serendipity family of Lagrange elements. These elements are unique since their shape functions cannot be formed by tensor products of complete polynomial interpolation functions.

Again using the same natural coordinate definition shown in Figure 4.1 and Equation 4.2 and Equation 4.3, the twelve shape functions can be expressed in the natural coordinate system.

$$\Psi_{i,w}(\eta,\nu) = \frac{1}{8}[1+\eta(Ni)\eta][1+\nu(Ni)\nu][2+\eta(Ni)\eta+\nu(Ni)\nu-\nu^2-\eta^2] \tag{4.7}$$
$$\Psi_{i,w_x}(\eta,\nu) = \frac{1}{8}[\eta(Ni)][\eta(Ni)\eta-1][1+\nu(Ni)\nu][1+\eta(Ni)\eta]^2$$
$$\Psi_{i,w_y}(\eta,\nu) = \frac{1}{8}[\nu(Ni)][\nu(Ni)\nu-1][1+\eta(Ni)\eta][1+\nu(Ni)\nu]^2$$
$$i = 1,2,3,4$$

Using Equation 4.7, the full element-wise transverse displacement approximation as a function of η and ν can be formulated as a matrix product.

$$w_e(\eta,\nu) = \mathbf{H}\begin{bmatrix} w(Ni) \\ \frac{\partial w_0}{\partial x}(Ni) \\ \frac{\partial w_0}{\partial y}(Ni) \end{bmatrix}_{\times 4, i=1,2,3,4} \tag{4.8}$$
$$\mathbf{H} = \begin{bmatrix} \Psi_{i,w} & \Psi_{i,w_x} & \Psi_{i,w_y} \end{bmatrix}_{\times 4, i=1,2,3,4}$$

The $\mathbf{H}$ is a 1×12 matrix containing all of the shape functions and $\begin{bmatrix} w(Ni) \\ \frac{\partial w_0}{\partial x}(Ni) \\ \frac{\partial w_0}{\partial y}(Ni) \end{bmatrix}_{\times 4, i=1,2,3,4}$ is a 12×1 vector containing all the nodal transverse displacements and slopes. Both of the variables mentioned above are constructed sequentially through the nodes, following the order of i = 1,2,3,4.

For simplicity of notation, the thin-shell element formulated by combining plane stress element with nonconforming bending element will be denoted as the "RQNC4" element (R for rectangular shape, Q for the plane-stress element, NC for nonconforming, and 4 for four nodes).

4.1.3 Conforming Element

The conforming plate bending element considered in this study is the Bogner-Fox-Schmit element. This element requires a total of sixteen nodal d.o.f., as for each node, the values of displacement (w_0), slopes ($\frac{\partial w_0}{\partial x} = w_x$, $\frac{\partial w_0}{\partial y} = w_y$), in addition to a second order mixed partial

term ($\frac{\partial^2 w_0}{\partial x \partial y} = w_{xy}$) are required. Evidently, the conforming element has more d.o.f. than nonconforming element, thus requiring more computational resource. Also, the addition of the mixed partial term is more difficult to interpret physically. Nonetheless, the polynomial terms forming the conforming shape functions are shown in Equation 4.9.

$$\begin{aligned}\Psi(x,y) = a + bx + cy + dx^2 + exy + fy^2 + gx^3 + hx^2y + ixy^2 + jy^3 + kx^3y \\ + lx^2y^2 + mxy^3 + nx^3y^2 + ox^2y^3 + px^3y^3\end{aligned} \tag{4.9}$$

The conforming element also belongs to the serendipity family. Using the same idea as described in the previous two sections, the sixteen shape functions can be expressed in the natural coordinates.

$$\begin{aligned}\Psi_{i,w}(\eta,\nu) &= \frac{1}{16}[\eta + \eta(Ni)]^2[\eta(Ni)\eta - 2][\nu + \nu(Ni)]^2[\nu(Ni)\nu - 2] \\ \Psi_{i,w_x}(\eta,\nu) &= \frac{1}{16}[\eta(Ni)][\eta(Ni) + \eta]^2[1 - \eta(Ni)\eta][\nu(Ni) + \nu]^2[\nu(Ni)\nu - 2] \\ \Psi_{i,w_y}(\eta,\nu) &= \frac{1}{16}[\nu(Ni)][\eta(Ni) + \eta]^2[\eta(Ni)\eta - 2][\nu(Ni) + \nu]^2[1 - \nu(Ni)\nu] \\ \Psi_{i,w_{xy}}(\eta,\nu) &= \frac{1}{16}[\nu(Ni)][\eta(Ni)][\eta + \eta(Ni)]^2[1 - \eta(Ni)\eta][\eta(Ni) + \eta]^2[1 - \nu(Ni)\nu] \\ & \qquad i = 1,2,3,4\end{aligned} \tag{4.10}$$

Using Equation 4.10, the full element-wise transverse displacement approximation as a function of η and ν can be formulated as a matrix product.

$$\begin{aligned}w_e(\eta,\nu) &= \mathbf{H}\begin{bmatrix} w(Ni) \\ \frac{\partial w_0}{\partial x}(Ni) \\ \frac{\partial w_0}{\partial y}(Ni) \\ \frac{\partial^2 w_0}{\partial x \partial y} \end{bmatrix}_{\times 4, i=1,2,3,4} \\ \mathbf{H} &= \begin{bmatrix} \Psi_{i,w} & \Psi_{i,w_x} & \Psi_{i,w_y} & \Psi_{i,w_{xy}} \end{bmatrix}_{\times 4, i=1,2,3,4}\end{aligned} \tag{4.11}$$

The $\mathbf{H}$ is a 1×16 matrix containing all of the shape functions, and $\begin{bmatrix} w(Ni) & \frac{\partial w_0}{\partial x}(Ni) & \frac{\partial w_0}{\partial y}(Ni) & \frac{\partial^2 w_0}{\partial x \partial y}(Ni) \end{bmatrix}^{\mathsf{T}}_{\times 4, i=1,2,3,4}$ is a 16×1 vector containing all the nodal transverse displacement, slopes, and the mixed partial term. Both of the variables mentioned above are constructed sequentially through the nodes, following the order of i = 1,2,3,4.

Again to simplify the notations, the thin-shell element formulated by plane stress element combined with conforming bending element will be denoted as the "RQC4" element (R for rectangular shape, Q for the plane-stress element, C for conforming, and 4 for four nodes)

4.2 Integrating Elements into the Weak Form

The element level displacement field approximations presented above can be incorporated directly into the weak form shown in Equation 3.32. The weak form will thus be applied in the element specific domain instead of the problem domain. Since the Galerkin method is applied, the weight functions (δu_0, δv_0, δw_0) in the weak form is approximated using the same set of shape function matrices ($\mathbf{G}$, $\mathbf{H}$) as the displacement field. It is important to note that for the displacement field, the matrices are multiplied with nodal d.o.f. to approximate the two-dimensional element displacement field. For the weight functions as discussed in section 3.2.2, the choice can be arbitrary as long as they belong to the appropriate Sobolev spaces as defined in Equation 3.33 and 3.34. Hence, the vector of constant multiples with the shape functions will be arbitrary for the weight functions. In fact, since the same weight functions appear on both side of Equation 3.32, this means that the vector of constants can be factored out, and the rest of the equation would still need to equal to satisfy the weak form.

Since the weak form consists of derivatives of the displacement field, in order to approximate the displacement variables using matrix products, two spatial derivatives operator matrices, ∇_s and ∇_z are defined.

$$\nabla_s = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \qquad \nabla_z = \begin{bmatrix} \frac{\partial^2}{\partial x^2} \\ \frac{\partial^2}{\partial y^2} \\ 2\frac{\partial^2}{\partial x \partial y} \end{bmatrix} \tag{4.12}$$

To simplify notations, define the nodal d.o.f. as vector quantities: $\hat{d} = \begin{bmatrix} u(Ni) & v(Ni) \end{bmatrix}^{\intercal}_{\times 4, i=1,2,3,4}$, and $\hat{w} = \begin{bmatrix} w(Ni) & \frac{\partial w_0}{\partial x}(Ni) & \frac{\partial w_0}{\partial y}(Ni) \end{bmatrix}^{\intercal}_{\times 4, i=1,2,3,4}$ or $\hat{w} = \begin{bmatrix} w(Ni) & \frac{\partial w_0}{\partial x}(Ni) & \frac{\partial w_0}{\partial y}(Ni) & \frac{\partial^2 w_0}{\partial x \partial y}(Ni) \end{bmatrix}^{\intercal}_{\times 4, i=1,2,3,4}$.

Now, combining the operators in Equation 4.12, the vector and matrix definitions above and in the previous two sections, as well as the relationship between generalized forces and displacement derivatives in Equation 3.11 and Equation 3.12, the element level weak form can be derived in matrix format:

$$\begin{bmatrix} \int_{\Omega_E}(\nabla_s \mathbf{G})^\intercal A(\nabla_s \mathbf{G})d\Omega_E & -\int_{\Omega_E}(\nabla_s \mathbf{G})^\intercal B(\nabla_z \mathbf{H})d\Omega_E \\ -\int_{\Omega_E}(\nabla_z \mathbf{H})^\intercal B(\nabla_s \mathbf{G})d\Omega_E & \int_{\Omega_E}(\nabla_z \mathbf{H})^\intercal D(\nabla_z \mathbf{H})d\Omega_E \end{bmatrix} \begin{bmatrix} \hat{d} \\ \hat{w} \end{bmatrix} = \mathbf{F};$$

(4.13)

$$\mathbf{F} = \begin{bmatrix} \int_{C_E} \mathbf{G}^\intercal \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} \bar{N_{xx}} \\ \bar{N_{yy}} \\ \bar{N_{xy}} \end{bmatrix} dS \\ -\int_{\Omega_E} \mathbf{H} q d\Omega_E + \int_{C_E} \left\{ \mathbf{H}\bar{Q}_n - \left(\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \mathbf{H} \right)^\intercal \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} \bar{M_{xx}} \\ \bar{M_{yy}} \\ \bar{M_{xy}} \end{bmatrix} \right\} dS \end{bmatrix}$$

The matrix multiplied with the nodal displacement vector ($\begin{bmatrix} \hat{d} \\ \hat{w} \end{bmatrix}$) is the FEA stiffness matrix. Recall that the A, B, and D matrices are the generalized forces and strains relation matrices. The left top quadrant of that matrix is a 8x8 sub-block containing the in-plane membrane stiffness. The right top and left bottom quadrants of that matrix are 8x12 and 12x8 for nonconforming element or 8x16 and 16x8 for conforming element, these sub-blocks prescribes the bending and stretching coupled twisting stiffness. The right bottom sub-block is 12x12 as it describes the bending stiffness of the laminate. The first part (8x1 vector) of the weighted boundary force vector ($\mathbf{F}$) is the in-plane applied forces, while the second part (12x1 vector) contains the transverse direction applied forces in addition to applied moments.

Since the thin-shell element used in this study is strictly rectangular in shape, thus the implementation of FEA will only consider analysis on rectangular shaped laminate plates. This means that C_E boundary curve in the integral will only be straight lines along x-axis or y-axis, while the θ value will be either 0 or 90 degrees depending on the boundary orientation. Nonetheless, the weak form matrix formulation shown above works for any laminate structure, no matter how complex the geometry or material properties are.

The last numerical method required to fully discretize the weak form into products of matrix with scalar values only is the incorporation of a numerical integration scheme. The numerical integration scheme implemented for this study is the Gaussian quadrature rule.

Table 4.1: Gauss Quadrature Points and the Corresponding Weighting for Integral Approximation Scheme

Number of Quadrature Points	**Quadrature Points Location in Natural Coordinate**	**Weights**
1	0	2
2	$\pm\sqrt{1/3}$	1
3	0 $\pm\sqrt{3/5}$	$\frac{8}{9}$ $\frac{5}{9}$
4	$\pm\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}$ $\pm\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18+\sqrt{30}}{36}$ $\frac{18-\sqrt{30}}{36}$

4.2.1 Gaussian Quadrature

The Gaussian quadrature approximates the integrals in Equation 4.13 by evaluating the terms in the integral at a specific number of Gauss points and multiplied them by the corresponding weight values. The sum of the weighted products can then approximate the integral. Since the shape function matrices **G** and **H** are both formulated in the natural coordinate frame, hence the Gaussian quadrature points will be selected in the same coordinates as well. The quadrature points and the corresponding weights are given in Table 4.1.

Since the numerical integration scheme is defined using natural coordinates, there are a few caveats to consider when evaluating the weak form integrals in Equation 4.13. The integrals are either computed over the element domain ($d\Omega_E$) or the element boundary (dS), both are defined according to the global coordinate system, i.e. $d\Omega_E = dxdy$ and $dS = dx$ or $dS = dy$. To make the integrated term and the integral domain compatible, Jacobians from transformation of variables and derivatives from chain rule (i.e. taking the x and y derivatives of **G** and **H** that are defined using natural coordinates) need to be computed and multiplied with the integrals. Note that the transformations of variables are already derived in Equation 4.2 and Equation 4.3. Using those expressions, it is straight forward to show that:

$$\begin{aligned} \frac{d\eta}{dx} &= \frac{2}{x(N2)-x(N1)} \\ \frac{d\nu}{dy} &= \frac{2}{y(N4)-y(N1)} \end{aligned} \tag{4.14}$$

4.3 FEA Computer Module Information

At this instance, all the necessary equations are derived and the FEA framework can be readily implemented into a computer code module. The MATLAB code implementation is shown in Appendix A. It is necessary to reiterate again that the current MATLAB suite can only handle rectangular laminate plate geometries.

The general purpose of the FEA module is that given a set of applied boundary forces and displacement restrictions, these can be incorporated into the element level weak form in Equation 4.13 (with the application of Gaussian quadrature), and the nodal displacement vector $\begin{bmatrix}\hat{d} & \hat{w}\end{bmatrix}^{\intercal}$ at static equilibrium can be obtained.

The first step in operating the program is to specify the dimensions of the laminate plate and the mesh size desired. This can be done by assigning values to "xdim" and "ydim" variables, which are the side lengths of the laminate, in addition to specifying "Ny" and "Nx" values, which are the number of mesh element desired along y and x dimensions. Subsequently, running the "meshing.m" function using these variables as inputs will generate the required mesh and geometry layout, as well as storing the relevant element and nodal information into "connec" and "nodecoord" matrices. Specifically, the "connec" variable is an N×5 matrix, where the N is the total number of elements in the mesh, and each row contains the element number as well as the numerical label of the four attached nodes arranged in counterclockwise orientation. The "nodecoord" matrix is M×4 in size, as M is the total number of nodes from meshing, and each row contains the numerical label of the node, x-coordinate, y-coordinate, and z-coordinate related to that node. An example of a discretized laminate plate domain is shown in Figure 4.2. By default, the program assigns one of the corners of the plate as the origin of the global coordinate system, and the x and y axes are as defined so that the laminate lies strictly on the $x > 0$ and $y > 0$ quadrant.

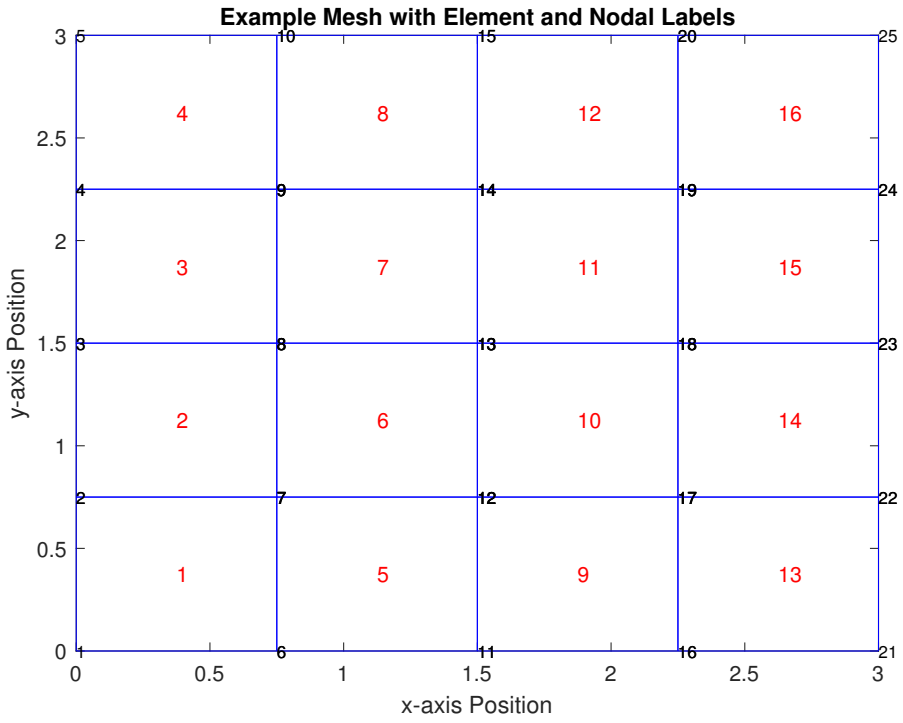

Figure 4.2: Graphical illustration of output from the meshing procedure. The nodes and the elements are labeled with numerical values.

The other input information needed is related to the lamina layer properties. This can be done by specifying an N×7 matrix called **ply** as shown below. The N represent the number of lamina layers required. Each row consists of material modulus (G_{12}, E_1, E_2), Poisson ratio (ν), fiber orientation in degrees (θ), and layer thickness (h) related to each lamina. Note that because of the symmetry of the constitutive model stiffness matrix (as shown in Equation 3.1), only 4 out of 5 non-geometry material constants are independent. The dependent relationship is given as: $\nu_{21} = \nu_{12}\frac{E_2}{E_1}$. All the units should be in SI unit or self-consistent.

$$\mathbf{ply} = \begin{bmatrix} G_{12} & E_1 & E_2 & \nu_{12} & \nu_{21} & \theta & h \end{bmatrix}_{\times N} \tag{4.15}$$

The last input variables needed are the boundary conditions. By default, all nodal displacements are assumed to be free to move and all generalized forces are assumed to be zero. To specify applied forces and enforce displacement conditions, a MATLAB structure called "BC" needs to be defined. The "BC" structure should have three field variables, namely "BC.pos", "BC.type", and "BC.val". All three field variables should be defined as Nx1 cell arrays, where N is the number of boundary conditions specified (including both

force and displacement). Each row/cell of "BC.pos" contains two entities, the first specifying the range of x-coordinates affected by the current boundary condition, and the second specifying the range of y-coordinates. Note that these positions can be entered as a vector containing the end points of the coordinate range, or as a scalar, which indicates a point condition. On the other hand, each row/cell of the "BC.type" field variable contains a list of strings indicating the type of boundary conditions applied at current positions. The list of strings available includes displacement specifications: "x", "y", "w", "wx", "wy" (and "wxy" for RQC4 element); or generalized force specifications: "Nx", "Ny", "Nxy", "Mx", "My", "Mxy","Q","q". Lastly, for the "BC.val" field variable, each row/cell contains the value of the boundary condition corresponding to the ones identified in "BC.type" field variable. Note that the value entries must follow the each sequence as the type specifications, and each applied boundary condition must be assigned a scalar value.

For examples on how to format all these input variables, please refer to Appendix A.

Using the mentioned input values alongside a desired Gaussian quadrature order (p), one can run the "maindriver.m" code segment to generate the full FEA stiffness matrix ("ktot"), as the process calls on element specific subroutines to compute element level stiffness matrix contributions. The subroutines are responsible for computing the shape functions, the numerical integration scheme for integrals, the modulus weighted midplane, and the relevant material matrices for each element. The code segment in "maindrive.m" is responsible for traversing through all the elements and assemble the output to correct positions in the overall stiffness matrix.

Using mainly the "ktot" and the "BC" variables, the "solve_noncon.m" (for RQNC4 element) and the "solv_con.m" (for RQC4 element) sub-functions can be called to solve for all the nodal displacement degree of freedoms after deformation. The main purposes of the two sub-functions are computing the global assembled $\mathbf{F}$ vector as mentioned in Equation 4.13 and use it to solve the global assembled displacement vector $\begin{bmatrix}\hat{d} & \hat{w}\end{bmatrix}^\intercal$. In the code, the assembly for the displacement vector containing the nodal d.o.f. is structured as shown below:

$$\hat{d}^\intercal = \begin{bmatrix} \underbrace{u \;\; v}_{\text{node 1}} & \underbrace{u \;\; v}_{\text{node 2}} & \cdots \end{bmatrix}$$

$$\hat{w}^\intercal = \begin{bmatrix} \underbrace{w \;\; w_x \;\; w_y}_{\text{node 1}} & \underbrace{w \;\; w_x \;\; w_y}_{\text{node 2}} & \cdots \end{bmatrix} \text{ or } \begin{bmatrix} \underbrace{w \;\; w_x \;\; w_y \;\; w_{xy}}_{\text{node 1}} & \underbrace{w \;\; w_x \;\; w_y \;\; w_{xy}}_{\text{node 2}} & \cdots \end{bmatrix} \tag{4.16}$$

Finally, the global displacement vector and the reaction force vector ($\mathbf{F}$) can be inputted into the graphing function to obtain various data contour graphs.

Chapter 5

FEA Model Verification

5.1 Overview

The process of FEA model verification is vital to assuring the integrity of the computer program implementation. Verification testing not only allows one to add confidence to the numerical procedures in the analysis suite, but it also serves the purpose of accessing and comparing the performance of the elements in different loading scenarios, thus shed light on how the choice of element will influence the accuracy of the outcome. In literature, the majority of the quantitative verification methodologies involves solving various numerical benchmark scenarios, and comparing the newly implemented FEA outputs with the results obtained using other well-established simulations, theoretical hand analysis, or other literature results [4, 20, 21, 26, 33]. In the duration, qualitative evaluations of model outputs by visualization of data contours are also encouraged. This could be done by examining the reaction force distribution and checking for discontinuities or distortions in deformed mesh [33]. The qualitative aspects allow physical interpretation of the model output, which would help in identifying phenomenons that would otherwise be hard to understand from numerical data only. For example, a well-known aspect of laminate FEA analysis using First Order Deformation Theory is the existence of hourglassing mode or shear-locking mode from the analysis. Hourglassing is usually caused by reduced integration scheme, while shear-locking is due to bending element being too stiff and occurs even with full integration scheme. Both could heavily deviate the accuracy of your simulation, however both could also be identified from deformed mesh visualization. For this manuscript, the model verification will be carried out in two steps. The first is verification of FEA stiffness matrix computations through rigid body modes analysis. The second is verification of FEA as an assembly by solving archetypal load tests, and comparing its output with that of ABAQUS simulations (a popular commer-

cial FEA software). The ABAQUS simulations in this case is considered as the ground truth of results.

5.2 Rigid Body Modes

The rigid body modes as represented by the vector nodal d.o.f. for the modeled laminate are related the eigenvectors corresponding to the zero eigenvalues of the FEA stiffness matrix. Since the implemented FEA treats laminate as 3D structure, the stiffness matrix is expected to have 3 translation modes and 3 rotational modes along the global axes. The importance of this analysis is that a correctly constructed stiffness matrix must have the appropriate amount and shape of rigid body modes, given arbitrary (reasonable) material properties and mesh dimensions. This can be confirmed through numerical eigenvalue analysis.

The arbitrarily (but reasonable) selected material input selected for the rigid body mode test case is given below, following the convention labeled in Equation 4.15.

$$\mathbf{ply} = \begin{bmatrix} 4.8e9 & 230e9 & 230e9 & 0.25 & 0.25 & 0 & 0.005 \end{bmatrix}$$

The number of rigid body modes can be obtained by checking the rank of the stiffness matrix ("ktot" variable) from the **ply** input. To check whether the rigid body mode shapes are correct, the eigenvectors related to pure translational and pure rotational motions can be constructed and multiplied with the stiffness matrix. If the product is zero, then that means these modes do correspond to zero eigenvalues. For the case of translational motions, the nodal d.o.f. vectors ($\begin{bmatrix} \hat{d} & \hat{w} \end{bmatrix}^{\mathsf{T}}$) can be constructed as shown below, following the convention given in Equation 4.3.

Axis of Translation	**Nodal d.o.f. Vector**
x-axis	$\hat{d} = \begin{bmatrix} 1 & 0 & 1 & 0 & ..... \end{bmatrix}$ $\hat{w} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & ..... \end{bmatrix}$
y-axis	$\hat{d} = \begin{bmatrix} 0 & 1 & 0 & 1 & ..... \end{bmatrix}$ $\hat{w} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & ..... \end{bmatrix}$
z-axis	$\hat{d} = \begin{bmatrix} 0 & 0 & 0 & 0 & ..... \end{bmatrix}$ RQNC4: $\hat{w} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & ..... \end{bmatrix}$ or RQC4: $\hat{w} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & ..... \end{bmatrix}$

For the rotational modes, the initial expectation is that the same method of constructing the nodal displacement vector can be done through appropriate rotational coordinate mapping. For example, for the in-plane rotational mode, the rotated coordinate mapping to original coordinates can be related through rotation matrix: $\begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$ [3]. However, as it turns out, the nodal displacement vector generated from such a mapping is not an eigenvector corresponding to zero eigenvalue of the stiffness matrix. Thus, a different procedure would be required to investigate the rotational modes. Recall that there are six eigenvectors corresponding to zero eigenvalues. These eigenvectors can be obtained through MATLAB using the "eigs" function. However, the formats of these eigenvectors are not as intuitive since the MATLAB numerically generated eigenvectors will not correspond to one single rigid body mode only, as typically they are a mixture of rigid body modes. Nonetheless, the advantage of structuring the nodal displacement vector as shown in Equation 4.3 is that since the in-plane rigid body modes only result in in-plane displacement, thus eigenvectors corresponding to pure in-plane modes will only have the first portion ($\hat{d}$ part) of the vector populated with values, while rest of the entries ($\hat{w}$ part) are zero. The same can also be said regarding eigenvectors for pure out-of-plane rigid body modes. Hence using the above logic, amongst the 6 eigenvectors generated from MATLAB, one could easily distinguish the three out-of-plane vectors with the three in-plane vectors. Since two of the in-plane modes, namely x-axis and y-axis translations, were readily identified, thus the in-plane rotational mode can be derived by treating this as a change of basis problem in linear algebra. Essentially, the vector space corresponding to in-plane rigid body motion has a dimension of three. The eigenvectors generated from MATLAB "eigs" forms a full basis to that vector space.

The eigenvectors from pure x-axis and y-axis translations are orthogonal to each other and belongs to a different basis of the same vector space, however missing a third component to complete the basis. Thus, to find that third basis vector, a linear algebra manipulation is required:

Theorem 1 *Let $U \subseteq \mathbb{R}^n$ be a vector space and has a dimension of 3. Let $\{g_1, g_2, g_3\} \subseteq U$ form a basis set for U. Let $q_1 \in U$ and $q_2 \in U$ be two non-zero linearly independent vectors. Thus, there exist two sets of non-zero constants, $[c_{11}, c_{21}, c_{31}]$ and $[c_{12}, c_{22}, c_{32}]$ such that:*
$q_1 = \sum_{i=1}^{3} c_{i1} g_i$ and $q_2 = \sum_{i=1}^{3} c_{i2} g_i$.
To find a third vector q_3 such that it is linearly independent to q_1 and q_2, the following procedure can be used:
Let $\boldsymbol{C} = \begin{bmatrix} c_{11} & c_{21} & c_{31} \\ c_{12} & c_{22} & c_{32} \end{bmatrix}$. $\boldsymbol{C}$ will have a rank of two since q_1 and q_2 are linearly independent. So a non-zero vector $[c_{13}, c_{23}, c_{33}] \in Null(\boldsymbol{C})$ can be obtained. Let $q_3 = \sum_{i=1}^{3} c_{i3} g_i$, this is the q_3 vector that's needed.

Theorem 1 can be readily implemented into MATLAB and used to obtain the in-plane rotation eigenvector. The plots for this rigid body motion using the eigenvectors are shown in Figure 5.1 for both RQNC4 and RQC4 element.

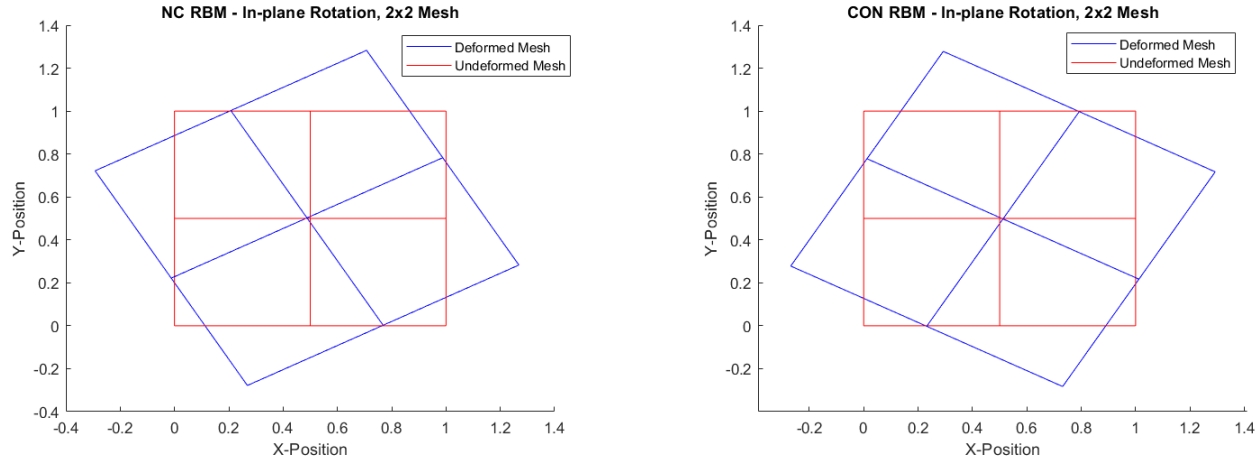

Figure 5.1: **Left**: In-plane rigid body rotation mode using RQNC4 element. **Right**: In-plane rigid body rotation mode using RQC4 element.

Evidently from Figure 5.1, the area of the rotated laminate plate increased compared to the original size, which explains why the displacement vector generated from coordinate

rotation did not qualify as an eigenvector. The area increase here does not mean that the stiffness matrix is implemented incorrectly, in fact it is an artefact due to usage of small strain equations in strain-displacement relationship for FEA derivation [3]. Rigid rotations will induce non-zero normal strains when the small strain tensor is used, and the magnitude of the strain scales with the rotation angle. Hence, this is more of an expected behavior.

The out-of-plane rotation modes can be obtained using the same algebraic manipulation presented in Theorem 1. Note that the process is repeated twice to generate two eigenvectors that are linearly independent to each other and to the z-axis translation vector. This is because there are two out-of-plane rotation modes. The graphical illustrations of the out-of-plane rotations about x-axis and y-axis for both elements are presented in Figure 5.2 and Figure 5.3.

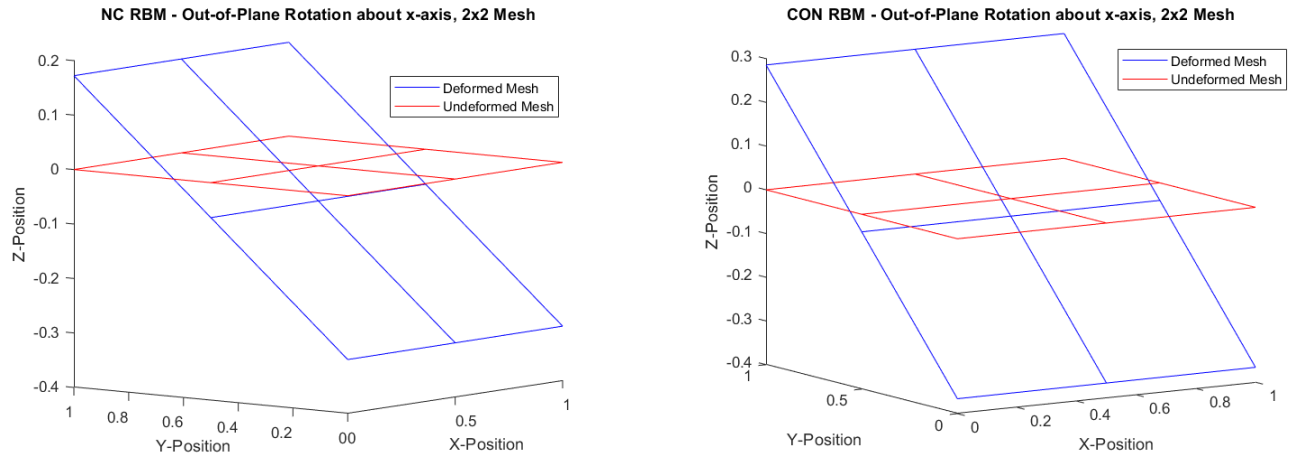

Figure 5.2: **Left**: Out-of-plane rigid body rotation mode about x-axis using RQNC4 element. **Right**: Out-of-plane rigid body rotation mode about x-axis using RQC4 element.

Another interesting phenomenon arose when analyzing the out-of-plane rigid body modes. For the case of RQC4 element, when a 1×1 mesh is selected alongside the two Gaussian point integration scheme, the resultant FEA stiffness matrix had an fourth out-of-plane "rigid body mode". This additional "rigid body mode" is independent of the laminate material input (i.e. the "ply" variable) and its deformed mesh configuration is illustrated in Figure 5.4. Evidently from the image, the is not a true rigid mode since warping occurs throughout the deformed structure and is exacerbated at the element corners. Through further analysis,

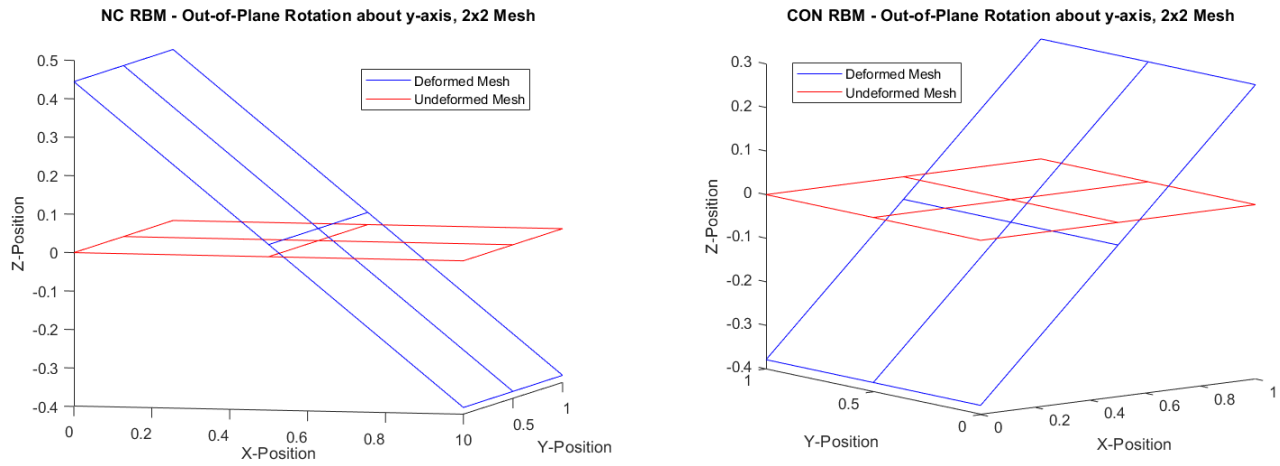

Figure 5.3: **Left**: Out-of-plane rigid body rotation mode about y-axis using RQNC4 element. **Right**: Out-of-plane rigid body rotation mode about y-axis using RQC4 element.

this additional mode was deemed as a numerical artefact. The reason is because the value of strain vector is zero at all four of the element Gauss quadrature points (two per dimension). Hence even with the displacement deformation, the strain energy related stays zero according to the numerical integration scheme. This issue can be alleviated by simply increasing either the number of Gauss quadrature points used or the mesh size.

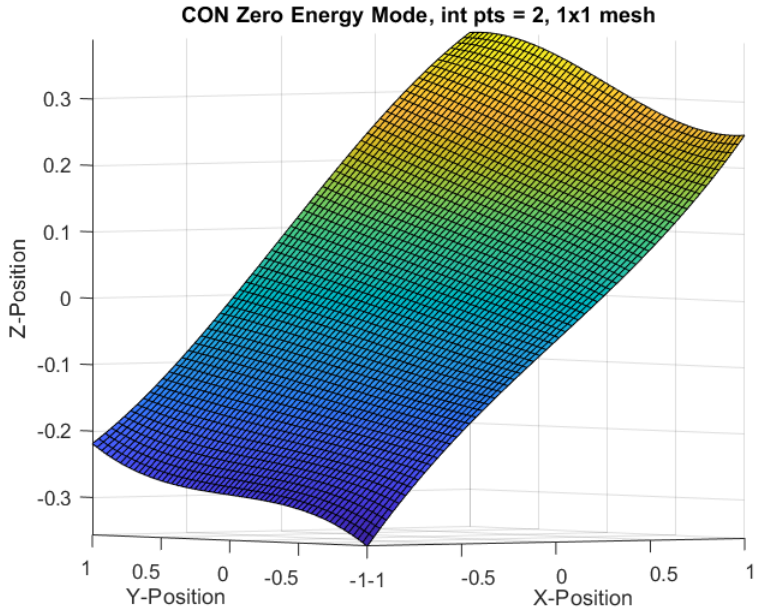

Figure 5.4: Graphical illustration the zero energy "rigid body mode" for RQC4 element when analyzed using 1×1 mesh and 2 Gauss quadrature points.

In conclusion, the implemented FEA program has the ability to generate all six of rigid body motions using both the RQNC4 and RQC4 element, hence adding confidence to the numerical integrity of the program.

5.3 Numerical Case Studies

Numerical case studies using various simple archetypal loading problems are done with the objectives of benchmarking the MATLAB FEA suite with commercial FEA software as well as comparing the performance of the RQNC4 and RQC4 elements. The commercial FEA software used in this study is the CAE ABAQUS software v2020. In terms of modeling, the 2D planar option was chosen alongside general purpose shell S4 element. The in-plane component of the S4 element uses a finite-membrane strain formulation, while the bending component is based on thick plate formulation that would also work for thin plate geometries [20]. For material properties input, the involvement of thick plate formulation means that the program requires additional shear modulus for the transverse direction (G_{23} and G_{13}) compared to the properties required for the MATLAB FEA suite. In order to maximize consistency, the transverse shear modulus were set to be 100 times more than the highest in-plane stiffness modulus during modeling. The idea is that the high transverse stiffness would artificially minimize displacement contributions from transverse shear. This makes the model more consistent with CLT, as the principal of CLT neglects transverse shear deformations completely. A key metric used to assess the performance of MATLAB FEA results with the ABAQUS output is the percent difference (%Diff) value. Let *var* represent the variable of interest for comparison, the percent difference for verification study is given as:

$$\%\text{Diff} = |\frac{var(\text{MATLAB FEA}) - var(\text{ABAQUS})}{var(\text{ABAQUS})}|$$

A few laminate nomenclature will also be defined here to help distinguish different cases in the analysis. For a laminate plate, the layerwise fiber orientation angles will be denoted by the vector:

$$layout = [\theta_1/\theta_2/\theta_3/...../\theta_n]$$

The θ_i value corresponds to the fiber orientation with respect to global x-axis at lamina i. This vector will be constructed in a top-down manner. A geometrical parameter called the Aspect Ratio (AR) will be defined as:

$$AR = \frac{\text{Shortest In-Plane Dimension}}{\text{Total Laminate Thickness}}$$

5.3.1 Plate Bending with Transversely Bounded Edges

For this case, a square plate is transversely bounded on all the outer edges and subjected to a distributed surface load of $5.0\,\mathrm{N\,m^{-2}}$ on the top surface as shown in Figure 5.5. Various different AR ratios were studied for this case to elucidate the potential influence of plate thickness on element accuracy. The detailed geometrical and material properties inputs are presented below:

Input Properties	**Variable Values**
plate side length	$3\,\mathrm{m}$
material properties	$E_1 = 172\,369 \times 10^6\,\mathrm{Pa}$ $E_2 = 6894.76 \times 10^6\,\mathrm{Pa}$ $G_{12} = 3447.38 \times 10^6\,\mathrm{Pa}$ $\nu_{12} = 0.25$ $\nu_{21} = \nu_{12}\frac{E_2}{E_1}$
	additional for ABAQUS: $G_{13} = G_{23} = 2 \times 10^{13}\,\mathrm{Pa}$
aspect ratio	AR = 5; 10; 25; 50; 100
laminate layout	$ply = [0/90/0]$ (see Figure 5.6)
loading applied	$q = 5.0\,\mathrm{N\,m^{-2}}$ at top surface.
boundary conditions	$w = 0;\ w_x = 0;\ w_y = 0;\ (w_{xy}) = 0$ for all four edges.

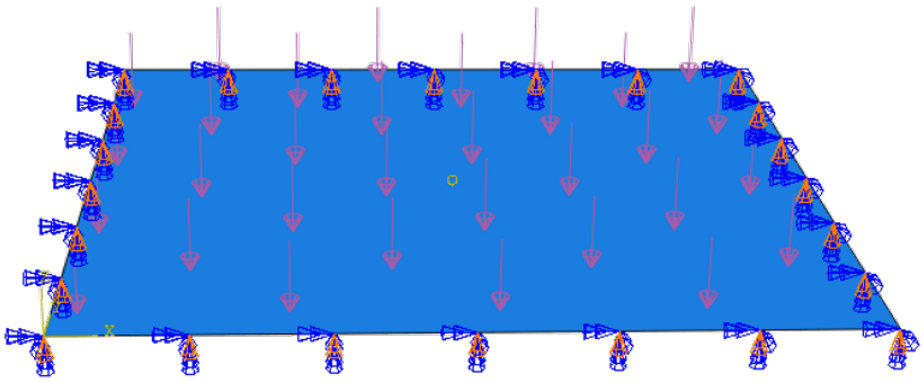

Figure 5.5: Boundary conditions and applied loading for the case study of square plate with transversely bounded edges. The red arrows represent the uniform pressure load, the triangular shapes on the boundary represent the relevant boundary condition.

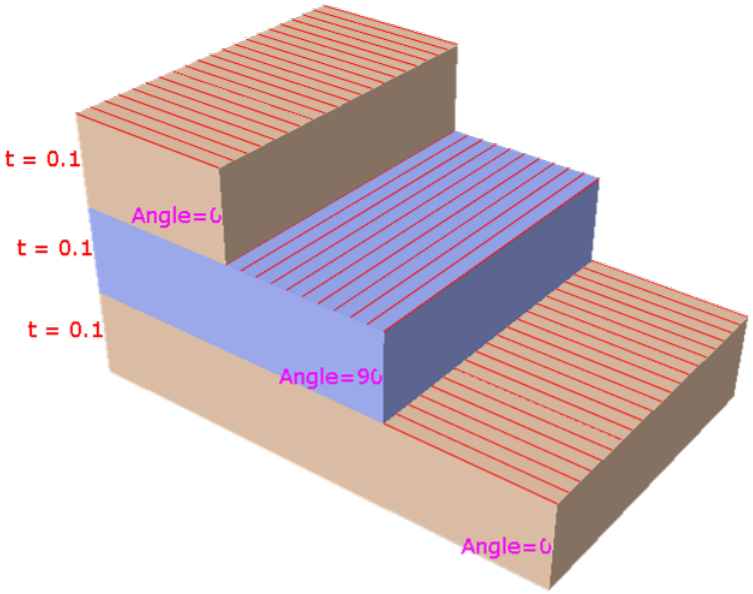

Figure 5.6: Laminate layout sequence for the case study of square plate with transversely bounded edges (plate thickness for this instance is 0.3 m).

The fiber angle layout for this problem was chosen to be a symmetrical one as this would decouple the bending mode and the in-plane membrane deformations, which allows pure comparison of the plate-bending portion of the shell-elements. The variation of AR for this numerical study adds geometrical variability in comparison cases. The transverse deflection at midpoint of the plate ($w(x = 1.5\,\text{m}, y = 1.5\,\text{m})$) was chosen to be the variable of interest (var) for both mesh convergence analysis and model verification study. The mesh sizes for both ABAQUS and MATLAB FEA were set so that both in-plane orientations have the same number of elements along the dimension.

Representations of the mesh convergence behaviors for MATLAB FEA are shown in Figure 5.8 and Figure 5.9 for the case of AR=100. An interesting observation about the RQNC4 element in Figure 5.8 is that it exhibits negligible convergence behavior with refinement of mesh compared to the RQC4 element. Note that the order of magnitude of both graphs are similar, however the RQC4 element's output changed by upto 50% while the RQNC4 element's output stayed constant. In fact this behavior was consistent for all loading scenarios that result in transverse displacement only or negligible in-plane displacements.

This phenomenon can be attributed to the fact that the nonconforming bending element does not satisfy the criteria of monotonic convergence with respect to mesh refinement. For FEA analysis, the monotonic convergence criteria includes a compatibility component and a completeness component [23, 41]. Often, a quantity called variational index, denoted by m, is defined as the highest spatial derivative order of displacement

variables in the weak form integrals. For the current study, as evident in Equation 3.32, the variational index has a value of $m = 2$ for the bending component due to the second order spatial derivatives on the w_0 variable. In addition, another parameter called the patch trial function is defined as the union of shape functions activated by setting a nodal d.o.f. to unity, while maintaining all other nodal d.o.f. at zero. Note that the related node is called the patch node. Hence, the compatibility part of the convergence criteria can be stated as: all the patch trial functions must be C^{m-1} continuous across element boundaries and C^m piecewise differentiable within each element domain. Now consider the case where the w_x nodal d.o.f. is set to unity for the nonconforming element. For rectangular mesh elements as shown in Figure 5.7, the central position of the patch node means that all four of the adjacent elements' Ψ_{w_x} shape functions (Equation 5.1) will be activated, and they will form the patch function set. For these patch functions, the second part of compatibility requirement is readily satisfied from shape function construction. The first requirement however, will not be satisfied. Consider specifically the quadrant 4 and quadrant 3 elements. For quadrant 4, the patch node is at the node 1 position (i=1), hence $\eta(N1) = -1$ and $\nu(N1) = -1$. For quadrant 3, the patch node is at the node 4 position (i=4), hence $\eta(N4) = -1$ and $\nu(N4) = 1$. Since the variational index is 2, the first compatibility requirement says that $\frac{\partial \Psi_{w_x}}{\partial y}$ must be continuous across the quadrant 3 and 4 boundary. This shape function derivative is computed for both quadrant and shown in Equation 5.1. Clearly, the functions for i=1 and i=4 are different, hence suggesting that the Ψ_{w_x} shape function is not C^1 continuous across the boundary of quadrant 3 and 4. This proves that the nonconforming element does not satisfy the monotonic convergence criteria, and thus is not guaranteed to experience convergence with mesh refinement.

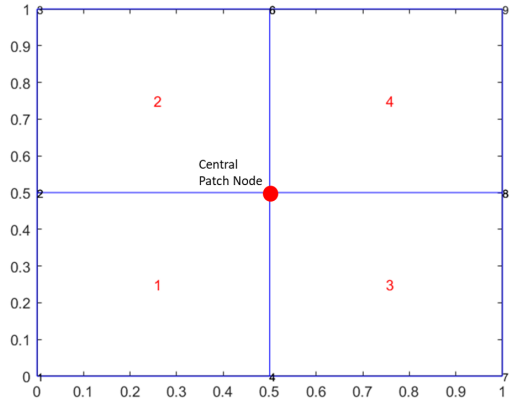

Figure 5.7: Central patch node and adjacent elements for rectangular mesh.

$$\Psi_{i,w_x}(\eta,\nu) = \frac{1}{8}[\eta(Ni)][\eta(Ni)\eta - 1][1+\nu(Ni)\nu][1+\eta(Ni)\eta]^2 \tag{5.1}$$

$$i = 1,2,3,4$$

$$\frac{\partial \Psi_{N1,w_x}}{\partial y} = \frac{1}{8}[\eta+1][-1][1-1\eta]^2\,; \quad i = 1$$

$$\frac{\partial \Psi_{N4,w_x}}{\partial y} = \frac{1}{8}[\eta+1][1][1-1\eta]^2\,; \quad i = 4$$

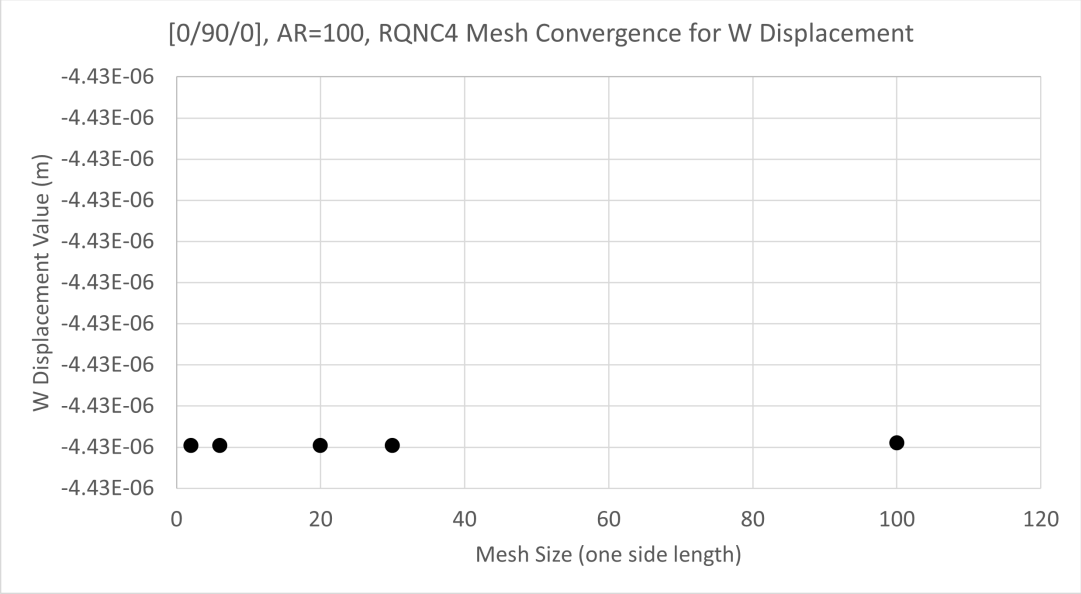

Figure 5.8: Representative mesh convergence behavior for RQNC4 element for the case of square plate with distributed load and transversely bounded edges.

The compatibility condition can be generalized to other plate elements as, if the normal slope across element boundaries is discontinuous, then it is considered as a nonconforming element and monotonic convergence is not guaranteed [25, 4, 29]. In fact, some research show that the convergence of nonconforming element might depend on mesh configuration [36].

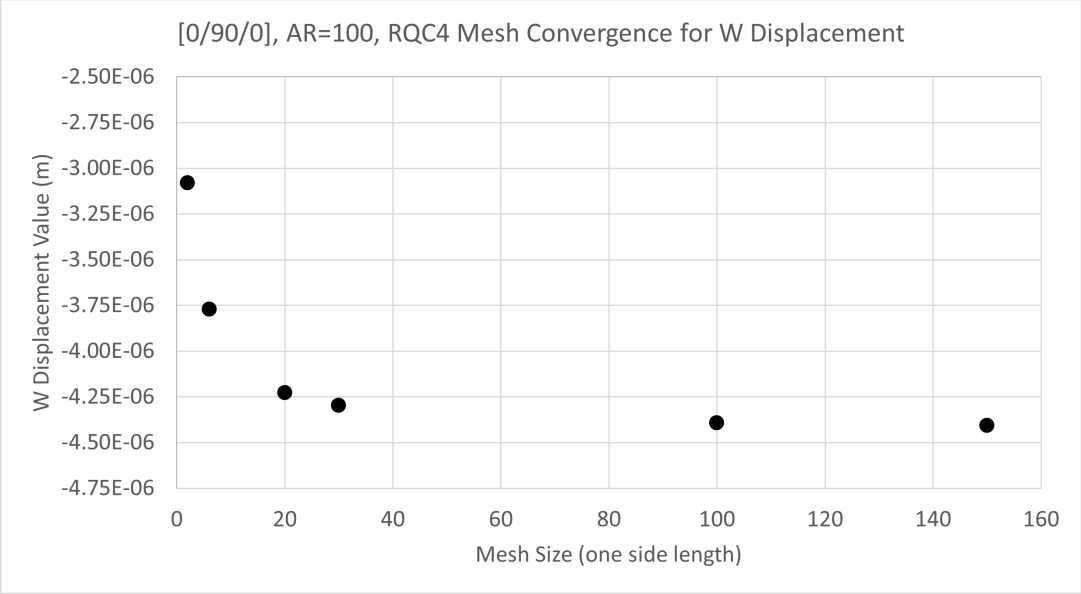

Figure 5.9: Representative mesh convergence behavior for RQC4 element for the case of square plate with distributed load and transversely bounded edges.

After performing mesh convergence analysis, the converged midplate transverse deflection values were extracted from numerical analysis using plates of various AR's. This was done both the elements, can the comparison between MATLAB FEA result with ABAQUS result in the form of %Diff is shown in Figure 5.10 as a function of AR. In general, Figure 5.10 shows that the performance of the two elements are similar in terms of accuracy since the maximum difference between the two %Diff data sets is within 2%. However, since the %Diff values for both data sets are near 50 %, this suggest that the accuracy of both elements are equality poor/accurate. Also, the precision for both is high over various aspect ratios since the maximum different within each data set across different AR is within 2%. These discoveries suggest when the effect of transverse shear deformations are excluded, both element presents equal amount of ability in terms of accurately predicting the transverse displacement. Also, both element can sustain the same level of accuracy for a wide range of plate thickness geometry.

On the other hand, the same case study was also modeled in ABAQUS by selecting a more realistic value for the transverse shear modulus components. The intention is that by removing the artificially stiffened transverse shear properties, the comparison would shed light on how addition of shear deflections would effect the accuracy of the MATLAB FEA model. For this analysis, the model in ABAQUS was set to have $G_{13} = 3447.38 \times 10^6$ Pa and $G_{23} = 1378.95 \times 10^6$ Pa. The comparison between this simulation result and the MATLAB FEA outputs were shown in Figure 5.11. Note that the MATLAB FEA data used here

is the same set of data used in the previous analysis, since the CLT based FEA cannot capture transverse shear effects. Evidently from Figure 5.11, the accuracy of the elements' performances is heavily influenced by the variation of AR, as the trend of both elements exhibits a convex behavior. In a heuristic perspective, the thin-plate theory used by CLT can predict plate bending to a reasonable accuracy when the range of AR is between 10 to 100 [9, 11]. As the AR gets smaller, the transverse shear deformation can no longer be considered as negligible, and as the AR gets larger, the geometrically nonlinearity of the deformation means that the strain-displacement relationship from Equation 3.4 is no longer accurate. With that being said, it is still important to only consider Figure 5.11 as a qualitative study only. The reason is the deformation contribution from transverse shear will be dependent on laminate geometry, fiber layout, loading conditions, and transverse modulus. Hence with the currently numerical case studies, it is impossible to accurate parametrize the transverse effects. Ultimately, the key takeaway from this case study is that both elements can predict bending deformation to equal amount of accuracy when transverse shear is neglected. However, consideration of transverse shear would then require case by case analysis.

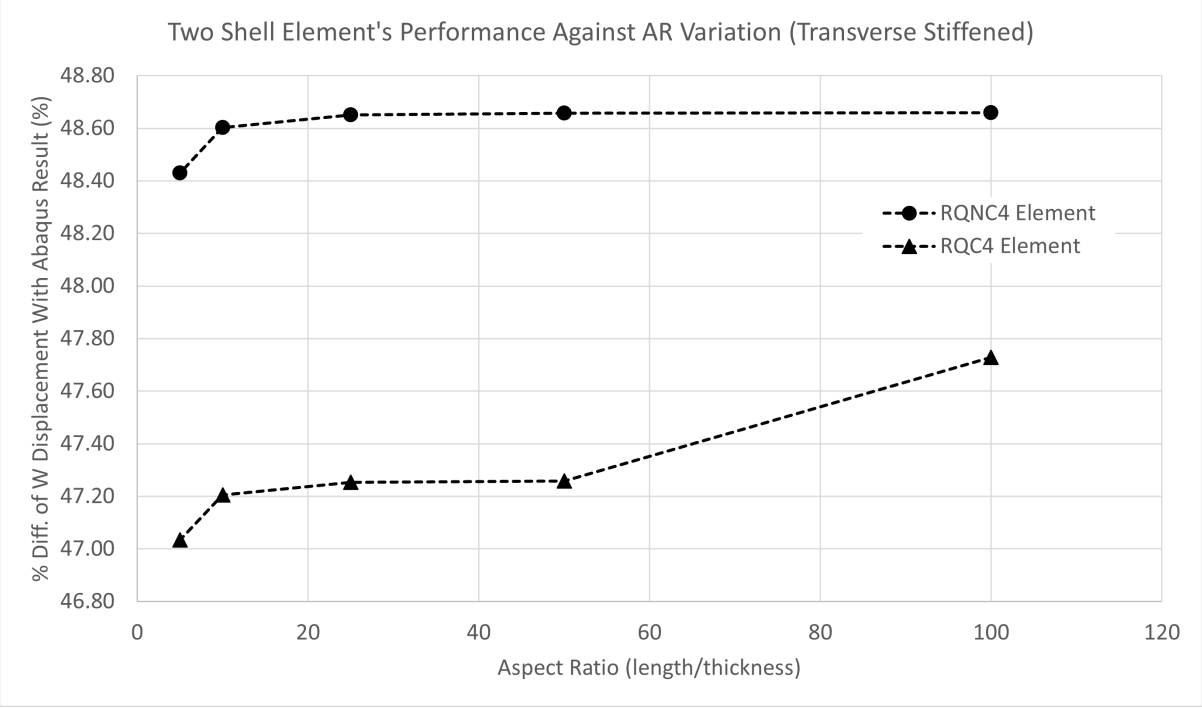

Figure 5.10: %Diff of midplane transverse deflection vs AR variation for transversely bounded square plate using both shell elements.

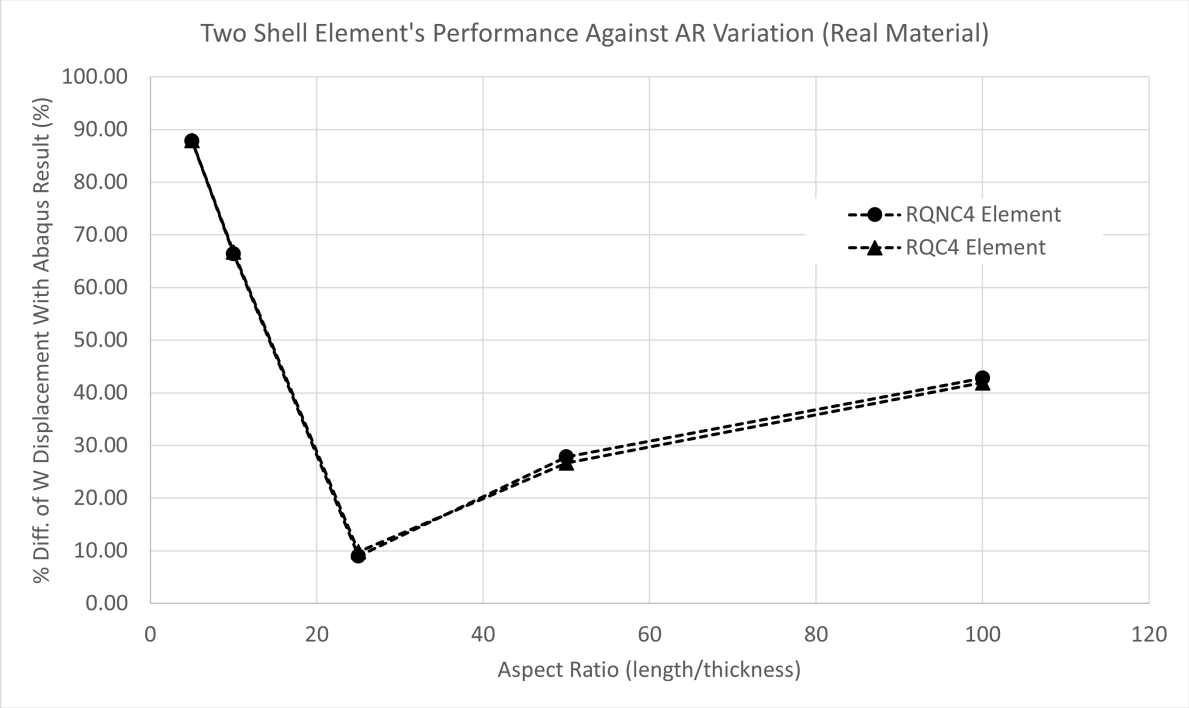

Figure 5.11: %Diff of midplane transverse deflection vs AR variation for transversely bounded square plate. ABAQUS model was done using realistic (not transversely stiffened) G_{13} and G_{23} values.

As mentioned in the Section 5.1, qualitative understanding of FEA results can also be very informing as it allows the incorporation of physical intuition when viewing the resultant data. Hence for the current square plate bending case, the representative illustrations of deformed mesh configuration, pseudo-reaction force contour, and transverse displacement contour map are shown in Figure 5.12 and Figure 5.13. The representative case was selected as the AR=5 case with a 20×20 mesh processed using RQNC4 element. Note that only the representative case is shown because like the mesh convergence analyses, the general behaviors of all the square plate cases are qualitatively similar. For the mesh deformation image shown in Figure 5.12, visually one can observe that the deformed plate obtains a smooth concave shape with the highest magnitude occurring at the center. The high smoothness of the individual contour lines show that the deformation mechanism of the shell element is not overly stiff. The pseudo-reaction force graph, as shown in Figure 5.13, was constructed by taking the F vector in the weak form (a vector of nodal values) and multiplying it with the same interpolation functions (shape functions) used by displacement approximation. The idea is that this graph would somewhat be representative of the reaction force experienced by the laminate plate. Evidently, the reaction forces are greatest on the corner of the plates, and also have non-negligible components distributed along the outer four edges. This is a reasonable

outcome considering the boundary conditions applied. The transverse displacement contour map shown in Figure 5.13 was constructed by multiplying the deformed w nodal d.o.f. with the element-wise shape functions. This graph essentially is a continuous and interpolated version of the mesh deformation map. The the smooth transition of the colour gradient in the displacement contour is consistent with the smooth contour lines seen in the mesh deformation plot.

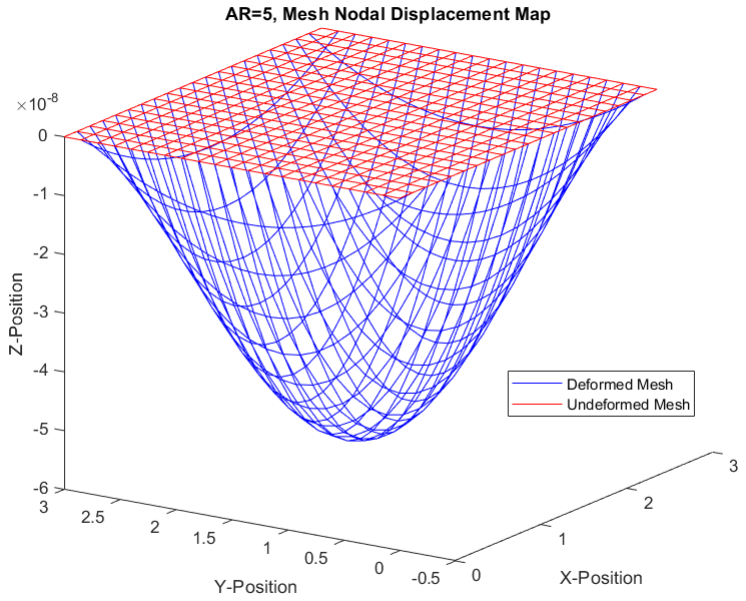

Figure 5.12: Representative mesh nodal deformation plot for the AR=5 square plate case, using the results from 20×20 mesh RQNC4 element.

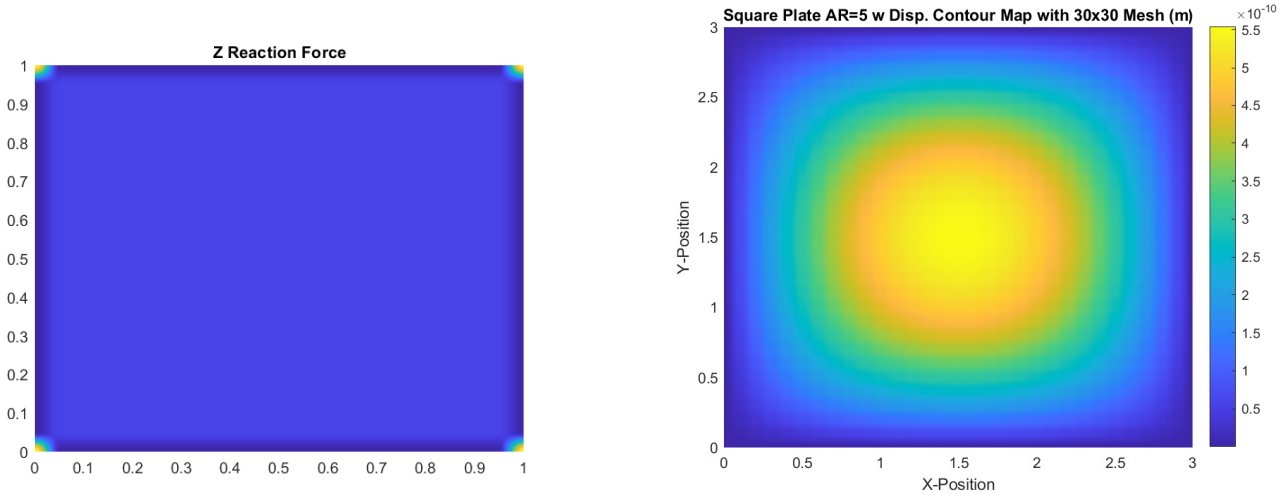

Figure 5.13: **Left**:Representative pseudo-reaction force map for the AR=5 square plate case, using the results from 20×20 mesh RQNC4 element. **Right**: Representative transverse displacement contour map for the AR=5 square plate case, using the results from 20×20 mesh RQNC4 element.

5.3.2 Cantilever Bending

The second bending deformation dominant case study is a cantilever structure clamped on one edge and applied with tip transverse shear loads on the opposite side, as shown in Figure 5.14. The detailed configurations and boundary conditions are shown in table below. In this load case, asymmetrical and angled fiber layouts are used to enable membrane-bending coupling, and in turn study the effect of that on the performance of the analysis suite. The variable of interest (var) for mesh convergence and verification study in this case was selected as the total magnitude of displacement taken from the corner of the loading edge, as shown in Figure 5.14.

Input Properties	Variable Values
in-plane dimensions	10 m by 2 m
material properties	$E_2 = 1 \times 10^8$ Pa $\quad E_1 = 25E_2 \quad G_{12} = 0.5E_2 \quad \nu_{12} = 0.25$ $\nu_{21} = \nu_{12}\frac{E_2}{E_1}$
	additional for ABAQUS: $G_{13} = G_{23} = 100E_1$
aspect ratio	AR = 10; 100
laminate layout (see Figure)	ply_symm = [0/90/90/0]
	ply_cross90 = [0/90/0/90]
	ply_cross45 = [+45/-45/+45/-45]
loading applied	$Q = 10.0\,\mathrm{N\,m^{-1}}$ at the free edge opposite of bounded edge.
boundary conditions	$u = 0$; $v = 0$; $w = 0$; $w_x = 0$; $w_y = 0$; $(w_{xy}) = 0$ at cantilever root.

Figure 5.14: Boundary conditions and applied loading for the case study of cantilever subjected to transverse shear load. The red arrows represent line of uniform transverse shear load and the triangular shapes on the boundary represent fully clamped boundary condition.

Cantilever Case - [0/90/90/0] Layout

The first simulated case had a symmetrical laminate layout of [0/90/90/0] as shown in Figure 5.15. This case serves as a baseline study since the midplane symmetry of the fiber angles will not result in membrane-bending coupling. The AR ratio considered for this case were 10 and 100, and representative graphs for mesh refinement behaviors for the two elements are shown in Figure 5.16 and Figure 5.17 for the case of AR=10. Since the deformation in this case is purely transverse, hence the RQNC4 element once again did not exhibit h-convergence with mesh refinement. For this load case, the RQC4 element also exhibited

negligible convergence behavior. Since the %Diff statistic as shown in Figure 5.17 are really small, hence this suggest that the current geometry and configurations resulted in a benign case that can be predicted with high accuracy by the MATLAB FEA with either one of the elements.

For visualization purposes, a representative displacment magnitude contour map alongside a representative deformed mesh configuration for an AR=10 case is shown in Figure 5.18. In general, both graphs exhibits the expected features of a cantilever bending.

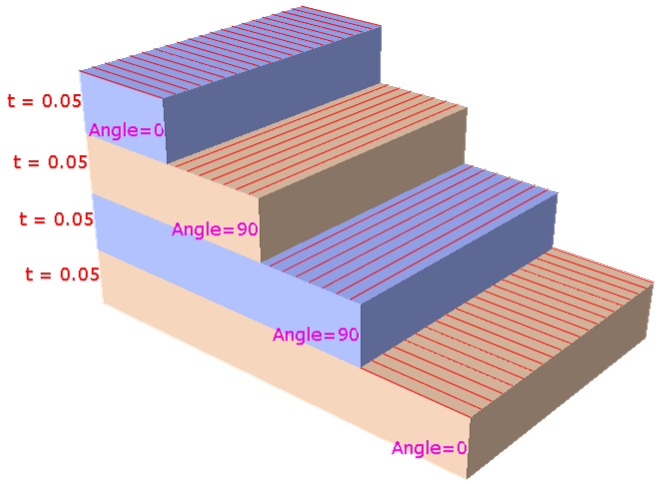

Figure 5.15: Laminate layout sequence [0/90/90/0] for the case study of cantilever in bending (plate thickness for this instance is 0.2 m).

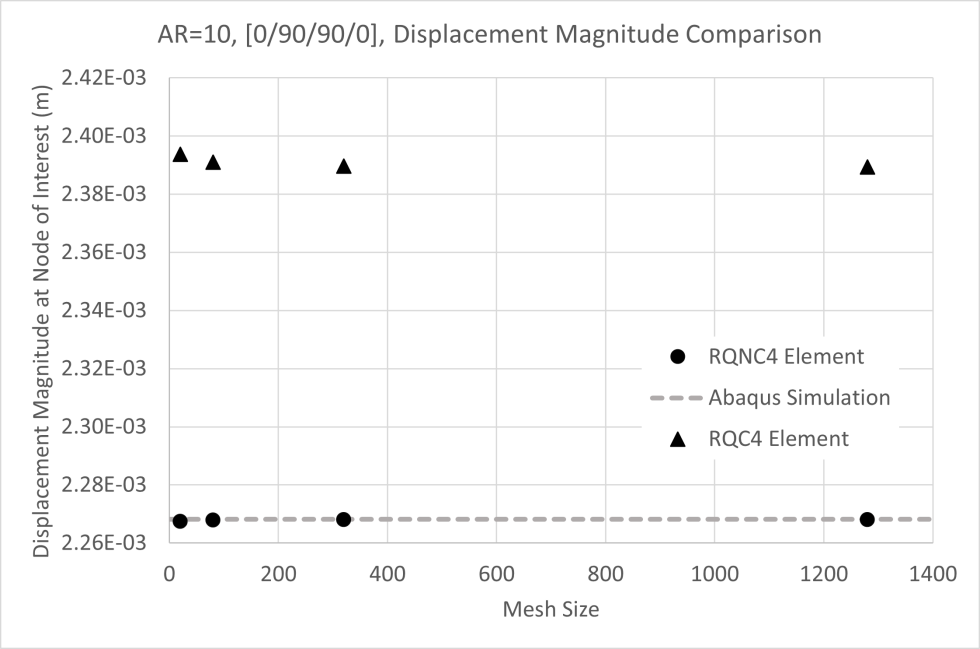

Figure 5.16: Representative mesh convergence behavior using total displacement magnitude at corner of cantilever for symmetrical uncoupled [0/90/90/0] layout case (AR=10).

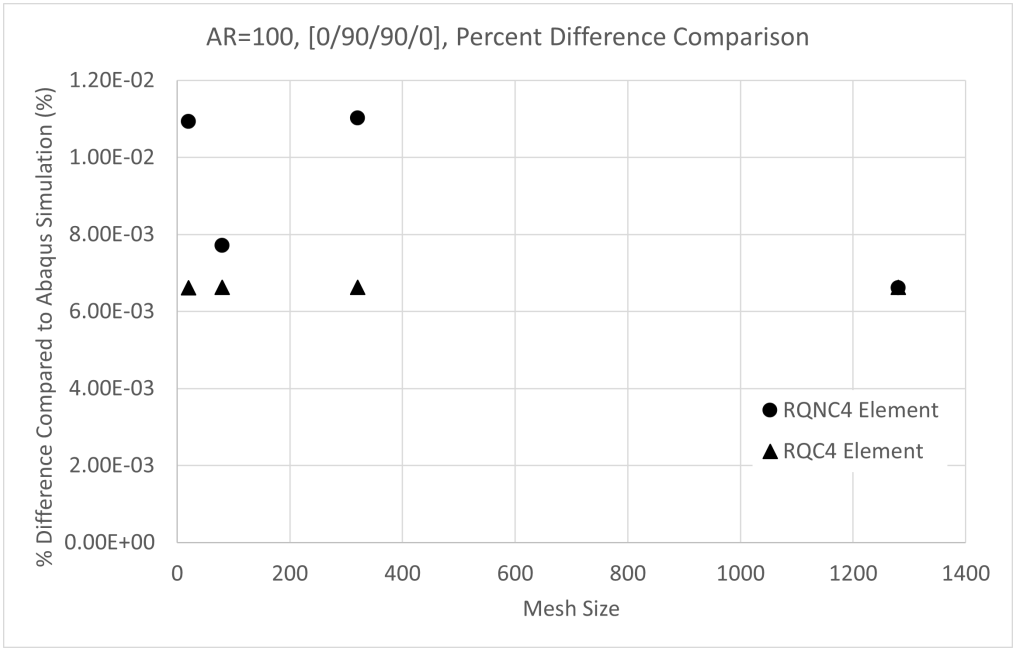

Figure 5.17: %Diff statistic over various mesh size from ABAQUS comparison of total displacement at corner of cantilever for symmetrical uncoupled [0/90/90/0] layout case (AR=10).

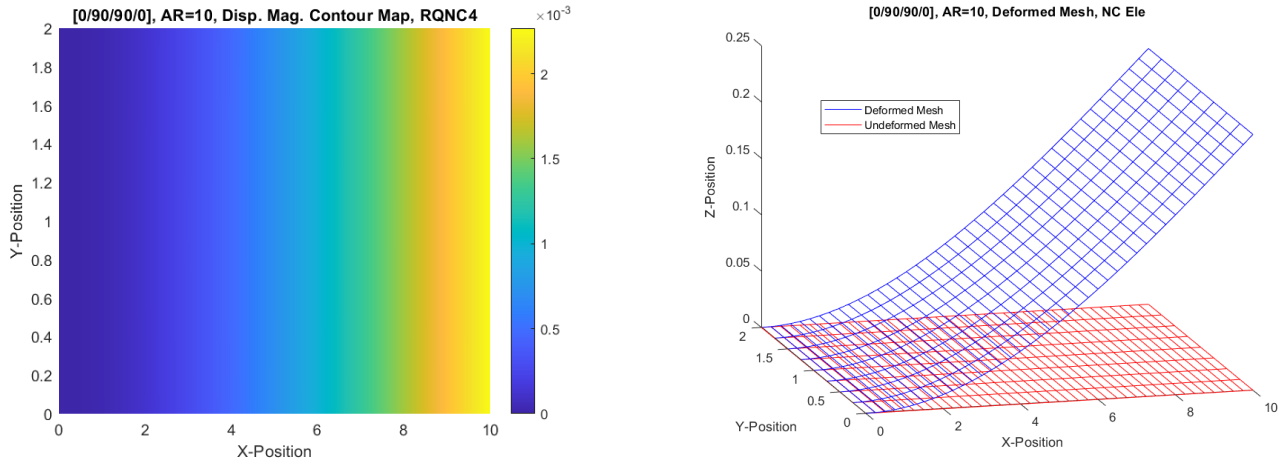

Figure 5.18: **Left**: Representative displacement magnitude contour map for [0/90/90/0] cantilever beam in bending using RQNC4 element. **Right**: Representative deformed mesh configuration for [0/90/90/0] cantilever beam in bending using RQNC4 element.

Cantilever Case - [0/90/0/90] Layout

The next cantilever case considered had an asymmetrical laminate layout of [0/90/0/90] as shown in Figure 5.19. The asymmetry about midplane resulted in membrane-bending coupled deformation, as the transverse direction loading resulted in slight in-plane deformations. This case was considered as the moderately membrane-bending coupled deformation case and the twist of the deformed configuration was not significant. Both AR=10 and AR=100 geometries were considered. Representative mesh convergence behaviors were shown in Figure 5.20 and Figure 5.21. Evidently from the graphs, both RQNC4 and RQC4 shell elements experienced convergence trend following mesh refinement. The coupling of in-plane and transverse deformations meant that the in-plane Q4 element was engaged. Since Q4 element satisfies the monotonic convergence criteria, both elements were expected to experience h-convergence features. From Figure 5.21, it can be observed that the %Diff statistic for both elements were similar and has a really low magnitude. This suggest that both elements can handle this asymmetric configuration really well and can generate accurate deformation predictions.

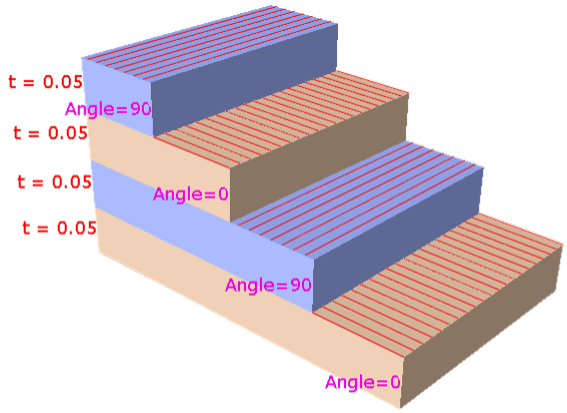

Figure 5.19: Laminate layout sequence [0/90/0/90] for the case study of cantilever in bending (plate thickness for this instance is 0.2 m).

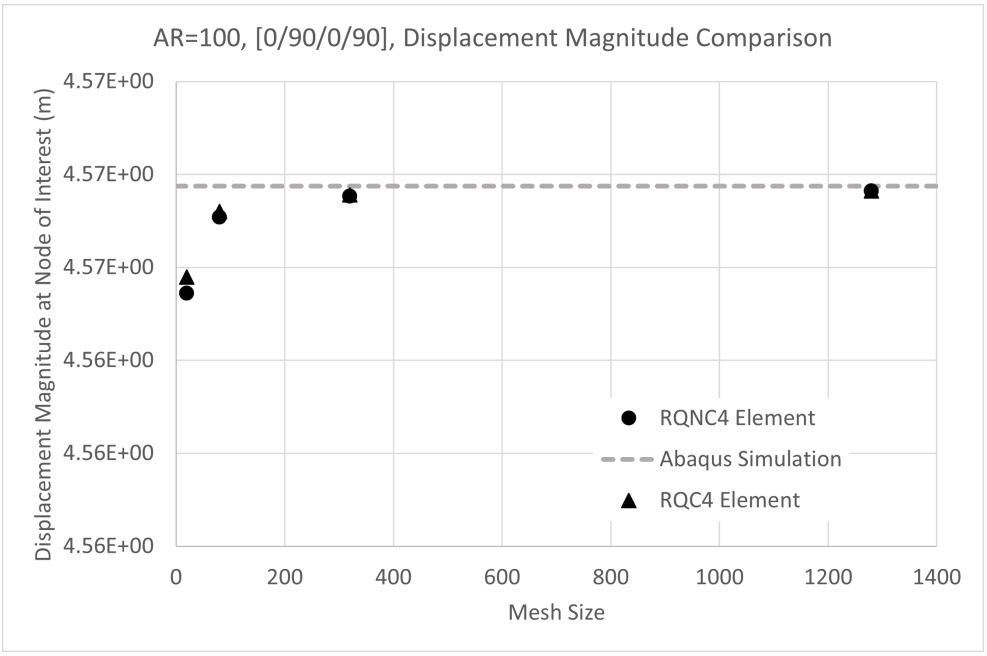

Figure 5.20: Representative mesh convergence behavior using total displacement magnitude at corner of cantilever for moderate membrane-bending coupling [0/90/0/90] layout case (AR=100).

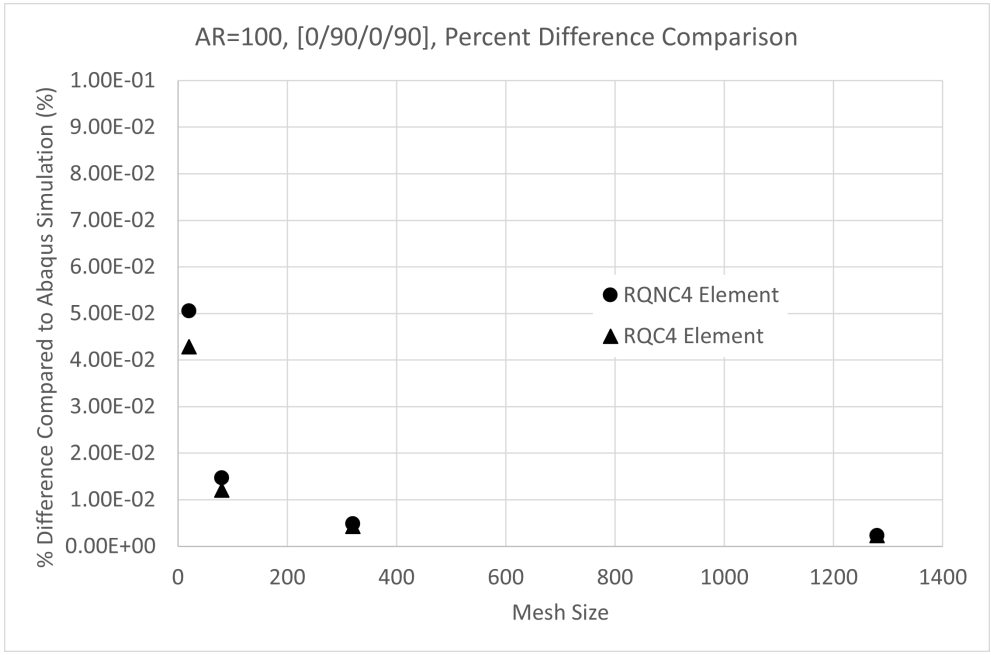

Figure 5.21: %Diff statistic over various mesh size from ABAQUS comparison of total displacement at corner of cantilever for moderate membrane-bending coupling [0/90/0/90] layout case (AR=100).

Cantilever Case - [+45/-45/+45/-45] Layout

The last bending cantilever case studied was the [+45/-45/+45/-45] layout, as shown in Figure 5.22, for both AR=10 and AR=100 geometries. Once again, the representative mesh convergence behavior of both shell elements are depicted in Figure 5.23 and Figure 5.24. Amongst the three cantilever configurations, this one lies on the significant membrane-bending coupling end of the spectrum, as the ratio between in-plane deformation component and transverse displacement considerably increased. This effect is demonstrated when comparing across Table 5.2, Table 5.3 , and Table 5.4. The engagement of the in-plane component resulted in h-convergence behavior as seen in Figure 5.24. Note this is more difficult to observe in Figure 5.23, since the large difference of MATLAB FEA and ABAQUS simulation stretched the y-axis scale, hence overwhelmed the convergence trends. The %Diff statistic for both elements in this case was at around 20%, which is a substantial increase compared to the previous 2 cantilever cases. Also, the poor accuracy was observed for both AR=10 and AR=100 case. This suggest that the accuracy of both elements reduces when the asymmetry level of the fiber layout increases, which would result in more significant twisting of the deformed laminate. For visualization purposes, representative deformed mesh configuration from ABAQUS simulation is shown in Figure 5.25, while the configurations from MATLAB FEA are shown in Figure 5.26 and Figure 5.27. With the aid of magnification, it is evident that the laminate experiences twisting deformation due to the bending-membrane coupled layout. The ABAQUS's image matches well in shape with MATLAB FEA's image, hence adding confidence to the correctness of the MATLAB FEA results.

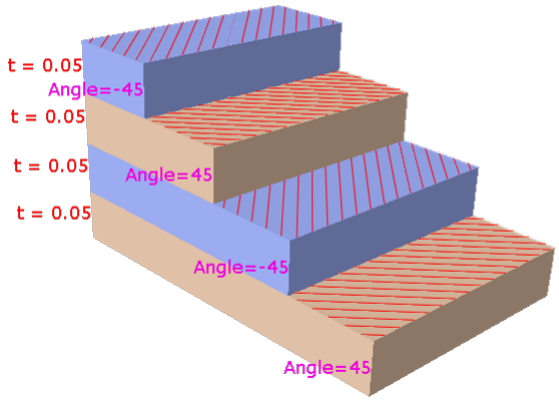

Figure 5.22: Laminate layout sequence [+45/-45/+45/-45] for the case study of cantilever in bending (plate thickness for this instance is 0.2 m).

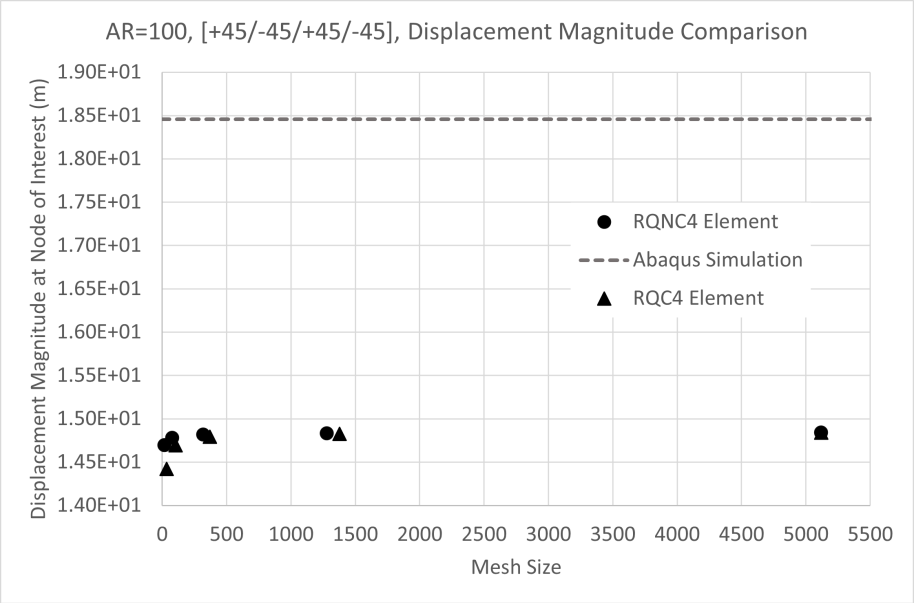

Figure 5.23: Representative mesh convergence behavior using total displacement magnitude at corner of cantilever for significant membrane-bending coupling [+45/-45/+45/-45] layout case (AR=10).

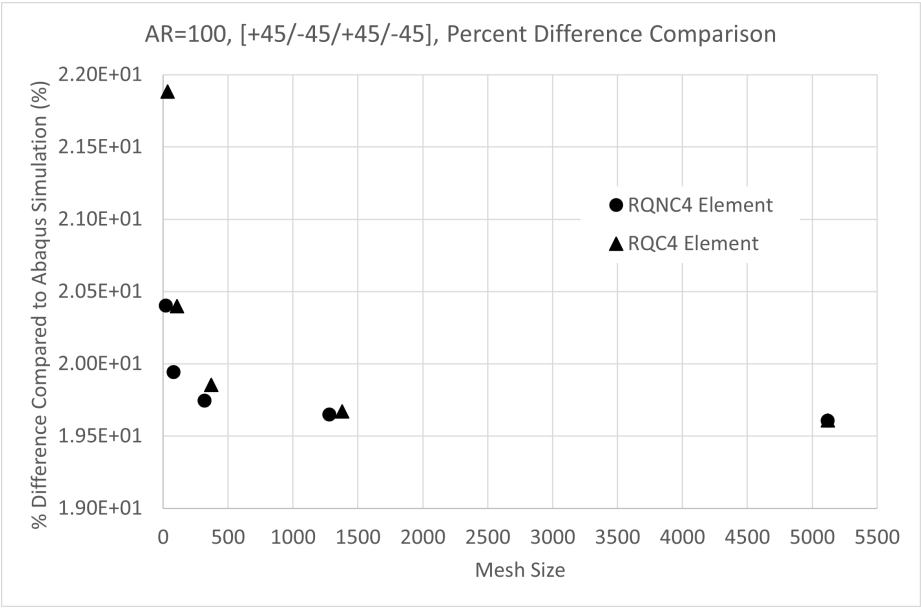

Figure 5.24: %Diff statistic over various mesh size from ABAQUS comparison of total displacement at corner of cantilever for significant membrane-bending coupling [+45/-45/+45/-45] layout case (AR=10).

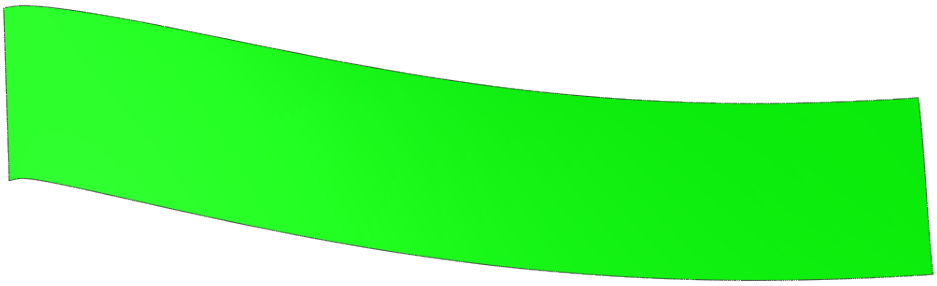

Figure 5.25: Top view of the representative deformed mesh configuration for [+45/-45/+45/-45] layout cantilever bending case, using ABAQUS S4 element simulation results. For visualization purposes, the in-plane displacements are magnified by scale of $3e4$ and transverse displacements are magnified by scale of 50.

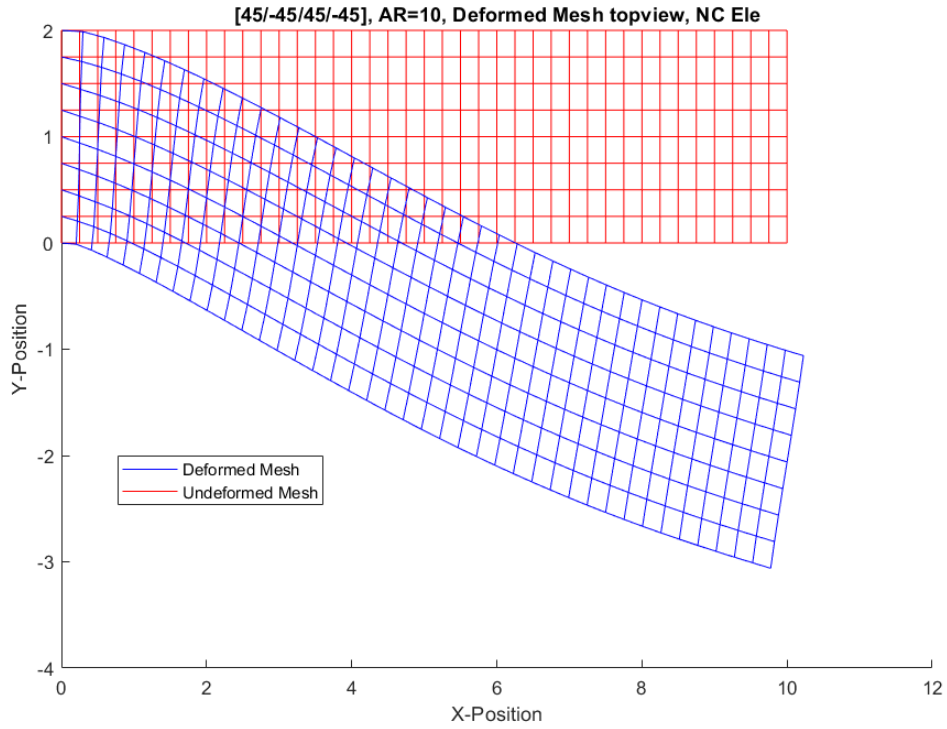

Figure 5.26: Top view of the representative deformed mesh configuration for [+45/-45/+45/-45] layout cantilever bending case, using RQNC4 element results. For visualization purposes, the in-plane displacements are magnified by scale of $3e4$ and transverse displacements are magnified by scale of 50.

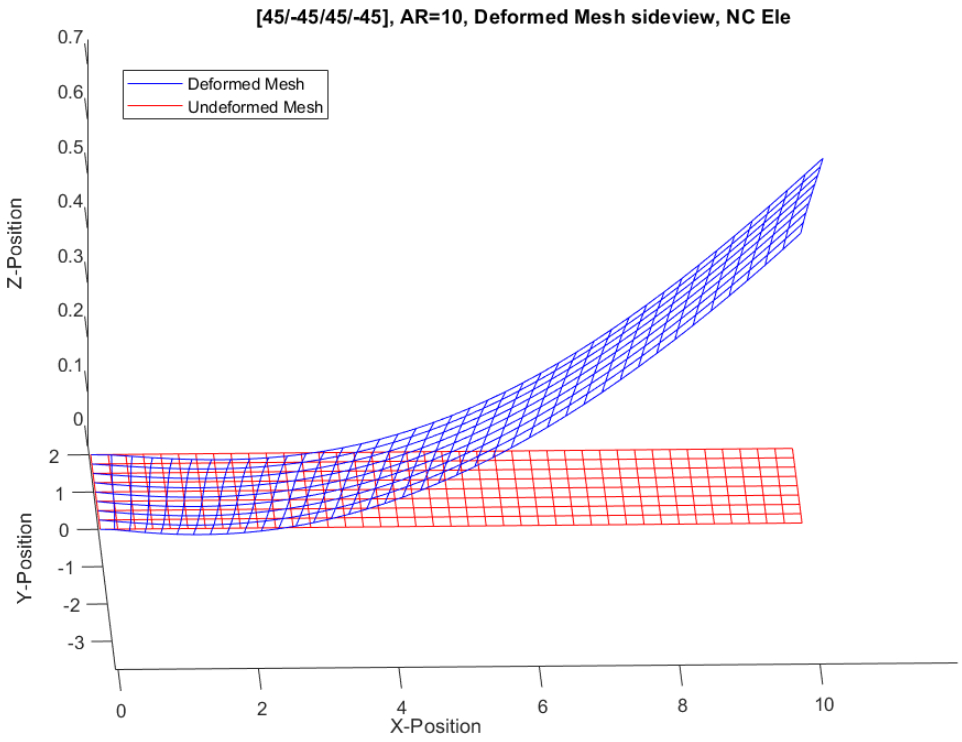

Figure 5.27: Side view of the representative deformed mesh configuration for [+45/-45/+45/-45] layout cantilever bending case, using RQNC4 element results. For visualization purposes, the in-plane displacements are magnified by scale of $3e4$ and transverse displacements are magnified by scale of 50.

Cantilever Cases - Summary of Data

The summary of mesh converged data for the case of square plate configurations is shown in Table 5.1. As mentioned, for this loading and geometry scenario, both RQNC4 and RQC4 produce results of similar accuracy compared to ABAQUS simulation. Also, the data shows that the performance of the elements was not influenced by the variation of AR at all. In addition, the RQNC4 element requires less nodal d.o.f. hence meaning that it has a faster computational speed. Since the RQNC4 element also does not exhibit convergence behavior with respect to mesh refinement, thus the result can be obtained using a small amount of discretization, which would require minimal computational resource.

The summary of mesh converged data for the case of cantilever bending configurations is shown in Table 5.2, Table 5.3, and Table 5.4. In general, the performance of both elements compared to ABAQUS simulation turns out to be similar once again. On the other hand, the accuracy of the elements were found to be influenced by the laminate layout. Both elements are capable of accurately predicting the [0/90/90/0] and [0/90/0/90] layouts, which results in minimum to moderate membrane-bending coupled deformations. However, for the [+45/-45/+45/-45] layout which resulted in significant bending induced twisting deformation, the accuracy of both elements deteriorated.

Table 5.1: Final Mesh Converged Values of Transverse Displacement at Midplate for the Case of Square Plate Under Distributed Surface Loading with Various AR.

Laminate Case	**AR**	**Solver and Mesh Size**	**Element**	**w(midplate) and %Diff**
square plate [0/90/0]	100	MATLAB 100×100	RQNC4	$w = -4.43 \times 10^{-6}$ m %Diff = 48.7%
square plate [0/90/0]	100	MATLAB 150×150	RQC4	$w = -4.41 \times 10^{-6}$ m %Diff = 47.7%
square plate [0/90/0]	100	ABAQUS 500×500	S4	$w = -2.98 \times 10^{-6}$ m
square plate [0/90/0]	50	MATLAB 100×100	RQNC4	$w = -5.54 \times 10^{-7}$ m %Diff = 48.7%
square plate [0/90/0]	50	MATLAB 100×100	RQC4	$w = -5.49 \times 10^{-7}$ m %Diff = 47.3%
square plate [0/90/0]	50	ABAQUS 500×500	S4	$w = -3.73 \times 10^{-7}$ m
square plate [0/90/0]	25	MATLAB 100×100	RQNC4	$w = -6.93 \times 10^{-8}$ m %Diff = 48.7%
square plate [0/90/0]	25	MATLAB 100×100	RQC4	$w = -6.86 \times 10^{-8}$ m %Diff = 47.3%
square plate [0/90/0]	25	ABAQUS 500×500	S4	$w = -4.66 \times 10^{-8}$ m
square plate [0/90/0]	10	MATLAB 100×100	RQNC4	$w = -4.43 \times 10^{-9}$ m %Diff = 48.6%
square plate [0/90/0]	10	MATLAB 100×100	RQC4	$w = -4.39 \times 10^{-9}$ m %Diff = 47.2%
square plate [0/90/0]	10	ABAQUS 500×500	S4	$w = -2.98 \times 10^{-9}$ m
square plate [0/90/0]	5	MATLAB 100×100	RQNC4	$w = -5.54 \times 10^{-10}$ m %Diff = 48.4%
square plate [0/90/0]	5	MATLAB 100×100	RQC4	$w = -5.49 \times 10^{-10}$ m %Diff = 47.0%
square plate [0/90/0]	5	ABAQUS 500×500	S4	$w = -3.73 \times 10^{-10}$ m

Table 5.2: Final Mesh Converged Values of Total Displacement Magnitude at Free Edge Corner the Case of Cantilever Beam Under Transverse Shear with [0/90/90/0] Layout.

Laminate Case	**AR**	**Solver and Mesh Size**	**Element**	**disp. mag. and %Diff**	**ratio of in-plane disp. vs trans-verse disp.** %
cantilever [0/90/90/0]	100	MATLAB, 1280	RQNC4	$mag = 2.2681\,\mathrm{m}$ %Diff = 0.0066%	0
cantilever [0/90/90/0]	100	MATLAB, 1280	RQC4	$mag = 2.2681\,\mathrm{m}$ %Diff = 0.0066%	0
cantilever [0/90/90/0]	100	ABAQUS, 5e4	S4	$mag = 2.2683\,\mathrm{m}$	0
cantilever [0/90/90/0]	10	MATLAB, 1280	RQNC4	mag=$2.2681 \times 10^{-3}\,\mathrm{m}$ %Diff = 7.8900×10^{-3} %	0
cantilever [0/90/90/0]	10	MATLAB, 1280	RQC4	$mag = 2.3983 \times 10^{-3}\,\mathrm{m}$ %Diff = 5.3361%	0
cantilever [0/90/90/0]	10	ABAQUS, 5e4	S4	$mag = 2.2683 \times 10^{-3}\,\mathrm{m}$	0

Table 5.3: Final Mesh Converged Values of Total Displacement Magnitude at Free Edge Corner the Case of Cantilever Beam Under Transverse Shear with [0/90/0/90] Layout

Laminate Case	**AR**	**Solver and Mesh Size**	**Element**	**disp. mag. and %Diff**	**ratio of in-plane disp. vs trans-verse disp.** %
cantilever [0/90/0/90]	100	MATLAB, 1280	RQNC4	$mag = 4.5677\,\mathrm{m}$ %Diff = 2.2956e−3%	5.8924e−2
cantilever [0/90/0/90]	100	MATLAB, 1280	RQC4	$mag = 4.5676\,\mathrm{m}$ %Diff = 2.3134e−3%	3.4615e−2
cantilever [0/90/90/0]	100	ABAQUS, 5e4	S4	$mag = 4.5678\,\mathrm{m}$	0
cantilever [0/90/0/90]	10	MATLAB, 1280	RQNC4	$mag = 4.5941 \times 10^{-3}\,\mathrm{m}$ %Diff = 5.7707e−1%	3.8657e−1
cantilever [0/90/0/90]	10	MATLAB, 5120	RQC4	$mag = 4.5677 \times 10^{-3}\,\mathrm{m}$ %Diff = 1.9015e−3%	3.4615e−1
cantilever [0/90/90/0]	10	ABAQUS, 5e4	S4	$mag = 4.5678 \times 10^{-3}\,\mathrm{m}$	3.6243e−1

Table 5.4: Final Mesh Converged Values of Total Displacement Magnitude at Free Edge Corner the Case of Cantilever Beam Under Transverse Shear with [+45/-45/+45/-45] Layout

Laminate Case	**AR**	**Solver and Mesh Size**	**Element**	**disp. mag. and %Diff**	**ratio of in-plane disp. vs trans-verse disp.** %
cantilever [45/-45/45/-45]	100	MATLAB, 5120	RQNC4	$mag = 14.840\,\mathrm{m}$ %Diff = 19.606%	6.7731e−2
cantilever [45/-45/45/-45]	100	MATLAB, 5120	RQC4	$mag = 14.839\,\mathrm{m}$ %Diff = 19.610%	6.7708e−2
cantilever [45/-45/45/-45]	100	ABAQUS, 5e4	S4	$mag = 18.459\,\mathrm{m}$	4.5460e−2
cantilever [45/-45/45/-45]	10	MATLAB, 5120	RQNC4	$mag = 1.4840 \times 10^{-2}\,\mathrm{m}$ %Diff = 19.710%	6.7731e−1
cantilever [45/-45/45/-45]	10	MATLAB, 5120	RQC4	$mag = 1.4839 \times 10^{-2}\,\mathrm{m}$ %Diff = 19.713%	6.7708e−1
cantilever [45/-45/45/-45]	10	ABAQUS, 5e4	S4	$mag = 1.8483 \times 10^{-2}\,\mathrm{m}$	3.2895e−1

Chapter 6

Conclusions and Future Work

Conclusions

The aim of this thesis was to rigorously develop a CLT based FEA analysis suite that will be used as a key building block to the formulation of a novel fiber layout design optimization method, which is the overarching goal of the project. To achieve this aim, several research objectives were proposed in the introduction and literature review sections, and are subsequently addressed in various parts of the thesis. These achieved objectives include:
1. Rigorous and thorough derivation of the strong form and weak form of FEA equations, starting from strain-displacement relationship and constitutive equation based on CLT theory. The process showcased in depth mathematical manipulations and element integration asepct, hence the final formulations can readily be implement into a computer program.
2. Implementation of FEA analysis suite in MATLAB incorporating two types of shell elements, RQC4 and RQNC4.
3. Verification study of the MATLAB FEA stiffness matrix was done using rigid body mode analyses. In summary, the FEA stiffness matrices corresponding to RQC4 or RQNC4, were all able to exhibit six rigid body modes, three translational and three rotational. This result added confidence to the correctness of the element-wise stiffness matrix computations and the global assembly processes.
4. Verification studies of the MATLAB FEA assembly were carried out using archetypal problems, specifically the square plate bending and cantilever bending load cases. The MATLAB FEA outputs were compared with ABAQUS simulation, which was treated as the ground truth for laminate deformation modeling. Both qualitative and quantitative tools were used to aid the verification process.

- For the square plate bending case, the plate was subjected to a distributed surface load

in addition to transversely bounded boundary conditions on outer edges. This case was analyzed repeatedly with varying plate AR value from 100 to 5, while maintaining a fiber layout of [0/90/0]. The quantitative comparison between MATLAB FEA and ABAQUS simulation was based on mid-plate transverse displacements. MATLAB FEA analysis using either of the elements produced a percentage difference value of near 50 % comparing to ABAQUS, which means that for this set of geometry and setup, the performance of MATLAB FEA is quite poor. Since the magnitudes of difference were consistent for all the AR values considered, hence this suggests that MATLAB FEA's ability to predict bending dominant transverse deformations is independent of geometry changes.

- For the cantilever bending case, the cantilever had a rectangular in-plane geometry, and was fully clamped on one edge, while a line of transverse shear load was applied to the opposite edge. This case was analyzed by mainly varying fiber layouts, including [0/90/90/0], [0/90/0/90], and [+45/-45/+45/-45] configurations. Two AR values, 10 and 100, were considered as a part of the analysis as well. The variable used for comparison between MATLAB FEA and ABAQUS was the total displacement magnitude at one corner of the edge with applied transverse shear loading. In general, for the [0/90/90/0] and [0/90/0/90] configurations, MATLAB FEA analysis using either of the elements resulted in percent difference of less than 1% for mass majorities of the trials considered, when compared to ABAQUS simulation. These two cases produced the lowest in-plane displacement component amongst the three configurations considered. The [+45/-45/+45/-45] configuration resulted in the highest ratio of in-plane displacement to the transverse displacement, and the comparison studies resulted in percent difference of around 20% for both MATLAB shell elements. Hence, the conclusion from the cantilever bending problem is that the MATLAB FEA suite can consistently predict deformations to a very high accuracy if the laminate layout configuration considered does not result in significant membrane-bending coupling. As the level of coupling increases, the performance of MATLAB FEA deteriorates.

5. Performance comparison between two shell elements (RQC4 and RQNC4) also done using the square plate and cantilever problems. The result allows that the performances of both elements in terms of accuracy were extremely similar for all investigated geometries and fiber pattern layouts. It can be concluded that in terms of the analysis capability, the choice of either element would not result in substantial differences. On the other hand, the RQNC4 element has less nodal degree of freedom compared to RQC4 element, and hence requires less computational resources. Hence in terms of obtaining results of similar accuracy

while minimizing computational intensities, the RQNC4 element is the optimal choice for all scenarios.

Future Work

One possible area for future investigation would be to expand the FEA analysis suite by considering other low-order and simple to implement models. The inherent nature of the CLT based FEA is limited in capability in terms of lamiante plate geometry. Specifically, for sufficiently small laminate length to thickness ratio, the transverse shear effect on deformation would no longer be negligible, hence the main assumptions employed by CLT will no longer be accurate. For this case, the FSDT theory, which is another low-order simplistic model can be considered. FSDT is constructed based on thick-plate theory, hence it has the ability to reasonably predict transverse shear deformations. On the other end of the spectrum, when the length to thickness ratio becomes really large, laminate bending would then result in high orders of geometrical non-linearity. As a result, the small strain theory based strain-displacement relationship used for the current FEA derivation will no longer be valid. A possible simple method to account for this scenario is by considering von-Karman strain based strain-displacement relationship, which incorporates the high order strain-displacecment terms to account for non-linearity. Having both of the mentioned improvements would theoretically increase the capability of the analysis suite, and hence allowing a wider range of analysis choices depending on problem scenarios.

Another area of investigation is considering the procedures required to expand this analysis suite to more complex geometries such as curvilinear fiber layouts. Recall that the ultimate objectives of this laminate analysis suite is to aid the process of fiber pattern design optimization. In problems with high complexity in loading or geometry, the theoretically optimized fiber layout would often not be linear for all the ply layers. FEA as a discretization based method has the natural advantage in terms of accurately approximating non-constant material properties in the problem domain. Hence, expanding FEA to account for continuous fiber distributions should be able to progress naturally, and the final product would greatly enhance the ability of the analysis suite.

Bibliography

[1] G. Allaire and G. Delgado. "Stacking sequence and shape optimization of laminated composite plates via a level-set method". In: *Journal of the Mechanics and Physics of Solids* 97 (2016). SI:Pierre Suquet Symposium, pp. 168 –196. ISSN: 0022-5096. DOI: https://doi.org/10.1016/j.jmps.2016.06.014. URL: http://www.sciencedirect.com/science/article/pii/S002250961630429X.

[2] Anonymous. *About shell elements*. Online. URL: https://abaqus-docs.mit.edu/2017/English/SIMACAEELMRefMap/simaelm-c-shelloverview.htm#:~:text=Shell%20elements%20are%20used%20to,geometry%20at%20a%20reference%20surface.

[3] anonymous. *Small Scale Strains*. Ed. by Continuum Mechanics. 2021. URL: http://www.continuummechanics.org/smallstrain.html.

[4] Babak Bigdeli. "An Investigation of C*-Convergence in the Finite Element Method". PhD thesis. The University of New South Wales, 1996.

[5] M. Bruyneel. "Optimization of laminated composite structures: problems, solution procedures and applications". In: 2008. URL: https://www.semanticscholar.org/paper/Optimization-of-laminated-composite-structures%3A-and-Bruyneel/73a92f0963b8ea60b95afe7472f2fdb9ccf81a19#citing-papers.

[6] Eleni Chatzi and Patrick Steffen. *Chapter 3 Variational Formulation and the Galerkin Method*. URL: https://ethz.ch/content/dam/ethz/special-interest/baug/ibk/structural-mechanics-dam/education/femI/lecture3.pdf.

[7] Jin Young Choi and Mark Timothy Kortschot. "Stiffness prediction of 3D printed fiber-reinforced thermoplastic composites". In: *Rapid Prototyping Journal* 26.3 (Feb. 2021), pp. 549–555. DOI: https://doi.org/10.1108/RPJ-11-2018-0283. URL: https://www.emerald.com/insight/content/doi/10.1108/RPJ-11-2018-0283/full/html?skipTracking=true.

[8] J. Cichosz. "Experimental Characterization and Numerical Modeling of the Mechanical Response for Biaxial Braided Composites". In: 2016. URL: https://www.semanticscholar.org/paper/Experimental-Characterization-and-Numerical-of-the-Cichosz/9d5c789b4b1840ca388fa9d025af7d1eab25e227.

[9] Fehmi Cirak. *Plates and Shells: Theory and Computation*. URL: http://www-g.eng.cam.ac.uk/csml/teaching.html.

[10] Matthew Crossan. "Mechanical Characterization and Shear Test Comparison for Continuous-Fiber Polymer Composites". MA thesis. The University of Western Ontario, Apr. 2018. URL: https://ir.lib.uwo.ca/cgi/viewcontent.cgi?article=7333&context=etd.

[11] S. Triantafyllou Eleni Chatzi Patrick Steffen. *Chapter 6 2D Elements*. URL: https://ethz.ch/content/dam/ethz/special-interest/baug/ibk/structural-mechanics-dam/education/femI/lecture6.pdf.

[12] Rafael Ferreira et al. "Optimisation of Fibre-Paths in Composites Produced by Additive Manufacturing". In: Jan. 2019, pp. 1083–1094. ISBN: 978-3-319-97772-0. DOI: 10.1007/978-3-319-97773-7_94.

[13] Roselita Fragoudakis. "Strengths and Limitations of Traditional Theoretical Approaches to FRP Laminate Design against Failure". In: *Engineering Failure Analysis*. Ed. by Kary Thanapalan. IntechOpen, Oct. 2019. Chap. 1. URL: https://www.intechopen.com/books/engineering-failure-analysis/strengths-and-limitations-of-traditional-theoretical-approaches-to-frp-laminate-design-against-failu.

[14] Y. M. Ghugal and R. P. Shimpi. "A Review of Refined Shear Deformation Theories of Isotropic and Anisotropic Laminated Plates". In: *Journal of Reinforced Plastics and Composites* 21.9 (2002), pp. 775–813. DOI: 10.1177/073168402128988481. URL: https://doi.org/10.1177/073168402128988481.

[15] Dane Haystead. "Determining the Optimal Orientation of Orthotropic Material for Maximizing Frequency Band Gaps". MA thesis. University of Toronto, 2012. URL: https://arrow.utias.utoronto.ca/~csteeves/Theses/Haystead_UTIAS_MASc_2012.pdf.

[16] A. Idlbi, M. Karama, and M. Touratier. "Comparison of various laminated plate theories". In: *Composite Structures* 37.2 (1997), pp. 173 –184. ISSN: 0263-8223. DOI: https://doi.org/10.1016/S0263-8223(97)80010-4. URL: http://www.sciencedirect.com/science/article/pii/S0263822397800104.

[17] Rasoul Khandan et al. “The development of laminated composite plate theories: A review”. In: *Journal of Materials Science* 47 (Aug. 2012), pp. 5901–5910. DOI: 10.1007/s10853-012-6329-y.

[18] Byung Chul Kim, Kevin Potter, and Paul M. Weaver. “Continuous tow shearing for manufacturing variable angle tow composites”. In: *Composites Part A: Applied Science and Manufacturing* 43.8 (2012), pp. 1347 –1356. ISSN: 1359-835X. DOI: https://doi.org/10.1016/j.compositesa.2012.02.024. URL: http://www.sciencedirect.com/science/article/pii/S1359835X12000929.

[19] Tae-Uk Kim and In Hee Hwang. “Optimal design of composite wing subjected to gust loads”. In: *Computers and Structures* 83.19 (2005), pp. 1546 –1554. ISSN: 0045-7949. DOI: https://doi.org/10.1016/j.compstruc.2005.02.002. URL: http://www.sciencedirect.com/science/article/pii/S0045794905000957.

[20] A. Laulusa et al. “Evaluation of some shear deformable shell elements”. In: *International Journal of Solids and Structures* 43.17 (2006), pp. 5033–5054. ISSN: 0020-7683. DOI: https://doi.org/10.1016/j.ijsolstr.2005.08.006. URL: https://www.sciencedirect.com/science/article/pii/S0020768305005238.

[21] Myung-Ha Lee. “Finite Element Analysis of Laminated Composite Plates”. MA thesis. Monterey, California: Naval Postgraduate School, Sept. 1988. URL: https://apps.dtic.mil/dtic/tr/fulltext/u2/a202192.pdf.

[22] M.F.N. Mohsen. “Some details of the Galerkin finite element method”. In: *Applied Mathematical Modelling* 6.3 (1982), pp. 165–170. ISSN: 0307-904X. DOI: https://doi.org/10.1016/0307-904X(82)90005-1. URL: https://www.sciencedirect.com/science/article/pii/0307904X82900051.

[23] Prasanth Nair. *Theoretical and Convergence Aspects of Finite Element Methods*.

[24] M Nurhaniza et al. “Finite element analysis of composites materials for aerospace applications”. In: *IOP Conference Series: Materials Science and Engineering* 11 (May 2010), p. 012010. DOI: 10.1088/1757-899x/11/1/012010. URL: https://doi.org/10.1088/1757-899x/11/1/012010.

[25] Bazeley G. P. et al. “Triangular Elements in Plate Bending - Conforming and Non-Conforming Solutions”. In: Nov. 1966, pp. 547–576. URL: http://contrails.iit.edu/items/show/8575.

[26] Brett Arnold Pauer. "Development of a Finite Element Method Program for the Analysis of Laminated Composite Plates Using First-Order Shear Deformation Theory". MA thesis. Columbus, Ohio: The Ohio State University, Aug. 1993. URL: https://apps.dtic.mil/dtic/tr/fulltext/u2/a270318.pdf.

[27] J. Reddy and D. Robbins. "THEORIES AND COMPUTATIONAL MODELS FOR COMPOSITE LAMINATES". In: *Applied Mechanics Reviews* 47.6 (June 1994), pp. 147–169. DOI: https://doi.org/10.1115/1.3111076. URL: https://www.semanticscholar.org/paper/THEORIES-AND-COMPUTATIONAL-MODELS-FOR-COMPOSITE-Reddy-Robbins/2ec1e1b866aba2569d2dd6915097ea13e1d54058.

[28] J. N. Reddy. "A Simple Higher-Order Theory for Laminated Composite Plates". In: *Journal of Applied Mechanics* 51.4 (Dec. 1984), pp. 745–752. ISSN: 0021-8936. DOI: 10.1115/1.3167719. eprint: https://asmedigitalcollection.asme.org/appliedmechanics/article-pdf/51/4/745/5457622/745_1.pdf. URL: https://doi.org/10.1115/1.3167719.

[29] J. N. Reddy. *Mechanics of Laminate Composite Plates and Shells*. 2nd ed. Boca Raton: CRC Press, 2003. DOI: https://doi.org/10.1201/b12409.

[30] Pedro Ribeiro et al. "A review on the mechanical behaviour of curvilinear fibre composite laminated panels". In: *Journal of Composite Materials* 48.22 (2014), pp. 2761–2777. DOI: 10.1177/0021998313502066. URL: https://doi.org/10.1177/0021998313502066.

[31] Klaus Rohwer, Stefan Friedrichs, and Christof Wehmeyer. "Analyzing Laminated Structures from Fibre-Reinforced Composite Material - An Assessment". In: *Technische Mechanik* 25 (Mar. 2005), pp. 59–77. URL: https://www.researchgate.net/publication/224778901_Analyzing_Laminated_Structures_from_Fibre-Reinforced_Composite_Material_-_An_Assessment.

[32] David Roylance. *LAMINATED COMPOSITE PLATES*. Feb. 2000.

[33] Patrick Safarian. *Finite Element Modeling and Analysis Validation*. URL: https://appliedcax.com/docs/femap/femap-symposium-2015-seattle-area/FEA-Validation-Requiremnents-and-Methods-Final-with-Transcript.pdf.

[34] Andreas Schranzinger. "Modeling of Long-Fiber-Reinforced Composites in ABAQUS". MA thesis. GRAZ UNIVERSITY OF TECHNOLOGY, Apr. 2018. URL: https://diglib.tugraz.at/download.php?id=58dc3334ef60f&location=browse.

[35] Marco Di Sciuva and Ugo Icardi. "Discrete-layer models for multilayered shells accounting for interlayer continuity". In: *Meccanica* 28.4 (Dec. 1993), pp. 281–291. DOI: https://doi.org/10.1007/BF00987164. URL: https://link.springer.com/article/10.1007/BF00987164#citeas.

[36] Zhong-Ci Shi. "Nonconforming finite element methods". In: *Journal of Computational and Applied Mathematics* 149.1 (2002), pp. 221–225. ISSN: 0377-0427. DOI: https://doi.org/10.1016/S0377-0427(02)00531-9. URL: https://www.sciencedirect.com/science/article/pii/S0377042702005319.

[37] René Sørensen and Erik Lund. "Ply-based Optimization of Laminated Composite Shell Structures under Manufacturing Constraints". In: June 2012. URL: https://www.researchgate.net/publication/283479808_Ply-based_Optimization_of_Laminated_Composite_Shell_Structures_under_Manufacturing_Constraints.

[38] S. Srinivas. "A refined analysis of composite laminates". In: *Journal of Sound and Vibration* 30.4 (1973), pp. 495 –507. ISSN: 0022-460X. DOI: https://doi.org/10.1016/S0022-460X(73)80170-1. URL: http://www.sciencedirect.com/science/article/pii/S0022460X73801701.

[39] Charles R. Steele and Chad D. Balch. *Introduction to the Theory of Plates*. Apr. 2009. URL: https://web.stanford.edu/~chasst/Course%20Notes/Introduction%20to%20the%20Theory%20of%20Plates.pdf.

[40] Craig Steeves. *Lecture notes in AER1403*.

[41] Truong Tich Thien. *FEM Convergence Requirements*. URL: http://www4.hcmut.edu.vn/~tttruong/IFEM.Ch19.Slides.pdf.

[42] R. Lipton V. B. Hammer M. P. Bendsoe and P. Pedersen. "Parametrization in laminate design for optimal compliance". In: *International Journal of Solids and Structures* (1997). URL: https://orbit.dtu.dk/en/publications/parametrization-in-laminate-design-for-optimal-compliance.

[43] Jack R. Vinson. "Plate and Panel Structures of Isotropic, Composite and Piezoelectric Materials, Including Sandwich Construction". In: vol. 120. Springer, Dordrecht, 1990. DOI: https://doi.org/10.1007/1-4020-3111-4. URL: https://link.springer.com/book/10.1007/1-4020-3111-4#about.

[44] *What Are Composites?* Online. 2021. URL: http://compositeslab.com/composites-101/what-are-composites/.

[45] Brice Matthew Willis. "COMPOSITE MODELING CAPABILITIES OF COMMERCIAL FINITE ELEMENT SOFTWARE". In: (2012). URL: https://core.ac.uk/download/pdf/159610535.pdf.

[46] Qi Xia and Tielin Shi. "Optimization of composite structures with continuous spatial variation of fiber angle through Shepard interpolation". In: *Composite Structures* 182 (Sept. 2017). DOI: 10.1016/j.compstruct.2017.09.052. URL: https://www.researchgate.net/publication/319974610_Optimization_of_composite_structures_with_continuous_spatial_variation_of_fiber_angle_through_Shepard_interpolation.

[47] Yusuke Yamanaka et al. "Fiber Line Optimization in Single Ply for 3D Printed Composites". In: *Open Journal of Composite Materials* 06 (Jan. 2016), pp. 121–131. DOI: 10.4236/ojcm.2016.64012. URL: https://www.researchgate.net/publication/309100430_Fiber_Line_Optimization_in_Single_Ply_for_3D_Printed_Composites.

[48] Qian Zhang. "Efficient and High-fidelity Finite Element Models for Fibre Composites". PhD thesis. University of Toronto, 2019. URL: https://tspace.library.utoronto.ca/bitstream/1807/95978/3/Zhang_Qian_201906_PhD_thesis.pdf.

Appendices

Appendix A

MATLAB FEA Code Module

Maindriver and Sample Inputs

```
%% Square Plate Bending - RQNC4 Element
clear all
clc

xdim = 3;
ydim = 3;

%mesh size 2x2: Ny = 2; Nx = 2;
%mesh size 6x6: Ny = 4; Nx = 4;
%mesh size 20x20: Ny = 20; Nx = 20;
%mesh size 30x30:
Ny = 30; Nx = 30;
%mesh size 100x100: Ny = 100; Nx = 100;

BC.pos = {{[0,xdim],ydim};
    {[0,xdim],0};
    {xdim,[0,ydim]};
    {0,[0,ydim]};
    {[0,xdim],[0,ydim]}};

BC.type = {{'w','wx','wy'};
    {'w','wx','wy'};
    {'w','wx','wy'};
    {'w','wx','wy'};
    {'q'}};

BC.val = {{0,0,0};
    {0,0,0};
    {0,0,0};
    {0,0,0};
    {-5}};

%ply: [G12,E1,E2,v12,v21,theta,h]
%case1a): AR = 1; L/T = 100
% ply = [3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.01;
%     3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,pi/2,0.01;
%     3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.01];

%case1b): AR = 1; L/T = 50
% ply = [3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.02;
%     3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,pi/2,0.02;
%     3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.02];

%case1c): AR = 1; L/T = 25
% ply = [3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.04;
%     3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,pi/2,0.04;
%     3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.04];
```

```
%case1d): AR = 1; L/T = 5
ply = [3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.2;
    3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,pi/2,0.2;
    3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.2];

%case1d): AR = 1; L/T = 10
% ply = [3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.1;
%     3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,pi/2,0.1;
%     3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.1];

[connec,nodecoord] = meshing(xdim,ydim,Ny,Nx);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Cantilever Test Case - RQNC4 Element
clear all
clc
xdim = 10; ydim = 2;
%mesh size 10x2: Ny = 2; Nx = 10;
%mesh size 20x4: Ny = 4; Nx = 20;
%mesh size 40x8:
Ny = 8; Nx = 40;
%mesh size 80x16: Ny = 16; Nx = 80;
%mesh size 160x32: Ny = 32; Nx = 160;

BC.pos = {{xdim,[0,ydim]};
    {0,[0,ydim]}};

BC.type = {{'Q'};
    {'w','wx','x','y','wy'}};

BC.val = {{10};
    {0,0,0,0,0}};

E2 = 10^8;
%AR = 100 - [0/90/90/0]
% ply = [0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),0,0.005;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/2,0.005;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/2,0.005;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),0,0.005];

% %AR = 10 - [0/90/90/0]
ply = [0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),0,0.05;
    0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/2,0.05;
    0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/2,0.05;
    0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),0,0.05];

%AR = 100 - [0/90/0/90]
% ply = [0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),0,0.005;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/2,0.005;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),0,0.005;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/2,0.005];

% %AR = 10 - [0/90/0/90]
% ply = [0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),0,0.05;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/2,0.05;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),0,0.05;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/2,0.05];

% %AR = 10 - [45/-45/45/-45]
% ply = [0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/4,0.05;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),-pi/4,0.05;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/4,0.05;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),-pi/4,0.05];


% %AR = 100 - [45/-45/45/-45]
% ply = [0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/4,0.005;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),-pi/4,0.005;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/4,0.005;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),-pi/4,0.005];

[connec,nodecoord] = meshing(xdim,ydim,Ny,Nx);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Nonconforming RBM tests

clear all
clc
xdim = 1; ydim = 1;
%Nx = 1; Ny = 1; %1x1 mesh
Nx = 2; Ny = 2; %2x2 mesh
ply = [4.8*10^9,230*10^9,230*10^9,0.25,0.25,0,0.005];
[connec,nodecoord] = meshing(xdim,ydim,Ny,Nx);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Maindrive for Nonconforming element
ktot = sparse(5*max(nodecoord(:,1)),5*max(nodecoord(:,1))); %5 dof per node
p = 2;
for i = 1:size(connec,1) %loop over elements
    x_e = nodecoord(connec(i,2:5),2);
    y_e = nodecoord(connec(i,2:5),3);
    [KA,KB,KC,KD] = elelaminatestiff_noncon(p,x_e,y_e,ply);

    rowpos1 = [connec(i,2)*2-1,connec(i,2)*2,connec(i,3)*2-1,connec(i,3)*2,connec(i,4)*2-1,connec(i,4)*2,connec(i,5)*2-1,connec(i,5)*2];
    rowpos2 = [connec(i,2)*3-2,connec(i,2)*3-1,connec(i,2)*3,connec(i,3)*3-2,connec(i,3)*3-1,connec(i,3)*3,...
            connec(i,4)*3-2,connec(i,4)*3-1,connec(i,4)*3,connec(i,5)*3-2,connec(i,5)*3-1,connec(i,5)*3]+ 2*max(nodecoord(:,1));
    for j = 1:8 %loop over the first 8 columns, populate rows
        colpos1 = (connec(i,ceil(j/2)+1)*2-mod(j,2))*ones(1,8);
        colpos2 = (connec(i,ceil(j/2)+1)*2-mod(j,2))*ones(1,12);
        tol = max(eps(ktot(rowpos1,colpos1(1))),eps(KA(:,j)));
        ktot = ktot + sparse(rowpos1,colpos1,KA(:,j),5*max(nodecoord(:,1)),5*max(nodecoord(:,1)));
        ktot(rowpos1(abs(ktot(rowpos1,colpos1(1)))<=100*tol),colpos1(1)) = 0;
        tol = max(eps(ktot(rowpos2,colpos2(1))),eps(KC(:,j)));
        ktot = ktot + sparse(rowpos2,colpos2,KC(:,j),5*max(nodecoord(:,1)),5*max(nodecoord(:,1)));
        ktot(rowpos2(abs(ktot(rowpos2,colpos2(1)))<=100*tol),colpos2(1)) = 0;
    end
    for j = 1:12 % loop over the last 12 columns
        colpos1 = (2*max(nodecoord(:,1))+connec(i,ceil(j/3)+1)*3-((ceil(j/3)-floor(j/3))*3-mod(j,3)))*ones(1,8);
        colpos2 = (2*max(nodecoord(:,1))+connec(i,ceil(j/3)+1)*3-((ceil(j/3)-floor(j/3))*3-mod(j,3)))*ones(1,12);
        tol = max(eps(ktot(rowpos1,colpos1(1))),eps(KB(:,j)));
        ktot = ktot + sparse(rowpos1,colpos1,KB(:,j),5*max(nodecoord(:,1)),5*max(nodecoord(:,1)));
        ktot(rowpos1(abs(ktot(rowpos1,colpos1(1)))<=100*tol),colpos1(1)) = 0;
        tol = max(eps(ktot(rowpos2,colpos2(1))),eps(KD(:,j)));
        ktot = ktot + sparse(rowpos2,colpos2,KD(:,j),5*max(nodecoord(:,1)),5*max(nodecoord(:,1)));
        ktot(rowpos2(abs(ktot(rowpos2,colpos2(1)))<=100*tol),colpos2(1)) = 0;
    end
end

[d,d_dof,f] = solve_noncon(connec,nodecoord,BC,ktot,p);
ele = 1;
graphing(connec,nodecoord,d,ele,f)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Rigid body mode analysis for nonconforming element - 1x1 mesh
%Note the number of integration point for the Gauss quadrature needs %to be 3 for the eigenvalues to not be imaginary.

K = full(ktot);
xmov = [repmat([1,0],1,4),zeros(1,20-8)];
ymov = [repmat([0,1],1,4),zeros(1,20-8)];
zmov = [zeros(1,8),repmat([1,0,0],1,4)];
f = zeros(length(zmov),1);

theta = 1;
zrot = [0,0,-15*sind(theta),-15+15*cosd(theta),...
    -(xdim-xdim*cosd(theta)),xdim*sind(theta),...
    -(xdim-(xdim*cosd(theta)-xdim*sind(theta))), xdim*sind(theta)+xdim*cosd(theta)-xdim,...
    zeros(1,12)];

[q,v] = eigs(K,6,"smallestab");

kk = q(1:8,4:6);
t1 = repmat([0,1],1,4);
t2 = repmat([1,0],1,4);
```

```
s1 = kk\t1';
s2 = kk\t2';
s3 = null([s1,s2]');
c3 = kk*s3;
graphing(connec,nodecoord,[c3./10^7.5;zeros(12,1)],1,f);

kz = q(9:end,1:3);
w1 = repmat([1,0,0],1,4);
ss1 = kz\w1';
sz = null([ss1]');
cz = kz*sz;
rx = cz*null(cz(2,:));
ry = cz*null(cz(3,:));

graphing(connec,nodecoord,[zeros(8,1);ry./10^2],1,f);
graphing(connec,nodecoord,[zeros(8,1);rx./10^2],1,f);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Rigid body mode analysis for nonconforming element - 2x2 mesh

%Note the number of integration point for the Gauss quadrature needs %to be 3 for the eigenvalues to not be imaginary.

K = full(ktot);
xmov = [repmat([1,0],1,9),zeros(1,45-18)];
ymov = [repmat([0,1],1,9),zeros(1,45-18)];
zmov = [zeros(1,18),repmat([1,0,0],1,9)];
f = zeros(length(zmov),1);
theta = 1;
zrot = [0,0,-15*sind(theta),-15+15*cosd(theta),...
    -(xdim-xdim*cosd(theta)),xdim*sind(theta),...
    -(xdim-(xdim*cosd(theta)-xdim*sind(theta))), xdim*sind(theta)+xdim*cosd(theta)-xdim,...
    zeros(1,12)];

[q,v] = eigs(K,6,"smallestab");

kk = q(1:18,4:6);
t1 = repmat([0,1],1,9);
t2 = repmat([1,0],1,9);
s1 = kk\t1';
s2 = kk\t2';
s3 = null([s1,s2]');
c3 = kk*s3;
graphing(connec,nodecoord,[c3./10^7.5;zeros(45-18,1)],1,f);

kz = q(19:end,1:3);
w1 = repmat([1,0,0],1,9);
ss1 = kz\w1';
sz = null([ss1]');
cz = kz*sz;
rx = cz*null(cz(2,:));
ry = cz*null(cz(3,:));

graphing(connec,nodecoord,[zeros(18,1);ry./10^2],1,f);
graphing(connec,nodecoord,[zeros(18,1);rx./10^2],1,f);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Square Plate Case - RQC4 Element Inputs
clear all
clc
xdim = 3; ydim = 3;
%mesh size 2x2: Ny = 2; Nx = 2;
%mesh size 6x6: Ny = 6; Nx = 6;
%mesh size 20x20: Ny = 20; Nx = 20;
%mesh size 30x30:
Ny = 30; Nx = 30;
%mesh size 100x100: Ny = 100; Nx = 100;
%mesh size 150x150: Ny = 150; Nx = 150;

BC.pos = {{[0,xdim],ydim};
```

```
    {[0,xdim],0};
    {xdim,[0,ydim]};
    {0,[0,ydim]};
    {[0,xdim],[0,ydim]}};

BC.type = {{'w','wx','wy','wxy'};
    {'w','wx','wy','wxy'};
    {'w','wx','wy','wxy'};
    {'w','wx','wy','wxy'};
    {'q'}};

BC.val = {{0,0,0,0};
    {0,0,0,0};
    {0,0,0,0};
    {0,0,0,0};
    {-5}};

%ply: [G12,E1,E2,v12,v21,theta,h]
%case1a): AR = 1; L/T = 100
% ply = [3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.01;
%     3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,pi/2,0.01;
%     3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.01];

%case1b): AR = 1; L/T = 50
% ply = [3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.02;
%     3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,pi/2,0.02;
%     3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.02];

%case1c): AR = 1; L/T = 25
% ply = [3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.04;
%     3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,pi/2,0.04;
%     3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.04];

%case1d): AR = 1; L/T = 5
ply = [3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.2;
    3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,pi/2,0.2;
    3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.2];

%case1d): AR = 1; L/T = 10
% ply = [3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.1;
%     3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,pi/2,0.1;
%     3447.38*10^6,172369*10^6,6894.76*10^6,0.25,6894.76*10^6*0.25/172369*10^6,0,0.1];

[connec,nodecoord] = meshing(xdim,ydim,Ny,Nx);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Cantilever Test Case - RQC4 Element Inputs

clear all
clc
xdim = 10; ydim = 2;

%mesh size 10x2:
Ny = 2; Nx = 10;
%mesh size 20x4: Ny = 4; Nx = 20;
%mesh size 40x8: Ny = 8; Nx = 40;
%mesh size 80x16: Ny = 16; Nx = 80;
%mesh size 160x32: Ny = 32; Nx = 160;

BC.pos = {{xdim,[0,ydim]};
    {0,[0,ydim]}};

BC.type = {{'Q'};
    {'w','wx','x','y','wy','wxy'}};

BC.val = {{10};
    {0,0,0,0,0,0}};

E2 = 10^8;
%AR = 100 [0/90/90/0]
% ply = [0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),0,0.005;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/2,0.005;
```

```
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/2,0.005;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),0,0.005];


% %AR = 10 [0/90/90/0]
% ply = [0.5*E2,25*E2,E2,0.25,E2*0.25/(0.5*E2),0,0.05;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(0.5*E2),pi/2,0.05;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(0.5*E2),pi/2,0.05;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(0.5*E2),0,0.05];

%AR = 100 -[0/90/0/90]
% ply = [0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),0,0.005;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/2,0.005;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),0,0.005;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/2,0.005];

% %AR = 10 - [0/90/0/90]
% ply = [0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),0,0.05;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/2,0.05;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),0,0.05;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/2,0.05];

% %AR = 10 - [45/-45/45/-45]
% ply = [0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/4,0.05;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),-pi/4,0.05;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/4,0.05;
%     0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),-pi/4,0.05];

% %AR = 100 - [45/-45/45/-45]
ply = [0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/4,0.005;
    0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),-pi/4,0.005;
    0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),pi/4,0.005;
    0.5*E2,25*E2,E2,0.25,E2*0.25/(25*E2),-pi/4,0.005];

[connec,nodecoord] = meshing(xdim,ydim,Ny,Nx);

%% RIGID BODY MOTION Inputs

clear all
clc
xdim = 1; ydim = 1;
Nx = 1; Ny = 1;
%Nx = 2; Ny = 2;
ply = [4.8*10^9,230*10^9,230*10^9,0.25,0.25,0,0.005];
%ply = [2*10^9,230*10^9,230*10^9,0.25,0.25,0,0.005];
[connec,nodecoord] = meshing(xdim,ydim,Ny,Nx);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Maindriver for Conforming element case

ktot = sparse(6*max(nodecoord(:,1)),6*max(nodecoord(:,1))); %6 dof per node
p = 3;
for i = 1:size(connec,1) %loop over elements
    x_e = nodecoord(connec(i,2:5),2);
    y_e = nodecoord(connec(i,2:5),3);
    [KA,KB,KC,KD] = elelaminatestiff_con(p,x_e,y_e,ply);

    rowpos1 = [connec(i,2)*2-1,connec(i,2)*2,connec(i,3)*2-1,connec(i,3)*2,connec(i,4)*2-1,connec(i,4)*2,connec(i,5)*2-1,connec(i,5)*2];
    rowpos2 = [connec(i,2)*4-3,connec(i,2)*4-2,connec(i,2)*4-1,connec(i,2)*4,connec(i,3)*4-3,connec(i,3)*4-2,connec(i,3)*4-1,connec(i,3)*4,...
            connec(i,4)*4-3,connec(i,4)*4-2,connec(i,4)*4-1,connec(i,4)*4,connec(i,5)*4-3,connec(i,5)*4-2,connec(i,5)*4-1,connec(i,5)*4]+ 2*max(nodecoord(:,1));
    for j = 1:8 %loop over the first 8 columns, populate rows
        colpos1 = (connec(i,ceil(j/2)+1)*2-mod(j,2))*ones(1,8);
        colpos2 = (connec(i,ceil(j/2)+1)*2-mod(j,2))*ones(1,16);

        tol = max(eps(ktot(rowpos1,colpos1(1))),eps(KA(:,j)));
        ktot = ktot + sparse(rowpos1,colpos1,KA(:,j),6*max(nodecoord(:,1)),6*max(nodecoord(:,1)));
        ktot(rowpos1(abs(ktot(rowpos1,colpos1(1)))<=100*tol),colpos1(1)) = 0;
        tol = max(eps(ktot(rowpos2,colpos2(1))),eps(KC(:,j)));
        ktot = ktot + sparse(rowpos2,colpos2,KC(:,j),6*max(nodecoord(:,1)),6*max(nodecoord(:,1)));
        ktot(rowpos2(abs(ktot(rowpos2,colpos2(1)))<=100*tol),colpos2(1)) = 0;
    end
    for j = 1:16 % loop over the last 16 columns
        colpos1 = (2*max(nodecoord(:,1))+connec(i,ceil(j/4)+1)*4-((ceil(j/4)-floor(j/4))*4-mod(j,4)))*ones(1,8);
```

```
        colpos2 = (2*max(nodecoord(:,1))+connec(i,ceil(j/4)+1)*4-((ceil(j/4)-floor(j/4))*4-mod(j,4)))*ones(1,16);
        tol = max(eps(ktot(rowpos1,colpos1(1))),eps(KB(:,j)));
        ktot = ktot + sparse(rowpos1,colpos1,KB(:,j),6*max(nodecoord(:,1)),6*max(nodecoord(:,1)));
        ktot(rowpos1(abs(ktot(rowpos1,colpos1(1)))<=100*tol),colpos1(1)) = 0;
        tol = max(eps(ktot(rowpos2,colpos2(1))),eps(KD(:,j)));
        ktot = ktot + sparse(rowpos2,colpos2,KD(:,j),6*max(nodecoord(:,1)),6*max(nodecoord(:,1)));
        ktot(rowpos2(abs(ktot(rowpos2,colpos2(1)))<=100*tol),colpos2(1)) = 0;
    end
end

[d,d_dof,f] = solve_con(connec,nodecoord,BC,ktot,p);
ele = 2;
graphing(connec,nodecoord,d,ele,f)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Rigid body mode analysis for conforming element - 1x1 mesh

K = full(ktot);
%1x1
xmov = [repmat([1,0],1,4),zeros(1,24-8)];
ymov = [repmat([0,1],1,4),zeros(1,24-8)];
zmov = [zeros(1,8),repmat([1,0,0,0],1,4)];
f = zeros(length(zmov),1);
[q,v] = eigs(K,7,'smallestab');
%qt = [q(:,3:5),q(:,1:2),q(:,6)];
qt = [q(:,1:3),q(:,4:6)];

kk = qt(1:8,4:6);
t1 = [1,0,1,0,1,0,1,0];
t2 = [0,1,0,1,0,1,0,1];
s1 = kk\t1';
s2 = kk\t2';
s3 = null([s1,s2]');
c3 = kk*s3;
graphing(connec,nodecoord,[c3./10^7.5;zeros(16,1)],2,f);

pos1 = [0,0]' + c3(1:2);
pos2 = [0,ydim]' + c3(3:4);
pos3 = [xdim,0]' + c3(5:6);
pos4 = [xdim,ydim]' + c3(7:8);
l1 = norm(pos2-pos1);
l2 = norm(pos3-pos1);
ll1 = norm(pos4-pos3);
ll2 = norm(pos4-pos2);

kz = qt(9:24,1:3);
w1 = [1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0];
ss1 = kz\w1';
sz = null([ss1]');
cz = kz*sz;
rx = cz*null(cz(2,:));
ry = cz*null(cz(3,:));


%1x1 mesh, p=2, "hourglassing/zero energy RBM mode":
% [q,v] = eigs(K,7,'smallestab');
% qt = [q(:,1:3),q(:,4:7)];
% kz = qt(9:24,1:4);
% w1 = [1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0];
% ss1 = kz\w1';
% sz = null([ss1]');
% cz = kz*sz;
% rx = cz*null(cz(1:2,:));
% ry = cz*null(cz(2:3,:));
% rxy_ss1 = null([cz\rx,cz\ry]');
% rxy = cz*rxy_ss1;
%after this, run the troubleshooting_wxy.m code to get the "RBM shape" and
%the corresponding strain values at guass points which shows that all the
%strain values are near zero, thus making this a zero energy mode

graphing(connec,nodecoord,[zeros(8,1);rx./10^2],2,f);
graphing(connec,nodecoord,[zeros(8,1);ry./10^2],2,f);
```

```
theta = 5;
yrot = [zeros(1,4),-xdim*(1-cosd(theta)),0,-xdim*(1-cosd(theta)),0,zeros(1,6),1,0,0,1,0,0];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Rigid body mode analysis for conforming element - 2x2 mesh

K = full(ktot);
xmov = [repmat([1,0],1,9),zeros(1,54-18)];
ymov = [repmat([0,1],1,9),zeros(1,54-18)];
zmov = [zeros(1,18),repmat([1,0,0,0],1,9)];
f = zeros(length(zmov),1);

[q,v] = eigs(K,6,'smallestab');
%qt = [q(:,3:5),q(:,1:2),q(:,6)];
qt = [q(:,1:3),q(:,4:6)];

kk = qt(1:18,4:6);
t1 = [repmat([1,0],1,9)];
t2 = [repmat([0,1],1,9)];
s1 = kk\t1';
s2 = kk\t2';
s3 = null([s1,s2]');
c3 = kk*s3;
graphing(connec,nodecoord,[c3;zeros(54-18,1)],2,f);

kz = qt(19:end,1:3);
w1 = [repmat([1,0,0,0],1,9)];
ss1 = kz\w1';
sz = null([ss1]');
cz = kz*sz;
rx = cz*null(cz(2,:));
ry = cz*null(cz(3,:));
%rxy_ss1 = null([cz\rx,cz\ry]');
%rxy = cz*rxy_ss1;
graphing(connec,nodecoord,[zeros(18,1);rx],2,f);
graphing(connec,nodecoord,[zeros(18,1);ry],2,f);
```

meshing function

```
function [connec,nodecoord] = meshing(xdim,ydim,Ny,Nx)

%input:
%1. xdim: total x length (m)
%2. ydim: total y length (m)
%3. Ny: number of rectangular elements in y direction
%4. Nx: number of rectangular elements in x direction

%output:
%1. connec: connectivity matrix -> %[ele #, node1, node2, node3, node4]
%2. nodecoord: nodal coordinates matrix -> %[node #, x coord, y coord, z coord]

xlen = xdim/Nx;
ylen = ydim/Ny;

nodecoord = zeros((Ny+1)*(Nx+1),4); %[node #, x coord, y coord, z coord]
connec = zeros(Ny*Nx,5); %[ele #, node1, node2, node3, node4]

for i = 1:Nx+1 %loop over the number of "columns" (x division lines)
    nodecoord((i-1)*(Ny+1)+1:i*(Ny+1),1) = linspace((i-1)*(Ny+1)+1,i*(Ny+1),Ny+1); %coord number
    nodecoord((i-1)*(Ny+1)+1:i*(Ny+1),2) = (i-1)*xlen; %x pos
    nodecoord((i-1)*(Ny+1)+1:i*(Ny+1),3) = linspace(0,Ny,Ny+1)*ylen; %y pos
    nodecoord((i-1)*(Ny+1)+1:i*(Ny+1),4) = zeros(Ny+1,1); %z pos

    if i > 1 %assign 4 nodes to each rectangular element
        connec((i-2)*Ny+1:(i-1)*Ny,1) = linspace((i-2)*Ny+1,(i-1)*Ny,Ny);
        connec((i-2)*Ny+1:(i-1)*Ny,2) = nodecoord((i-2)*(Ny+1)+1:1:(i-1)*(Ny+1)-1,1);
        connec((i-2)*Ny+1:(i-1)*Ny,3) = nodecoord((i-1)*(Ny+1)+1:1:(i)*(Ny+1)-1,1);
        connec((i-2)*Ny+1:(i-1)*Ny,4) = nodecoord((i-1)*(Ny+1)+2:1:(i)*(Ny+1),1);
```

```
        connec((i-2)*Ny+1:(i-1)*Ny,5) = nodecoord((i-2)*(Ny+1)+2:1:(i-1)*(Ny+1),1);
    end
end
% %
figure,
%plotting the mesh
for i = 1:size(connec,1)
    plot([nodecoord(connec(i,2),2),nodecoord(connec(i,3),2)],[nodecoord(connec(i,2),3),nodecoord(connec(i,3),3)],'b');
    hold on,
    plot([nodecoord(connec(i,3),2),nodecoord(connec(i,4),2)],[nodecoord(connec(i,3),3),nodecoord(connec(i,4),3)],'b');
    hold on,
    plot([nodecoord(connec(i,4),2),nodecoord(connec(i,5),2)],[nodecoord(connec(i,4),3),nodecoord(connec(i,5),3)],'b');
    hold on,
    plot([nodecoord(connec(i,2),2),nodecoord(connec(i,5),2)],[nodecoord(connec(i,2),3),nodecoord(connec(i,5),3)],'b');
    hold on,
    text(0.5*(nodecoord(connec(i,2),2)+nodecoord(connec(i,3),2)),0.5*(nodecoord(connec(i,2),3)+nodecoord(connec(i,4),3)),int2str(connec(i,1)),'Color','red','FontSize',10)
    hold on,
    text(nodecoord(connec(i,2),2),nodecoord(connec(i,2),3),int2str(nodecoord(connec(i,2),1)),'FontSize',7);
    hold on,
    text(nodecoord(connec(i,3),2),nodecoord(connec(i,3),3),int2str(nodecoord(connec(i,3),1)),'FontSize',7);
    hold on,
    text(nodecoord(connec(i,4),2),nodecoord(connec(i,4),3),int2str(nodecoord(connec(i,4),1)),'FontSize',7);
    hold on,
    text(nodecoord(connec(i,5),2),nodecoord(connec(i,5),3),int2str(nodecoord(connec(i,5),1)),'FontSize',7);
    hold on,
end
end
```

elelaminatestiff_noncon function

```
function [KA,KB,KC,KD] = elelaminatestiff_noncon(p,x_e,y_e,ply)
%this function computes the complete [A,B;B,D] laminate stiffness
%matrix with multiplication with the shape functions and the appropriate
%derivatives

%ply: [G12,E1,E2,v12,v21,theta,h]
plynum = size(ply,1);
%independent material constants:
G12 = ply(:,1); E1 = ply(:,2); E2 = ply(:,3); v12 = ply(:,4); v21 = ply(:,5);
theta = ply(:,6); h = ply(:,7);
%note: v12, v21, E1, E2 are coupled: v21 = E2.*v12./E1

%material compliance matrix for plies
for i = 1:plynum
    S_o(:,:,i) = [1/E1(i),-v21(i)/E2(i),0;-v12(i)/E1(i),1/E2(i),0;0,0,1/G12(i)].*10^10;
    %D_o(:,:,i) =[E1(i)/(1-v12(i)*v21(i)),v12(i)*E2(i)/(1-v12(i)*v21(i)),0;v12(i)*E2(i)/(1-v12(i)*v21(i)),E2(i)/(1-v12(i)*v21(i)),0;0,0,G12(i)];
end

%set up variables for finding the modulus-weighted midplane
E0 = max(E1); %reference modulus for midplane calculation
hstar = sum((E1./E0).*h);
z_hstar = 0;

%transformation of compliance matrix to principal directions for the
%laminate:
R = [1,0,0;0,1,0;0,0,2]; %Reuter's matrix
for i = 1:plynum
    c2 = cosd(theta(i)*(180/pi))^2;
    s2 = sind(theta(i)*(180/pi))^2;
    sc = sind(theta(i)*(180/pi))*cosd(theta(i)*(180/pi));
    At(:,:,i) = [c2,s2,2*sc;s2,c2,-2*sc;-sc,sc,c2-s2]; %stress rotation matrix
    %strain rotation matrix would be RAR^-1
    S_bar = R*inv(At(:,:,i))*inv(R)*S_o(:,:,i)*At(:,:,i); %transformed compliance matrix
    D_bar(:,:,i) = inv(S_bar./10^10); %transformed stiffness matrix
    z_hstar = z_hstar + E1(i)/E0*h(i)*(sum(h(1:i))*2-h(i))/2;
end

%zstar is the location of the modulus weighted midplane with respect to the
%reference coordinate system where the bottom of the laminate is defined as
%the z = 0 plane.
```

```
zstar = z_hstar/hstar;

%forming the laminate stiffness matrix components
A = zeros(size(D_bar(:,:,1)));
B = zeros(size(D_bar(:,:,1)));
D = zeros(size(D_bar(:,:,1)));
for i = 1:plynum
    tol = max(eps(A),eps(D_bar(:,:,i)*((sum(h(1:i))-zstar)-(sum(h(1:i))-h(i)-zstar))));
    A = A+D_bar(:,:,i)*((sum(h(1:i))-zstar)-(sum(h(1:i))-h(i)-zstar));
    A(abs(A)<=100*tol) = 0;
    tol = max(eps(B),eps(D_bar(:,:,i)*((sum(h(1:i))-zstar)^2-(sum(h(1:i))-h(i)-zstar)^2)));
    B = B+0.5*D_bar(:,:,i)*((sum(h(1:i))-zstar)^2-(sum(h(1:i))-h(i)-zstar)^2);
    B(abs(B)<=100*tol) = 0;
    tol = max(eps(D),eps(D_bar(:,:,i)*((sum(h(1:i))-zstar)^3-(sum(h(1:i))-h(i)-zstar)^3)));
    D = D+(1/3)*D_bar(:,:,i)*((sum(h(1:i))-zstar)^3-(sum(h(1:i))-h(i)-zstar)^3);
    D(abs(D) <= 100*tol) = 0;
end

area = abs((x_e(3) - x_e(1)) * (y_e(3) - y_e(1))); % area of element (m^2)

% initialize a few variables
% total stiffness matrix for the element (segmented to 4 sub-matrices)
KA = zeros(8,8);
KB = zeros(8,12);
KC = zeros(12,8);
KD = zeros(12,12);

% set up Gauss quadrature
if p == 1
    eta = 0; %quadrature points
    w = 2; %quadrature weights
elseif p == 2
    eta = [-1/sqrt(3),1/sqrt(3)];
    w = [1,1];
elseif p == 3
    eta = [0,-sqrt(3/5),sqrt(3/5)];
    w = [8/9,5/9,5/9];
elseif p == 4
    eta = [-sqrt(3/7 - (2/7)*sqrt(6/5)),sqrt(3/7 - (2/7)*sqrt(6/5)),-sqrt(3/7 + (2/7)*sqrt(6/5)),sqrt(3/7 + (2/7)*sqrt(6/5))];
        w = [(18+sqrt(30))/36,(18+sqrt(30))/36,(18-sqrt(30))/36,(18-sqrt(30))/36];
end

J_det = area / 4; % determinant of Jacobian of transformation to (xi, eta)
dedx = 2/abs(x_e(2)-x_e(1));
dndy = 2/abs(y_e(3)-y_e(2));

%%%%%%%%%%%%%%%%%%%% shape function computation %%%%%%%%%%%%%%%%%%%%%%%%%

Hw =  @(a,b)reshape([(a.*2.0+1.0).*(b-1.0).*(-1.0./4.0)-(a./8.0-1.0./8.0).*(b-1.0).*2.0, ...
    (a./8.0-1.0./8.0).*(b-1.0).*-2.0-(a./8.0-1.0./8.0).*(b.*2.0+1.0).*2.0, ...
    a.*(-1.0./4.0)-b./4.0-((b.*2.0+1.0).*(b-1.0))./4.0-(a.*2.0+1.0).*(a./8.0-1.0./8.0).*2.0-a.^2./4.0-b.^2./4.0+1.0./2.0, ...
    (a.*2.0-2.0).*(b-1.0).*(-1.0./4.0)-((a+1.0).*(b-1.0))./4.0,0.0, ...
    (a-1.0).^2.*(-1.0./4.0)-((a.*2.0-2.0).*(a+1.0))./4.0,0.0, ...
    (b.*2.0-2.0).*(a-1.0).*(-1.0./4.0)-((a-1.0).*(b+1.0))./4.0, ...
    (b-1.0).^2.*(-1.0./4.0)-((b.*2.0-2.0).*(b+1.0))./4.0, ...
    ((a.*2.0-1.0).*(b-1.0))./4.0+(a./8.0+1.0./8.0).*(b-1.0).*2.0, ...
    (a./8.0+1.0./8.0).*(b-1.0).*2.0+(a./8.0+1.0./8.0).*(b.*2.0+1.0).*2.0, ...
    a.*(-1.0./4.0)+b./4.0+((b.*2.0+1.0).*(b-1.0))./4.0+(a.*2.0-1.0).*(a./8.0+1.0./8.0).*2.0+a.^2./4.0+b.^2./4.0-1.0./2.0, ...
    (a.*2.0+2.0).*(b-1.0).*(-1.0./4.0)-((a-1.0).*(b-1.0))./4.0, ...
    0.0,(a+1.0).^2.*(-1.0./4.0)-((a.*2.0+2.0).*(a-1.0))./4.0, ...
    0.0,((b.*2.0-2.0).*(a+1.0))./4.0+((a+1.0).*(b+1.0))./4.0, ...
    (b-1.0).^2./4.0+((b.*2.0-2.0).*(b+1.0))./4.0, ...
    (a.*2.0-1.0).*(b+1.0).*(-1.0./4.0)-(a./8.0+1.0./8.0).*(b+1.0).*2.0, ...
    (a./8.0+1.0./8.0).*(b+1.0).*-2.0-(a./8.0+1.0./8.0).*(b.*2.0-1.0).*2.0, ...
    a./4.0+b./4.0-((b.*2.0-1.0).*(b+1.0))./4.0-(a.*2.0-1.0).*(a./8.0+1.0./8.0).*2.0-a.^2./4.0-b.^2./4.0+1.0./2.0, ...
    ((a.*2.0+2.0).*(b+1.0))./4.0+((a-1.0).*(b+1.0))./4.0, ...
    0.0,(a+1.0).^2./4.0+((a.*2.0+2.0).*(a-1.0))./4.0,0.0, ...
    ((b.*2.0+2.0).*(a+1.0))./4.0+((a+1.0).*(b-1.0))./4.0, ...
    (b+1.0).^2./4.0+((b.*2.0+2.0).*(b-1.0))./4.0, ...
    ((a.*2.0+1.0).*(b+1.0))./4.0+(a./8.0-1.0./8.0).*(b+1.0).*2.0, ...
    (a./8.0-1.0./8.0).*(b+1.0).*2.0+(a./8.0-1.0./8.0).*(b.*2.0-1.0).*2.0, ...
    a./4.0-b./4.0+((b.*2.0-1.0).*(b+1.0))./4.0+(a.*2.0+1.0).*(a./8.0-1.0./8.0).*2.0+a.^2./4.0+b.^2./4.0-1.0./2.0, ...
    ((a.*2.0-2.0).*(b+1.0))./4.0+((a+1.0).*(b+1.0))./4.0,0.0, ...
```

```
    (a-1.0).^2./4.0+((a.*2.0-2.0).*(a+1.0))./4.0,0.0,(b.*2.0+2.0).*(a-1.0).*(-1.0./4.0)-((a-1.0).*(b-1.0))./4.0, ...
    (b+1.0).^2.*(-1.0./4.0)-((b.*2.0+2.0).*(b-1.0))./4.0],[3,12]);

% assemble stiffness matrix
for i = 1:p %loop over eta gauss points
     for j = 1:p %loop over nu gauss points
         % current values of x and y at Gauss points
         x = 0.25*[(1-eta(i))*(1-eta(j)),(1+eta(i))*(1-eta(j)),(1+eta(i))*(1+eta(j)),(1-eta(i))*(1+eta(j))]*x_e(:);
         y = 0.25*[(1-eta(i))*(1-eta(j)),(1+eta(i))*(1-eta(j)),(1+eta(i))*(1+eta(j)),(1-eta(i))*(1+eta(j))]*y_e(:);
         % value of H at current Gauss point H is a 3 by 8 matrix
         Hsub = (1 / area) * [(y - y_e(4)), 0, -(y - y_e(4)), 0, (y - y_e(1)), 0 , - (y - y_e(1)), 0;
             0, (x - x_e(2)), 0, -(x - x_e(1)), 0, (x - x_e(1)), 0, -(x - x_e(2));
             (x - x_e(2)), (y - y_e(4)), -(x - x_e(1)), -(y - y_e(4)), (x - x_e(1)), (y - y_e(1)), -(x - x_e(2)), -(y - y_e(1))];

         % value of Hw at current Gauss point is a 3 by 12 matrix
         Hwsub = Hw(eta(i),eta(j));
         Hwsub(1,:) = Hwsub(1,:)*dedx^2;
         Hwsub(2,:) = Hwsub(2,:)*dndy^2;
         Hwsub(3,:) = Hwsub(3,:)*dndy*dedx*2;



         % contribution to stiffness matrix from current Gauss point
         tol = max(eps(KA),eps(double(w(i) * w(j) * J_det * Hsub' * A * Hsub)));
         KA = KA + double(w(i) * w(j) * J_det * Hsub' * A * Hsub);   %12 by 12
         KA(abs(KA) <= 100*tol) = 0;

         tol = max(eps(KB),eps(double(w(i) * w(j) * J_det * Hsub' * B * Hwsub)));
         KB = KB - double(w(i) * w(j) * J_det * Hsub' * B * Hwsub);   %8 by 12
         KB(abs(KB) <= 100*tol) = 0;

         tol = max(eps(KC),eps(double(w(i) * w(j) * J_det * Hwsub' * B * Hsub)));
         KC = KC - double(w(i) * w(j) * J_det * Hwsub' * B * Hsub);   %12 by 8
         KC(abs(KC) <= 100*tol) = 0;

         tol = max(eps(KD),eps(double(w(i) * w(j) * J_det * Hwsub' * D * Hwsub)));
         KD = KD + double(w(i) * w(j) * J_det * Hwsub' * D * Hwsub);   %12 by 12
         KD(abs(KD) <= 100*tol) = 0;
     end
end

end
```

elelaminatestiff_con function

```
function [KA,KB,KC,KD] = elelaminatestiff_con(p,x_e,y_e,ply)
%this function computes the complete [A,B;B,D] laminate stiffness
%matrix with multiplication with the shape functions and the appropriate
%derivatives

%ply: [G12,E1,E2,v12,v21,theta,h]
plynum = size(ply,1);
%independent material constants:
G12 = ply(:,1); E1 = ply(:,2); E2 = ply(:,3); v12 = ply(:,4); v21 = ply(:,5);
theta = ply(:,6); h = ply(:,7);
%note: v12, v21, E1, E2 are coupled: v21 = E2.*v12./E1

%material compliance matrix for plies
for i = 1:plynum
    S_o(:,:,i) = [1/E1(i),-v21(i)/E2(i),0;-v12(i)/E1(i),1/E2(i),0;0,0,1/G12(i)].*10^10;
    %D_o(:,:,i) =[E1(i)/(1-v12(i)*v21(i)),v12(i)*E2(i)/(1-v12(i)*v21(i)),0;v12(i)*E2(i)/(1-v12(i)*v21(i)),E2(i)/(1-v12(i)*v21(i)),0;0,0,G12(i)];
end


%set up variables for finding the modulus-weighted midplane
E0 = max(E1); %reference modulus for midplane calculation
hstar = sum((E1./E0).*h);
z_hstar = 0;

%transformation of compliance matrix to principal directions for the
```

```
%laminate:
R = [1,0,0;0,1,0;0,0,2]; %Reuter's matrix
for i = 1:plynum
    c2 = cosd(theta(i)*(180/pi))^2;
    s2 = sind(theta(i)*(180/pi))^2;
    sc = sind(theta(i)*(180/pi))*cosd(theta(i)*(180/pi));
    At(:,:,i) = [c2,s2,2*sc;s2,c2,-2*sc;-sc,sc,c2-s2]; %stress rotation matrix
    %strain rotation matrix would be RAR^-1
    S_bar = R*inv(At(:,:,i))*inv(R)*S_o(:,:,i)*At(:,:,i); %transformed compliance matrix
    D_bar(:,:,i) = inv(S_bar./10^10); %transformed stiffness matrix
    z_hstar = z_hstar + E1(i)/E0*h(i)*(sum(h(1:i))*2-h(i))/2;
end

%zstar is the location of the modulus weighted midplane with respect to the
%reference coordinate system where the bottom of the laminate is defined as
%the z = 0 plane.
zstar = z_hstar/hstar;

%forming the laminate stiffness matrix components
A = zeros(size(D_bar(:,:,1)));
B = zeros(size(D_bar(:,:,1)));
D = zeros(size(D_bar(:,:,1)));
for i = 1:plynum
    A = A+D_bar(:,:,i)*((sum(h(1:i))-zstar)-(sum(h(1:i))-h(i)-zstar));
    B = B+0.5*D_bar(:,:,i)*((sum(h(1:i))-zstar)^2-(sum(h(1:i))-h(i)-zstar)^2);
    D = D+(1/3)*D_bar(:,:,i)*((sum(h(1:i))-zstar)^3-(sum(h(1:i))-h(i)-zstar)^3);
end

area = abs((x_e(3) - x_e(1)) * (y_e(3) - y_e(1))); % area of element (m^2)

% initialize a few variables
% total stiffness matrix for the element (segmented to 4 sub-matrices)
KA = zeros(8,8);
KB = zeros(8,16);
KC = zeros(16,8);
KD = zeros(16,16);

% set up Gauss quadrature
if p == 1
    eta = 0; %quadrature points
    w = 2; %quadrature weights
elseif p == 2
    eta = [-1/sqrt(3),1/sqrt(3)];
    w = [1,1];
elseif p == 3
    eta = [0,-sqrt(3/5),sqrt(3/5)];
    w = [8/9,5/9,5/9];
elseif p == 4
    eta = [-sqrt(3/7 - (2/7)*sqrt(6/5)),sqrt(3/7 - (2/7)*sqrt(6/5)),-sqrt(3/7 + (2/7)*sqrt(6/5)),sqrt(3/7 + (2/7)*sqrt(6/5))];
        w = [(18+sqrt(30))/36,(18+sqrt(30))/36,(18-sqrt(30))/36,(18-sqrt(30))/36];
end

J_det = area / 4; % determinant of Jacobian of transformation to (xi, eta)
dedx = 2/abs(x_e(2)-x_e(1));
dndy = 2/abs(y_e(3)-y_e(2));

%Hw is a 3 by 16 matrix
Hw = @(a,b)reshape([((a+2.0).*(b-1.0).^2.*(b+2.0))./8.0+((a.*2.0-2.0).*(b-1.0).^2.*(b+2.0))./8.0, ...
    ((a-1.0).^2.*(a+2.0).*(b+2.0))./8.0+((b.*2.0-2.0).*(a-1.0).^2.*(a+2.0))./8.0, ...
    ((a-1.0).^2.*(b-1.0).^2)./8.0+((a.*2.0-2.0).*(a+2.0).*(b-1.0).^2)./8.0+((b.*2.0-2.0).*(a-1.0).^2.*(b+2.0))./8.0+((a.*2.0-2.0).*(b.*2.0-2.0).*(a+2.0).*(b+2.0))./8.0, .
    ((a+1.0).*(b-1.0).^2.*(b+2.0))./8.0+((a.*2.0-2.0).*(b-1.0).^2.*(b+2.0))./8.0, ...
    ((a-1.0).^2.*(a+1.0).*(b+2.0))./8.0+((b.*2.0-2.0).*(a-1.0).^2.*(a+1.0))./8.0, ...
    ((a-1.0).^2.*(b-1.0).^2)./8.0+((a.*2.0-2.0).*(a+1.0).*(b-1.0).^2)./8.0+((b.*2.0-2.0).*(a-1.0).^2.*(b+2.0))./8.0+((a.*2.0-2.0).*(b.*2.0-2.0).*(a+1.0).*(b+2.0))./8.0, .
    ((a+2.0).*(b-1.0).^2.*(b+1.0))./8.0+((a.*2.0-2.0).*(b-1.0).^2.*(b+1.0))./8.0, ...
    ((a-1.0).^2.*(a+2.0).*(b+1.0))./8.0+((b.*2.0-2.0).*(a-1.0).^2.*(a+2.0))./8.0, ...
    ((a-1.0).^2.*(b-1.0).^2)./8.0+((a.*2.0-2.0).*(a+2.0).*(b-1.0).^2)./8.0+((b.*2.0-2.0).*(a-1.0).^2.*(b+1.0))./8.0+((a.*2.0-2.0).*(b.*2.0-2.0).*(a+2.0).*(b+1.0))./8.0, .
    ((a+1.0).*(b-1.0).^2.*(b+1.0))./8.0+((a.*2.0-2.0).*(b-1.0).^2.*(b+1.0))./8.0, ...
    ((a-1.0).^2.*(a+1.0).*(b+1.0))./8.0+((b.*2.0-2.0).*(a-1.0).^2.*(a+1.0))./8.0, ...
    ((a-1.0).^2.*(b-1.0).^2)./8.0+((a.*2.0-2.0).*(a+1.0).*(b-1.0).^2)./8.0+((b.*2.0-2.0).*(a-1.0).^2.*(b+1.0))./8.0+((a.*2.0-2.0).*(b.*2.0-2.0).*(a+1.0).*(b+1.0))./8.0, .
    (a-2.0).*(b-1.0).^2.*(b+2.0).*(-1.0./8.0)-((a.*2.0+2.0).*(b-1.0).^2.*(b+2.0))./8.0,(a+1.0).^2.*(a-2.0).*(b+2.0).*(-1.0./8.0)-((b.*2.0-2.0).*(a+1.0).^2.*(a-2.0))./8.0,
    (a+1.0).^2.*(b-1.0).^2.*(-1.0./8.0)-((a.*2.0+2.0).*(a-2.0).*(b-1.0).^2)./8.0-((b.*2.0-2.0).*(a+1.0).^2.*(b+2.0))./8.0-((a.*2.0+2.0).*(b.*2.0-2.0).*(a-2.0).*(b+2.0))./
    ((a-1.0).*(b-1.0).^2.*(b+2.0))./8.0+((a.*2.0+2.0).*(b-1.0).^2.*(b+2.0))./8.0,((a-1.0).*(a+1.0).^2.*(b+2.0))./8.0+((b.*2.0-2.0).*(a-1.0).*(a+1.0).^2)./8.0, ...
    ((a+1.0).^2.*(b-1.0).^2)./8.0+((a.*2.0+2.0).*(a-1.0).*(b-1.0).^2)./8.0+((b.*2.0-2.0).*(a+1.0).^2.*(b+2.0))./8.0+((a.*2.0+2.0).*(b.*2.0-2.0).*(a-1.0).*(b+2.0))./8.0, .
```

```
    (a-2.0).*(b-1.0).^2.*(b+1.0).*(-1.0./8.0)-((a.*2.0+2.0).*(b-1.0).^2.*(b+1.0))./8.0,(a+1.0).^2.*(a-2.0).*(b+1.0).*(-1.0./8.0)-((b.*2.0-2.0).*(a+1.0).^2.*(a-2.0))./8.0,
    (a+1.0).^2.*(b-1.0).^2.*(-1.0./8.0)-((a.*2.0+2.0).*(a-2.0).*(b-1.0).^2)./8.0-((b.*2.0-2.0).*(a+1.0).^2.*(b+1.0))./8.0-((a.*2.0+2.0).*(b.*2.0-2.0).*(a-2.0).*(b+1.0))./
    ((a-1.0).*(b-1.0).^2.*(b+1.0))./8.0+((a.*2.0+2.0).*(b-1.0).^2.*(b+1.0))./8.0,((a-1.0).*(a+1.0).^2.*(b+1.0))./8.0+((b.*2.0-2.0).*(a-1.0).*(a+1.0).^2)./8.0, ...
    ((a+1.0).^2.*(b-1.0).^2)./8.0+((a.*2.0+2.0).*(a-1.0).*(b-1.0).^2)./8.0+((b.*2.0-2.0).*(a+1.0).^2.*(b+1.0))./8.0+((a.*2.0+2.0).*(b.*2.0-2.0).*(a-1.0).*(b+1.0))./8.0, .
    ((a-2.0).*(b+1.0).^2.*(b-2.0))./8.0+((a.*2.0+2.0).*(b+1.0).^2.*(b-2.0))./8.0,((a+1.0).^2.*(a-2.0).*(b-2.0))./8.0+((b.*2.0+2.0).*(a+1.0).^2.*(a-2.0))./8.0, ...
    ((a+1.0).^2.*(b+1.0).^2)./8.0+((a.*2.0+2.0).*(a-2.0).*(b+1.0).^2)./8.0+((b.*2.0+2.0).*(a+1.0).^2.*(b-2.0))./8.0+((a.*2.0+2.0).*(b.*2.0+2.0).*(a-2.0).*(b-2.0))./8.0, .
    (a-1.0).*(b+1.0).^2.*(b-2.0).*(-1.0./8.0)-((a.*2.0+2.0).*(b+1.0).^2.*(b-2.0))./8.0,(a-1.0).*(a+1.0).^2.*(b-2.0).*(-1.0./8.0)-((b.*2.0+2.0).*(a-1.0).*(a+1.0).^2)./8.0,
    (a+1.0).^2.*(b+1.0).^2.*(-1.0./8.0)-((a.*2.0+2.0).*(a-1.0).*(b+1.0).^2)./8.0-((b.*2.0+2.0).*(a+1.0).^2.*(b-2.0))./8.0-((a.*2.0+2.0).*(b.*2.0+2.0).*(a-1.0).*(b-2.0))./
    (a-2.0).*(b-1.0).*(b+1.0).^2.*(-1.0./8.0)-((a.*2.0+2.0).*(b-1.0).*(b+1.0).^2)./8.0,(a+1.0).^2.*(a-2.0).*(b-1.0).*(-1.0./8.0)-((b.*2.0+2.0).*(a+1.0).^2.*(a-2.0))./8.0,
    (a+1.0).^2.*(b+1.0).^2.*(-1.0./8.0)-((a.*2.0+2.0).*(a-2.0).*(b+1.0).^2)./8.0-((b.*2.0+2.0).*(a+1.0).^2.*(b-1.0))./8.0-((a.*2.0+2.0).*(b.*2.0+2.0).*(a-2.0).*(b-1.0))./
    ((a-1.0).*(b-1.0).*(b+1.0).^2)./8.0+((a.*2.0+2.0).*(b-1.0).*(b+1.0).^2)./8.0,((a-1.0).*(a+1.0).^2.*(b-1.0))./8.0+((b.*2.0+2.0).*(a-1.0).*(a+1.0).^2)./8.0, ...
    ((a+1.0).^2.*(b+1.0).^2)./8.0+((a.*2.0+2.0).*(a-1.0).*(b+1.0).^2)./8.0+((b.*2.0+2.0).*(a+1.0).^2.*(b-1.0))./8.0+((a.*2.0+2.0).*(b.*2.0+2.0).*(a-1.0).*(b-1.0))./8.0, .
    (a+2.0).*(b+1.0).^2.*(b-2.0).*(-1.0./8.0)-((a.*2.0-2.0).*(b+1.0).^2.*(b-2.0))./8.0,(a-1.0).^2.*(a+2.0).*(b-2.0).*(-1.0./8.0)-((b.*2.0+2.0).*(a-1.0).^2.*(a+2.0))./8.0,
    (a-1.0).^2.*(b+1.0).^2.*(-1.0./8.0)-((a.*2.0-2.0).*(a+2.0).*(b+1.0).^2)./8.0-((b.*2.0+2.0).*(a-1.0).^2.*(b-2.0))./8.0-((a.*2.0-2.0).*(b.*2.0+2.0).*(a+2.0).*(b-2.0))./
    (a+1.0).*(b+1.0).^2.*(b-2.0).*(-1.0./8.0)-((a.*2.0-2.0).*(b+1.0).^2.*(b-2.0))./8.0,(a-1.0).^2.*(a+1.0).*(b-2.0).*(-1.0./8.0)-((b.*2.0+2.0).*(a-1.0).^2.*(a+1.0))./8.0,
    (a-1.0).^2.*(b+1.0).^2.*(-1.0./8.0)-((a.*2.0-2.0).*(a+1.0).*(b+1.0).^2)./8.0-((b.*2.0+2.0).*(a-1.0).^2.*(b-2.0))./8.0-((a.*2.0-2.0).*(b.*2.0+2.0).*(a+1.0).*(b-2.0))./
    ((a+2.0).*(b-1.0).*(b+1.0).^2)./8.0+((a.*2.0-2.0).*(b-1.0).*(b+1.0).^2)./8.0,((a-1.0).^2.*(a+2.0).*(b-1.0))./8.0+((b.*2.0+2.0).*(a-1.0).^2.*(a+2.0))./8.0, ...
    ((a-1.0).^2.*(b+1.0).^2)./8.0+((a.*2.0-2.0).*(a+2.0).*(b+1.0).^2)./8.0+((b.*2.0+2.0).*(a-1.0).^2.*(b-1.0))./8.0+((a.*2.0-2.0).*(b.*2.0+2.0).*(a+2.0).*(b-1.0))./8.0, .
    ((a+1.0).*(b-1.0).*(b+1.0).^2)./8.0+((a.*2.0-2.0).*(b-1.0).*(b+1.0).^2)./8.0,((a-1.0).^2.*(a+1.0).*(b-1.0))./8.0+((b.*2.0+2.0).*(a-1.0).^2.*(a+1.0))./8.0, ...
    ((a-1.0).^2.*(b+1.0).^2)./8.0+((a.*2.0-2.0).*(a+1.0).*(b+1.0).^2)./8.0+((b.*2.0+2.0).*(a-1.0).^2.*(b-1.0))./8.0+((a.*2.0-2.0).*(b.*2.0+2.0).*(a+1.0).*(b-1.0))./8.0],[

% assemble stiffness matrix
for i = 1:p %loop over eta gauss points
     for j = 1:p %loop over nu gauss points
         % current values of x and y at Gauss points
         x = 0.25*[(1-eta(i))*(1-eta(j)),(1+eta(i))*(1-eta(j)),(1+eta(i))*(1+eta(j)),(1-eta(i))*(1+eta(j))]*x_e(:);
         y = 0.25*[(1-eta(i))*(1-eta(j)),(1+eta(i))*(1-eta(j)),(1+eta(i))*(1+eta(j)),(1-eta(i))*(1+eta(j))]*y_e(:);

         % value of H at current Gauss point H is a 3 by 8 matrix
         H = (1 / area) * [(y - y_e(4)), 0, -(y - y_e(4)), 0, (y - y_e(1)), 0 , - (y - y_e(1)), 0;
             0, (x - x_e(2)), 0, -(x - x_e(1)), 0, (x - x_e(1)), 0, -(x - x_e(2));
             (x - x_e(2)), (y - y_e(4)), -(x - x_e(1)), -(y - y_e(4)), (x - x_e(1)), (y - y_e(1)), -(x - x_e(2)), -(y - y_e(1))];

         % value of Hw at current Gauss point is a 3 by 12 matrix
         Hwsub = Hw(eta(i),eta(j));
         Hwsub(1,:) = Hwsub(1,:)*dedx^2;
         Hwsub(2,:) = Hwsub(2,:)*dndy^2;
         Hwsub(3,:) = Hwsub(3,:)*dndy*dedx*2;

         % contribution to stiffness matrix from current Gauss point
         tol = max(eps(KA),eps(double(w(i) * w(j) * J_det * H' * A * H)));
         KA = KA + w(i) * w(j) * J_det * H' * A * H;    %8 by 8
         KA(abs(KA) <= 100*tol) = 0;

         tol = max(eps(KB),eps(double(w(i) * w(j) * J_det * H' * B * Hwsub)));
         KB = KB - double(w(i) * w(j) * J_det * H' * B * Hwsub);   %8 by 16
         KB(abs(KB) <= 100*tol) = 0;

         tol = max(eps(KC),eps(double(w(i) * w(j) * J_det * Hwsub' * B * H)));
         KC = KC - double(w(i) * w(j) * J_det * Hwsub' * B * H);   %16 by 8
         KC(abs(KC) <= 100*tol) = 0;

         tol = max(eps(KD),eps(double(w(i) * w(j) * J_det * Hwsub' * D * Hwsub)));
         KD = KD + double(w(i) * w(j) * J_det * Hwsub' * D * Hwsub);   %16 by 16
         KD(abs(KD) <= 100*tol) = 0;
     end
end

end
```

solve_con function

```
function [d,d_dof,f] = solve_con(connec,nodecoord,BC,ktot,p)
%BC is a MATLAB data struct that contains the boundary conditions applied,
%including the physical location, the type of BC, and the corresponding
%value or function handle.

%BC.type condition cases: x,y,w,wx,wy,wxy - Displacement BCs
    %                    Nx,Ny,Nxy - Distributed Axial Load BCs (N/m)
    %                    Fx,Fy,Fxy - Point Axial Load BCs (N)
```

```
    %                     Mx,My,Mxy - Distributed Axial Moment BCs (Nm/m)
    %                     q - Domain Transverse Load (N/m^2)
    %                     Q - Distributed Transverse Load BC(transverse shear force) (N/m)

d = zeros(6*max(nodecoord(:,1)),1);
f = zeros(6*max(nodecoord(:,1)),1);
d_dof = zeros(6*max(nodecoord(:,1)),1);

%loop through all BC input and construct the F (force) vector
for i = 1:size(BC.pos,1)
    %identifying relevant nodes for displacement BCs
    if length(BC.pos{i}{1}) == 2 %range of coordinate specified for x-axis
        if length(BC.pos{i}{2}) == 2 %an area specified
            nodes = nodecoord(nodecoord(:,2)>=BC.pos{i}{1}(1)&nodecoord(:,2)<=BC.pos{i}{1}(2)& ...
                nodecoord(:,3)>=BC.pos{i}{2}(1)&nodecoord(:,3)<=BC.pos{i}{2}(2),1);
        else %only a line of x-coords
            nodes = nodecoord(nodecoord(:,2)>=BC.pos{i}{1}(1)&nodecoord(:,2)<=BC.pos{i}{1}(2)&BC.pos{i}{2}==nodecoord(:,3),1);
        end
    elseif length(BC.pos{i}{2}) == 2 %range of coordinate specified for y-axis (only a line of y-coord)
        nodes = nodecoord(nodecoord(:,3)>=BC.pos{i}{2}(1)&nodecoord(:,3)<=BC.pos{i}{2}(2)&BC.pos{i}{1}==nodecoord(:,2),1);
    else %point coordinate specification
        nodes = nodecoord(nodecoord(:,2)==BC.pos{i}{1}&nodecoord(:,3)==BC.pos{i}{2},1);
    end

    %BC condition cases: x,y,w,wx,wy,wxy - Displacement BCs
    %                    Nx,Ny,Nxy - Distributed Axial Load BCs (N/m)
    %                    Mx,My,Mxy - Distributed Axial Moment BCs (Nm/m)
    %                    q - Domain Transverse Load (N/m^2)
    %                    Q - Distributed Transverse Load BC (N/m)
    %all of the above could be point loads/moments
    for j = 1:size(BC.type{i},2)
        if BC.type{i}{j} == 'x'  %selected node(s)'s x displacement
            d(nodes*2-1) = BC.val{i}{j};
            d_dof(nodes*2-1) = 1;
        elseif BC.type{i}{j} == 'y'  %selected node(s)'s y displacement
            d(nodes*2) = BC.val{i}{j};
            d_dof(nodes*2) = 1;
        elseif BC.type{i}{j} == 'w'  %selected node(s)'s z displacement
            d(nodes*4-3+2*max(nodecoord(:,1))) = BC.val{i}{j};
            d_dof(nodes*4-3+2*max(nodecoord(:,1))) = 1;
        elseif strcmp(BC.type{i}{j},'wx')  %selected node(s)'s rotation about x
            d(nodes*4-2+2*max(nodecoord(:,1))) = BC.val{i}{j};
            d_dof(nodes*4-2+2*max(nodecoord(:,1))) = 1;
        elseif strcmp(BC.type{i}{j},'wy')  %selected node(s)'s rotation about y
            d(nodes*4-1+2*max(nodecoord(:,1))) = BC.val{i}{j};
            d_dof(nodes*4-1+2*max(nodecoord(:,1))) = 1;
        elseif strcmp(BC.type{i}{j},'wxy')  %selected node(s)'s twist xy rotation
            d(nodes*4+2*max(nodecoord(:,1))) = BC.val{i}{j};
            d_dof(nodes*4+2*max(nodecoord(:,1))) = 1;
        elseif strcmp(BC.type{i}{j},'Nx') %selected node(s)'s load along x-axis
            %(this should only be specified for boundary parallel to y-axis
            if length(BC.pos{i}{1}) == 1
                BCval = [BC.val{i}{j};0];
                curpos = BC.pos{i};
                ax = 2;
                shapefcn = 1;
                f = constructF_con(connec, nodecoord, f, curpos, BCval, ax, shapefcn, p);
            end
        elseif strcmp(BC.type{i}{j},'Ny') %selected node(s)'s load along y-axis
            %(this should only be specified for boundary parallel to x-axis
            if length(BC.pos{i}{2}) == 1
                BCval = [0;BC.val{i}{j}];
                curpos = BC.pos{i};
                ax = 1;
                shapefcn = 1;
                f = constructF_con(connec, nodecoord, f, curpos, BCval, ax, shapefcn, p);
            end
        elseif strcmp(BC.type{i}{j},'Nxy') %this is for shear load parallel to y-axis (on constant x surface)
            %note the boundary surface is also parallel to y-axis
            if length(BC.pos{i}{1}) == 1
                BCval = [0;BC.val{i}{j}];
                curpos = BC.pos{i};
                ax = 2;
```

```
            shapefcn = 1;
            f = constructF_con(connec, nodecoord, f, curpos, BCval, ax, shapefcn, p);
        end
    elseif strcmp(BC.type{i}{j},'Nyx') %this is for shear load parallel to x-axis (on constant y surface)
        %note the boundary surface is also parallel to x-axis
        if length(BC.pos{i}{2}) == 1
            BCval = [BC.val{i}{j};0];
            curpos = BC.pos{i};
            ax = 1;
            shapefcn = 1;
            f = constructF_con(connec, nodecoord, f, curpos, BCval, ax, shapefcn, p);
        end
    elseif strcmp(BC.type{i}{j},'Mx') %this is for bending moment applied along the x-axis (rotate about y-axis)
        %note the boundary surface is also parallel to y-axis (constant x)
        if length(BC.pos{i}{1}) == 1
            BCval = [BC.val{i}{j};0];
            curpos = BC.pos{i};
            ax = 2;
            shapefcn = 2;
            f = constructF_con(connec, nodecoord, f, curpos, BCval, ax, shapefcn, p);
        end
    elseif strcmp(BC.type{i}{j},'My') %this is for bending moment applied along the y-axis (rotate about x-axis)
        %note the boundary surface is also parallel to x-axis (constant y)
        if length(BC.pos{i}{2}) == 1
            BCval = [0;BC.val{i}{j}];
            curpos = BC.pos{i};
            ax = 1;
            shapefcn = 2;
            f = constructF_con(connec, nodecoord, f, curpos, BCval, ax, shapefcn, p);
        end
    elseif strcmp(BC.type{i}{j},'Mxy') %this is for twisting moment applied along the y-axis (rotate about x-axis)
        %note the boundary surface is parallel to y-axis (constant x)
        if length(BC.pos{i}{1}) == 1
            BCval = [0;BC.val{i}{j}];
            curpos = BC.pos{i};
            ax = 2;
            shapefcn = 2;
            f = constructF_con(connec, nodecoord, f, curpos, BCval, ax, shapefcn, p);
        end
    elseif strcmp(BC.type{i}{j}, 'Myx') %this is for twisting moment applied along the x-axis (rotate about y-axis)
        %note the boundary surface is also parallel to x-axis (constant y)
        if length(BC.pos{i}{2}) == 1
            BCval = [BC.val{i}{j};0];
            curpos = BC.pos{i};
            ax = 1;
            shapefcn = 2;
            f = constructF_con(connec, nodecoord, f, curpos, BCval, ax, shapefcn, p);
        end
    elseif strcmp(BC.type{i}{j},'Q') %this is for transverse(through-thickness shear)
        if length(BC.pos{i}{1}) == 1
            ax = 2;
        elseif length(BC.pos{i}{2}) == 1
            ax = 1;
        end
        curpos = BC.pos{i};
        shapefcn = 3;
        f = constructF_con(connec, nodecoord, f, curpos, BC.val{i}{j}, ax, shapefcn, p);
    elseif strcmp(BC.type{i}{j},'q')  %Domain Transverse Load (N/m^2)
        curpos = BC.pos{i};
        shapefcn = 4;
        ax = 0;
        f = constructF_con(connec, nodecoord, f, curpos, BC.val{i}{j}, ax, shapefcn, p);
    end
  end
end

fdiff = f(d_dof==0)-ktot(d_dof==0,d_dof==1)*d(d_dof==1);
d(d_dof == 0) = ktot(d_dof == 0,d_dof==0)\fdiff;
f(d_dof == 1) = ktot(d_dof==1,:)*d;

end
```

solve_noncon function

```
function [d,d_dof,f] = solve_noncon(connec,nodecoord,BC,ktot,p)
%for non-conforming case
%BC is a MATLAB data struct that contains the boundary conditions applied,
%including the physical location, the type of BC, and the corresponding
%value or function handle.

%BC.type condition cases: x,y,w,wx,wy - Displacement BCs
    %                    Nx,Ny,Nxy - Distributed Axial Load BCs (N/m)
    %                    Fx,Fy,Fxy - Point Axial Load BCs (N)
    %                    Mx,My,Mxy - Distributed Axial Moment BCs (Nm/m)
    %                    q - Domain Transverse Load (N/m^2)
    %                    Q - Distributed Transverse Load BC(transverse shear force) (N/m)

d = zeros(5*max(nodecoord(:,1)),1);
f = zeros(5*max(nodecoord(:,1)),1);
d_dof = zeros(5*max(nodecoord(:,1)),1);

%loop through all BC input and construct the F (force) vector
for i = 1:size(BC.pos,1)
    %identifying relevant nodes for displacement BCs
    if length(BC.pos{i}{1}) == 2 %range of coordinate specified for x-axis
        if length(BC.pos{i}{2}) == 2 %an area specified
            nodes = nodecoord(nodecoord(:,2)>=BC.pos{i}{1}(1)&nodecoord(:,2)<=BC.pos{i}{1}(2)& ...
                nodecoord(:,3)>=BC.pos{i}{2}(1)&nodecoord(:,3)<=BC.pos{i}{2}(2),1);
        else %only a line of x-coords
            nodes = nodecoord(nodecoord(:,2)>=BC.pos{i}{1}(1)&nodecoord(:,2)<=BC.pos{i}{1}(2)&BC.pos{i}{2}==nodecoord(:,3),1);
        end
    elseif length(BC.pos{i}{2}) == 2 %range of coordinate specified for y-axis (only a line of y-coord)
        nodes = nodecoord(nodecoord(:,3)>=BC.pos{i}{2}(1)&nodecoord(:,3)<=BC.pos{i}{2}(2)&BC.pos{i}{1}==nodecoord(:,2),1);
    else %point coordinate specification
        nodes = nodecoord(nodecoord(:,2)==BC.pos{i}{1}&nodecoord(:,3)==BC.pos{i}{2},1);
    end

    %BC condition cases: x,y,w,wx,wy - Displacement BCs
    %                    Nx,Ny,Nxy - Distributed Axial Load BCs (N/m)
    %                    Mx,My,Mxy - Distributed Axial Moment BCs (Nm/m)
    %                    q - Domain Transverse Load (N/m^2)
    %                    Q - Distributed Transverse Load BC (N/m)
    %all of the above could be point loads/moments
    for j = 1:size(BC.type{i},2)
        if BC.type{i}{j} == 'x'  %selected node(s)'s x displacement
            d(nodes*2-1) = BC.val{i}{j};
            d_dof(nodes*2-1) = 1;
        elseif BC.type{i}{j} == 'y'  %selected node(s)'s y displacement
            d(nodes*2) = BC.val{i}{j};
            d_dof(nodes*2) = 1;
        elseif BC.type{i}{j} == 'w'  %selected node(s)'s z displacement
            d(nodes*3-2+2*max(nodecoord(:,1))) = BC.val{i}{j};
            d_dof(nodes*3-2+2*max(nodecoord(:,1))) = 1;
        elseif strcmp(BC.type{i}{j},'wx')  %selected node(s)'s rotation about x
            d(nodes*3-1+2*max(nodecoord(:,1))) = BC.val{i}{j};
            d_dof(nodes*3-1+2*max(nodecoord(:,1))) = 1;
        elseif strcmp(BC.type{i}{j},'wy')  %selected node(s)'s rotation about y
            d(nodes*3+2*max(nodecoord(:,1))) = BC.val{i}{j};
            d_dof(nodes*3+2*max(nodecoord(:,1))) = 1;
        elseif strcmp(BC.type{i}{j},'Nx') %selected node(s)'s load along x-axis
            %(this should only be specified for boundary parallel to y-axis
            if length(BC.pos{i}{1}) == 1
                BCval = [BC.val{i}{j};0];
                curpos = BC.pos{i};
                ax = 2;
                shapefcn = 1;
                f = constructF_noncon(connec, nodecoord, f, curpos, BCval, ax, shapefcn, p);
            end
        elseif strcmp(BC.type{i}{j},'Ny') %selected node(s)'s load along y-axis
            %(this should only be specified for boundary parallel to x-axis
            if length(BC.pos{i}{2}) == 1
                BCval = [0;BC.val{i}{j}];
                curpos = BC.pos{i};
                ax = 1;
                shapefcn = 1;
```

```
            f = constructF_noncon(connec, nodecoord, f, curpos, BCval, ax, shapefcn, p);
        end
    elseif strcmp(BC.type{i}{j},'Nxy') %this is for shear load parallel to y-axis (on constant x surface)
        %note the boundary surface is also parallel to y-axis
        if length(BC.pos{i}{1}) == 1
            BCval = [0;BC.val{i}{j}];
            curpos = BC.pos{i};
            ax = 2;
            shapefcn = 1;
            f = constructF_noncon(connec, nodecoord, f, curpos, BCval, ax, shapefcn, p);
        end
    elseif strcmp(BC.type{i}{j},'Nyx') %this is for shear load parallel to x-axis (on constant y surface)
        %note the boundary surface is also parallel to x-axis
        if length(BC.pos{i}{2}) == 1
            BCval = [BC.val{i}{j};0];
            curpos = BC.pos{i};
            ax = 1;
            shapefcn = 1;
            f = constructF_noncon(connec, nodecoord, f, curpos, BCval, ax, shapefcn, p);
        end
    elseif strcmp(BC.type{i}{j},'Mx') %this is for bending moment applied along the x-axis (rotate about y-axis)
        %note the boundary surface is also parallel to y-axis (constant x)
        if length(BC.pos{i}{1}) == 1
            BCval = [BC.val{i}{j};0];
            curpos = BC.pos{i};
            ax = 2;
            shapefcn = 2;
            f = constructF_noncon(connec, nodecoord, f, curpos, BCval, ax, shapefcn, p);
        end
    elseif strcmp(BC.type{i}{j},'My') %this is for bending moment applied along the y-axis (rotate about x-axis)
        %note the boundary surface is also parallel to x-axis (constant y)
        if length(BC.pos{i}{2}) == 1
            BCval = [0;BC.val{i}{j}];
            curpos = BC.pos{i};
            ax = 1;
            shapefcn = 2;
            f = constructF_noncon(connec, nodecoord, f, curpos, BCval, ax, shapefcn, p);
        end
    elseif strcmp(BC.type{i}{j},'Mxy') %this is for twisting moment applied along the y-axis (rotate about x-axis)
        %note the boundary surface is parallel to y-axis (constant x)
        if length(BC.pos{i}{1}) == 1
            BCval = [0;BC.val{i}{j}];
            curpos = BC.pos{i};
            ax = 2;
            shapefcn = 2;
            f = constructF_noncon(connec, nodecoord, f, curpos, BCval, ax, shapefcn, p);
        end
    elseif strcmp(BC.type{i}{j}, 'Myx') %this is for twisting moment applied along the x-axis (rotate about y-axis)
        %note the boundary surface is also parallel to x-axis (constant y)
        if length(BC.pos{i}{2}) == 1
            BCval = [BC.val{i}{j};0];
            curpos = BC.pos{i};
            ax = 1;
            shapefcn = 2;
            f = constructF_noncon(connec, nodecoord, f, curpos, BCval, ax, shapefcn, p);
        end
    elseif strcmp(BC.type{i}{j},'Q') %this is for transverse(through-thickness shear)
        if length(BC.pos{i}{1}) == 1
            ax = 2;
        elseif length(BC.pos{i}{2}) == 1
            ax = 1;
        end
        curpos = BC.pos{i};
        shapefcn = 3;
        f = constructF_noncon(connec, nodecoord, f, curpos, BC.val{i}{j}, ax, shapefcn, p);
    elseif strcmp(BC.type{i}{j},'q')  %Domain Transverse Load (N/m^2)
        curpos = BC.pos{i};
        shapefcn = 4;
        ax = 0;
        f = constructF_noncon(connec, nodecoord, f, curpos, BC.val{i}{j}, ax, shapefcn, p);

    end
end
```

```
end

fdiff = f(d_dof==0)-ktot(d_dof==0,d_dof==1)*d(d_dof==1);
d(d_dof == 0) = ktot(d_dof == 0,d_dof==0)\fdiff;
f(d_dof == 1) = ktot(d_dof==1,:)*d;
end
```

constructF_noncon function

```
function [fout] = constructF_noncon(connec, nodecoord, f, curpos, BCval, ax, shapefcn, p)
%for non-conforming case
%input var: connec, nodecoord, nodes, f, curpos = BC.pos{i}, ax,
%shapefcn,p, BCval

%connec, nodecoord, nodes are taken from solve as it is

%f is the last updated force vector

%BCval is a syms or value vector/scalar

%curpos = BC.pos{i} from solve, it's a cell vector

%p: Gauss quadrature order

%shapefcn: associated shape function
%shapefcn = 1: in-plane stress loading(Nx,Ny,Nxy)
%shapefcn = 2: moments(Mx,My,Mxy)
%shapefcn = 3 and 4: transverse loading (Q,q)

%ax: integration boundary
%ax = 0: area of integration
%ax = 1: boundary parallel to x-axis (constant y)
%ax = 2: boundary parallel to y-axis (constant x)

%find all the relevant elements
elem = [];
ispoint = 0;
%for i = 1:length(nodes) %loop over all the relevant BC nodes

%assume the applied BC is either a point load/moment, or the length scale
%of the BC is at least the size of one element's dimension
for i = 1:size(connec,1) %loop over all elements to find relevant ones
    %val = nodes(i); %current node
    %find all elements attached to the node
    curelem = connec(i,:);
    flag = 0;
    if sum(curelem(1)==elem)==0 %make sure element is not a duplicate
        if length(curpos{1}) == 1 && length(curpos{2})==1 %point load case
            ispoint = 1;
            if curpos{1} >= nodecoord(curelem(2),2) && ...
                    curpos{1} <= nodecoord(curelem(3),2) && ...
                    curpos{2} >= nodecoord(curelem(2),3) && ...
                    curpos{2} <= nodecoord(curelem(4),3)
                %the point load application is with in the current element
            else
                %the point load is not in the current element
                flag = 1;
            end
        else

            if ax == 0 %distributed area loading
                %if at least 1 node connected to the element are within BC
                %application area, then leave flag to be 0
                if nodecoord(curelem(2),2) <= curpos{1}(end) && nodecoord(curelem(2),3) <= curpos{2}(end) ...
                    && nodecoord(curelem(2),2) >= curpos{1}(1) && nodecoord(curelem(2),3) >= curpos{2}(1)
                elseif nodecoord(curelem(3),2) <= curpos{1}(end) && nodecoord(curelem(3),3) <= curpos{2}(end) ...
                    && nodecoord(curelem(3),2) >= curpos{1}(1) && nodecoord(curelem(3),3) >= curpos{2}(1)
                elseif nodecoord(curelem(3),2) <= curpos{1}(end) && nodecoord(curelem(3),3) <= curpos{2}(end) ...
                    && nodecoord(curelem(3),2) >= curpos{1}(1) && nodecoord(curelem(3),3) >= curpos{2}(1)
                elseif nodecoord(curelem(4),2) <= curpos{1}(end) && nodecoord(curelem(4),3) <= curpos{2}(end) ...
```

```
                    && nodecoord(curelem(4),2) >= curpos{1}(1) && nodecoord(curelem(4),3) >= curpos{2}(1)
                else
                    %if none of nodes connected to the element is inside of BC
                    %boundary, then change flag to be 1
                    flag = 1;
                end
            elseif ax == 1 %distributed boundary line loading along x (constant y surface)
                if nodecoord(curelem(2),3) == curpos{2} || nodecoord(curelem(4),3) == curpos{2}
                    if nodecoord(curelem(2),2) <= curpos{1}(end) && nodecoord(curelem(2),2) >= curpos{1}(1)
                        %if the any of the nodes connected to the element is within BC
                        %x-axis boundary, then leave flag to be 0
                    elseif nodecoord(curelem(3),2) <= curpos{1}(end) && nodecoord(curelem(3),2) >= curpos{1}(1)
                    else
                        flag = 1;
                    end
                else
                    %otherwise, change flag to be 1
                    flag = 1;
                end
            elseif ax == 2 %distributed boundary line loading along y (constant x surface)
                if nodecoord(curelem(2),2) == curpos{1} || nodecoord(curelem(4),2) == curpos{1}
                    if nodecoord(curelem(2),3) <= curpos{2}(end) && nodecoord(curelem(2),3) >= curpos{2}(1)
                    %if the all nodes connected to the element is within BC
                    %boundary, then leave flag to be 0
                    elseif nodecoord(curelem(4),3) <= curpos{2}(end) && nodecoord(curelem(4),3) >= curpos{2}(1)
                    else
                        flag = 1;
                    end
                else
                    %if the any nodes connected to the element is outside of BC
                    %boundary, then change flag to be 1
                    flag = 1;
                end
            end
            %end
        end
        %if all nodes of the element is within BC boundary, then this
        %element will be apart of the Gauss quadrature computation
        if flag == 0
            elem = [elem;curelem(1)];
        end
    end
    %end
end

% set up Gauss quadrature
if p == 1
    eta = 0; %quadrature points
    w = 2; %quadrature weights
elseif p == 2
    eta = [-1/sqrt(3),1/sqrt(3)];
    w = [1,1];
elseif p == 3
    eta = [0,-sqrt(3/5),sqrt(3/5)];
    w = [8/9,5/9,5/9];
elseif p == 4
    eta = [-sqrt(3/7 - (2/7)*sqrt(6/5)),sqrt(3/7 - (2/7)*sqrt(6/5)),-sqrt(3/7 + (2/7)*sqrt(6/5)),sqrt(3/7 + (2/7)*sqrt(6/5))];
        w = [(18+sqrt(30))/36,(18+sqrt(30))/36,(18-sqrt(30))/36,(18-sqrt(30))/36];
end

%%%%%%%%%%%%%%%%%%%%% shape function computation %%%%%%%%%%%%%%%%%%%%%%%%%%
%shapefcn = 1: in-plane stress loading(Nx,Ny,Nxy)
%shapefcn = 2: moments(Mx,My,Mxy)
%shapefcn = 3 and 4: transverse loading (Q,q)

if shapefcn == 2
    B = @(a,b)reshape([(b-1.0).*(a+b+a.^2+b.^2-2.0).*(-1.0./8.0)-(a.*2.0+1.0).*(a./8.0-1.0./8.0).*(b-1.0), ...
        -(a./8.0-1.0./8.0).*(a+b+a.^2+b.^2-2.0)-(a./8.0-1.0./8.0).*(b.*2.0+1.0).*(b-1.0),(a-1.0).^2.*(b-1.0).*(-1.0./8.0)-((a.*2.0-2.0).*(a+1.0).*(b-1.0))./8.0, ...
        (a-1.0).^2.*(a+1.0).*(-1.0./8.0),(b-1.0).^2.*(b+1.0).*(-1.0./8.0),(a-1.0).*(b-1.0).^2.*(-1.0./8.0)-((b.*2.0-2.0).*(a-1.0).*(b+1.0))./8.0, ...
        ((b-1.0).*(-a+b+a.^2+b.^2-2.0))./8.0+(a.*2.0-1.0).*(a./8.0+1.0./8.0).*(b-1.0), ...
        (a./8.0+1.0./8.0).*(-a+b+a.^2+b.^2-2.0)+(a./8.0+1.0./8.0).*(b.*2.0+1.0).*(b-1.0), ...
        (a+1.0).^2.*(b-1.0).*(-1.0./8.0)-((a.*2.0+2.0).*(a-1.0).*(b-1.0))./8.0, ...
        (a-1.0).*(a+1.0).^2.*(-1.0./8.0),((b-1.0).^2.*(b+1.0))./8.0, ...
```

```
        ((a+1.0).*(b-1.0).^2)./8.0+((b.*2.0-2.0).*(a+1.0).*(b+1.0))./8.0, ...
        ((b+1.0).*(a+b-a.^2-b.^2+2.0))./8.0-(a.*2.0-1.0).*(a./8.0+1.0./8.0).*(b+1.0), ...
        (a./8.0+1.0./8.0).*(a+b-a.^2-b.^2+2.0)-(a./8.0+1.0./8.0).*(b.*2.0-1.0).*(b+1.0), ...
        ((a+1.0).^2.*(b+1.0))./8.0+((a.*2.0+2.0).*(a-1.0).*(b+1.0))./8.0, ...
        ((a-1.0).*(a+1.0).^2)./8.0,((b-1.0).*(b+1.0).^2)./8.0, ...
        ((a+1.0).*(b+1.0).^2)./8.0+((b.*2.0+2.0).*(a+1.0).*(b-1.0))./8.0, ...
        ((b+1.0).*(a-b+a.^2+b.^2-2.0))./8.0+(a.*2.0+1.0).*(a./8.0-1.0./8.0).*(b+1.0), ...
        (a./8.0-1.0./8.0).*(a-b+a.^2+b.^2-2.0)+(a./8.0-1.0./8.0).*(b.*2.0-1.0).*(b+1.0), ...
        ((a-1.0).^2.*(b+1.0))./8.0+((a.*2.0-2.0).*(a+1.0).*(b+1.0))./8.0,((a-1.0).^2.*(a+1.0))./8.0, ...
        (b-1.0).*(b+1.0).^2.*(-1.0./8.0),(a-1.0).*(b+1.0).^2.*(-1.0./8.0)-((b.*2.0+2.0).*(a-1.0).*(b-1.0))./8.0],[2,12]);

elseif shapefcn == 3 || shapefcn == 4
    B = @(a,b)[-(a./8.0-1.0./8.0).*(b-1.0).*(a+b+a.^2+b.^2-2.0), ...
        (a-1.0).^2.*(a+1.0).*(b-1.0).*(-1.0./8.0),(a-1.0).*(b-1.0).^2.*(b+1.0).*(-1.0./8.0), ...
        (a./8.0+1.0./8.0).*(b-1.0).*(-a+b+a.^2+b.^2-2.0),(a-1.0).*(a+1.0).^2.*(b-1.0).*(-1.0./8.0), ...
        ((a+1.0).*(b-1.0).^2.*(b+1.0))./8.0,(a./8.0+1.0./8.0).*(b+1.0).*(a+b-a.^2-b.^2+2.0), ...
        ((a-1.0).*(a+1.0).^2.*(b+1.0))./8.0,((a+1.0).*(b-1.0).*(b+1.0).^2)./8.0, ...
        (a./8.0-1.0./8.0).*(b+1.0).*(a-b+a.^2+b.^2-2.0),((a-1.0).^2.*(a+1.0).*(b+1.0))./8.0, ...
        (a-1.0).*(b-1.0).*(b+1.0).^2.*(-1.0./8.0)];

elseif shapefcn == 1
    B = @(a,b)reshape([(a./4.0-1.0./4.0).*(b-1.0),0.0,0.0,(a./4.0-1.0./4.0).*(b-1.0), ...
        -(a./4.0+1.0./4.0).*(b-1.0),0.0,0.0,-(a./4.0+1.0./4.0).*(b-1.0),(a./4.0+1.0./4.0).*(b+1.0),0.0,0.0, ...
        (a./4.0+1.0./4.0).*(b+1.0),-(a./4.0-1.0./4.0).*(b+1.0),0.0,0.0,-(a./4.0-1.0./4.0).*(b+1.0)],[2,8]);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fout = f;
if ispoint == 1 %case of point load/moment
    split = BCval./(length(elem)); %redistribute the original value to all the attached elements
end
%Populate fout
syms x y
for i = 1:length(elem) %loop over all relevant elements
    x_e = nodecoord(connec(elem(i),2:5),2);
    y_e = nodecoord(connec(elem(i),2:5),3);
    dedx = 2/abs(x_e(2)-x_e(1));
    dndy = 2/abs(y_e(3)-y_e(2));
    rowpos1 = [connec(elem(i),2)*2-1,connec(elem(i),2)*2,connec(elem(i),3)*2-1,connec(elem(i),3)*2,connec(elem(i),4)*2-1,connec(elem(i),4)*2,connec(elem(i),5)*2-1,connec(
    rowpos2 = [connec(elem(i),2)*3-2,connec(elem(i),2)*3-1,connec(elem(i),2)*3,connec(elem(i),3)*3-2,connec(elem(i),3)*3-1,connec(elem(i),3)*3, ...
            connec(elem(i),4)*3-2,connec(elem(i),4)*3-1,connec(elem(i),4)*3,connec(elem(i),5)*3-2,connec(elem(i),5)*3-1,connec(elem(i),5)*3]+ 2*max(nodecoord(:,1));
    if ispoint == 0 %not a point load
        if ax == 0 %the case of distributed q load
            ftemp = zeros(12,1);
            if isa(BCval,'sym')
                for k = 1:p %loop over eta gauss points
                     for j = 1:p %loop over nu gauss points
                         % current values of x and y at Gauss points
                         xc = 0.25*[(1-eta(k))*(1-eta(j)),(1+eta(k))*(1-eta(j)),(1+eta(k))*(1+eta(j)),(1-eta(k))*(1+eta(j))]*x_e(:);
                         yc = 0.25*[(1-eta(k))*(1-eta(j)),(1+eta(k))*(1-eta(j)),(1+eta(k))*(1+eta(j)),(1-eta(k))*(1+eta(j))]*y_e(:);
                         J_det = (1/dedx)*(1/dndy);
                         % contribution to fout vector from current Gauss point
                         if xc < curpos{1}(1) || xc > curpos{1}(end) || yc < curpos{2}(1) || yc > curpos{2}(end)
                         else
                            Bsub = B(eta(k),eta(j));
                            ftemp = ftemp - double(w(k) * w(j) * J_det * Bsub' * subs(BCval,{x,y},{xc,yc}));
                         end
                     end
                end
            else
                for k = 1:p %loop over eta gauss points
                     for j = 1:p %loop over nu gauss points
                         xc = 0.25*[(1-eta(k))*(1-eta(j)),(1+eta(k))*(1-eta(j)),(1+eta(k))*(1+eta(j)),(1-eta(k))*(1+eta(j))]*x_e(:);
                         yc = 0.25*[(1-eta(k))*(1-eta(j)),(1+eta(k))*(1-eta(j)),(1+eta(k))*(1+eta(j)),(1-eta(k))*(1+eta(j))]*y_e(:);
                         J_det = (1/dedx)*(1/dndy);
                         %J_det = 1;
                         % contribution to fout vector from current Gauss point
                         if xc < curpos{1}(1) || xc > curpos{1}(end) || yc < curpos{2}(1) || yc > curpos{2}(end)
                         else
                            Bsub = B(eta(k),eta(j));
                            ftemp = ftemp + (1*BCval)*w(k) * w(j) * J_det * Bsub';   %12 by 1
                         end
                     end
```

```
        end
    end
    fout(rowpos2) = fout(rowpos2)+ftemp;
elseif ax==1 %boundary parallel to x-axis (constant y surface)
    if shapefcn == 1
        ftemp = zeros(8,1);
    elseif shapefcn == 3 || shapefcn == 2
        ftemp = zeros(12,1);
    end
    yc = curpos{2}; %y-value is constant
    nu = (2*curpos{2}-y_e(1)-y_e(4))/(y_e(4)-y_e(1));
    % current values of x and y at Gauss points
    J_det = (1/dedx);
    if isa(BCval,'sym')
        for k = 1:p %loop over eta gauss points
            % contribution to fout vector from current Gauss point
            xc = 0.25*[1-eta(k),1+eta(k),1+eta(k),1-eta(k)]*x_e(:);
            if xc < curpos{1}(1) || xc > curpos{1}(end)
            else
                if shapefcn == 2
                    %Bsub = subs(B,{a,b},{eta(k),nu});
                    Bsub = B(eta(k),nu);
                    Bsub(1,:) = Bsub(1,:)*dedx;
                    Bsub(2,:) = Bsub(2,:)*dndy;
                else
                    %Bsub = subs(B,{a,b},{eta(k),nu});
                    Bsub = B(eta(k),nu);
                end
                ftemp = ftemp + double(w(k) * J_det * Bsub' * subs(BCval,{x,y},{xc,yc}));
            end
        end
    else
        for k = 1:p %loop over eta gauss points
            % contribution to fout vector from current Gauss point
            xc = 0.25*[1-eta(k),1+eta(k),1+eta(k),1-eta(k)]*x_e(:);
            if xc < curpos{1}(1) || xc > curpos{1}(end)
            else
                if shapefcn == 2
                    Bsub = B(eta(k),nu);
                    Bsub(1,:) = Bsub(1,:)*dedx;
                    Bsub(2,:) = Bsub(2,:)*dndy;
                else
                    Bsub = B(eta(k),nu);
                end
                ftemp = ftemp + double(w(k)* J_det * Bsub')*BCval;
            end
        end
    end
    if shapefcn == 1
        fout(rowpos1) = fout(rowpos1)+ftemp;
    elseif shapefcn == 3 || shapefcn == 2
        fout(rowpos2) = fout(rowpos2)+ftemp;
    end
elseif ax == 2 %boundary parallel to y-axis
    if shapefcn == 1
        ftemp = zeros(8,1);
    elseif shapefcn == 3 || shapefcn == 2
        ftemp = zeros(12,1);
    end
    xc = curpos{1}; %x-value is constant
    nu = (2*curpos{1}-x_e(1)-x_e(2))/(x_e(2)-x_e(1));
    J_det = (1/dndy);
    if isa(BCval,'sym')
        for k = 1:p %loop over eta gauss points
            % current values of x and y at Gauss points
            yc = 0.25*[1-eta(k),1+eta(k),1+eta(k),1-eta(k)]*y_e(:);
            if yc < curpos{2}(1) || yc > curpos{2}(end)
            else
                if shapefcn == 2
                    Bsub = B(nu,eta(k));
                    Bsub(1,:) = Bsub(1,:)*dedx;
                    Bsub(2,:) = Bsub(2,:)*dndy;
                else
```

```
                        Bsub = B(nu,eta(k));
                    end
                    % contribution to fout vector from current Gauss point
                    ftemp = ftemp + double(w(k) * J_det * Bsub' * subs(BCval,{x,y},{xc,yc}));
                end
            end
        else
            for k = 1:p %loop over eta gauss points
                yc = 0.25*[1-eta(k),1+eta(k),1+eta(k),1-eta(k)]*y_e(:);
                if yc < curpos{2}(1) || yc > curpos{2}(end)
                else
                    if shapefcn == 2
                        Bsub = B(nu,eta(k));
                        Bsub(1,:) = Bsub(1,:)*dedx;
                        Bsub(2,:) = Bsub(2,:)*dndy;
                    else
                        Bsub = B(nu,eta(k));
                    end

                    % contribution to fout vector from current Gauss point
                    ftemp = ftemp + double(w(k)* J_det * Bsub')*BCval;
                end
            end
        end
        if shapefcn == 1
            fout(rowpos1) = fout(rowpos1)+ftemp;
        elseif shapefcn == 3 || shapefcn == 2
            fout(rowpos2) = fout(rowpos2)+ftemp;
        end
    end
elseif ispoint == 1 %point load/moment
    if shapefcn == 1
        ftemp = zeros(8,1);
    elseif shapefcn == 3 || shapefcn == 2 || shapefcn == 4
        ftemp = zeros(12,1);
    end
    yc = curpos{2}; %y-value is constant
    nuc = (2*yc-y_e(1)-y_e(4))/(y_e(4)-y_e(1));
    xc = curpos{1}; %x-value is constant
    etac = (2*xc-x_e(1)-x_e(2))/(x_e(2)-x_e(1));
    Bsub = B(etac,nuc);
    if shapefcn == 2 %case of point moment M
        Bsub(1,:) = Bsub(1,:)*dedx;
        Bsub(2,:) = Bsub(2,:)*dndy;
    end
    if shapefcn == 4  %case of point transverse load q
        ftemp = ftemp - double(Bsub' * split);
    elseif shapefcn == 3 || shapefcn == 1 || shapefcn == 2
        %case of point transverse shear Q, point in-plane load N, point moment M
        ftemp = ftemp + double(Bsub' * split);
    end
    if shapefcn == 1
        fout(rowpos1) = fout(rowpos1)+ftemp;
    elseif shapefcn == 3 || shapefcn == 2 || shapefcn == 4
        fout(rowpos2) = fout(rowpos2)+ftemp;
    end
end
end

end
```

constructF_con function

```
function [fout] = constructF_con(connec, nodecoord, f, curpos, BCval, ax, shapefcn, p)
%for non-conforming case
%input var: connec, nodecoord, nodes, f, curpos = BC.pos{i}, ax,
%shapefcn,p, BCval

%connec, nodecoord, nodes are taken from solve as it is
```

```
%f is the last updated force vector

%BCval is a syms or value vector/scalar

%curpos = BC.pos{i} from solve, it's a cell vector

%p: Gauss quadrature order

%shapefcn: associated shape function
%shapefcn = 1: in-plane stress loading(Nx,Ny,Nxy)
%shapefcn = 2: moments(Mx,My,Mxy)
%shapefcn = 3 and 4: transverse loading (Q,q)

%ax: integration boundary
%ax = 0: area of integration
%ax = 1: boundary parallel to x-axis (constant y)
%ax = 2: boundary parallel to y-axis (constant x)

%find all the relevant elements
elem = [];
ispoint = 0;

%assume the applied BC is either a point load/moment, or the length scale
%of the BC is at least the size of one element's dimension
for i = 1:size(connec,1) %loop over all elements to find relevant ones
    %find all elements attached to the node
    curelem = connec(i,:);
    flag = 0;
    if sum(curelem(1)==elem)==0 %make sure element is not a duplicate
        if length(curpos{1}) == 1 && length(curpos{2})==1 %point load case
            ispoint = 1;
            if curpos{1} >= nodecoord(curelem(2),2) && ...
                    curpos{1} <= nodecoord(curelem(3),2) && ...
                    curpos{2} >= nodecoord(curelem(2),3) && ...
                    curpos{2} <= nodecoord(curelem(4),3)
                %the point load application is with in the current element
            else
                %the point load is not in the current element
                flag = 1;
            end
        else
            if ax == 0 %distributed area loading
                %if at least 1 node connected to the element are within BC
                %application area, then leave flag to be 0
                if nodecoord(curelem(2),2) <= curpos{1}(end) && nodecoord(curelem(2),3) <= curpos{2}(end) ...
                    && nodecoord(curelem(2),2) >= curpos{1}(1) && nodecoord(curelem(2),3) >= curpos{2}(1)
                elseif nodecoord(curelem(3),2) <= curpos{1}(end) && nodecoord(curelem(3),3) <= curpos{2}(end) ...
                    && nodecoord(curelem(3),2) >= curpos{1}(1) && nodecoord(curelem(3),3) >= curpos{2}(1)
                elseif nodecoord(curelem(3),2) <= curpos{1}(end) && nodecoord(curelem(3),3) <= curpos{2}(end) ...
                    && nodecoord(curelem(3),2) >= curpos{1}(1) && nodecoord(curelem(3),3) >= curpos{2}(1)
                elseif nodecoord(curelem(4),2) <= curpos{1}(end) && nodecoord(curelem(4),3) <= curpos{2}(end) ...
                    && nodecoord(curelem(4),2) >= curpos{1}(1) && nodecoord(curelem(4),3) >= curpos{2}(1)
                else
                    %if none of nodes connected to the element is inside of BC
                    %boundary, then change flag to be 1
                    flag = 1;
                end
            elseif ax == 1 %distributed boundary line loading along x (constant y surface)
                if nodecoord(curelem(2),3) == curpos{2} || nodecoord(curelem(4),3) == curpos{2}
                    if nodecoord(curelem(2),2) <= curpos{1}(end) && nodecoord(curelem(2),2) >= curpos{1}(1)
                        %if the any of the nodes connected to the element is within BC
                        %x-axis boundary, then leave flag to be 0
                    elseif nodecoord(curelem(3),2) <= curpos{1}(end) && nodecoord(curelem(3),2) >= curpos{1}(1)
                    else
                        flag = 1;
                    end
                else
                    %otherwise, change flag to be 1
                    flag = 1;
                end
            elseif ax == 2 %distributed boundary line loading along y (constant x surface)
                if nodecoord(curelem(2),2) == curpos{1} || nodecoord(curelem(4),2) == curpos{1}
                    if nodecoord(curelem(2),3) <= curpos{2}(end) && nodecoord(curelem(2),3) >= curpos{2}(1)
```

```
                    %if the all nodes connected to the element is within BC
                    %boundary, then leave flag to be 0
                    elseif nodecoord(curelem(4),3) <= curpos{2}(end) && nodecoord(curelem(4),3) >= curpos{2}(1)
                    else
                        flag = 1;
                    end
                else
                    %if the any nodes connected to the element is outside of BC
                    %boundary, then change flag to be 1
                    flag = 1;
                end
            end
            %end
        end
        %if all nodes of the element is within BC boundary, then this
        %element will be apart of the Gauss quadrature computation
        if flag == 0
            elem = [elem;curelem(1)];
        end
    end
    %end
end

% set up Gauss quadrature
if p == 1
    eta = 0; %quadrature points
    w = 2; %quadrature weights
elseif p == 2
    eta = [-1/sqrt(3),1/sqrt(3)];
    w = [1,1];
elseif p == 3
    eta = [0,-sqrt(3/5),sqrt(3/5)];
    w = [8/9,5/9,5/9];
elseif p == 4
    eta = [-sqrt(3/7 - (2/7)*sqrt(6/5)),sqrt(3/7 - (2/7)*sqrt(6/5)),-sqrt(3/7 + (2/7)*sqrt(6/5)),sqrt(3/7 + (2/7)*sqrt(6/5))];
        w = [(18+sqrt(30))/36,(18+sqrt(30))/36,(18-sqrt(30))/36,(18-sqrt(30))/36];
end

%%%%%%%%%%%%%%%%%%%% shape function computation %%%%%%%%%%%%%%%%%%%%%%%%%
%shapefcn = 1: in-plane stress loading(Nx,Ny,Nxy)
%shapefcn = 2: moments(Mx,My,Mxy)
%shapefcn = 3 and 4: transverse loading (Q,q)
if shapefcn == 2
    %B is a 2 by 16 matrix
    B = @(a,b)reshape([((a-1.0).^2.*(b-1.0).^2.*(b+2.0))./1.6e+1+((a.*2.0-2.0).*(a+2.0).*(b-1.0).^2.*(b+2.0))./1.6e+1, ...
        ((a-1.0).^2.*(a+2.0).*(b-1.0).^2)./1.6e+1+((b.*2.0-2.0).*(a-1.0).^2.*(a+2.0).*(b+2.0))./1.6e+1, ...
        ((a-1.0).^2.*(b-1.0).^2.*(b+2.0))./1.6e+1+((a.*2.0-2.0).*(a+1.0).*(b-1.0).^2.*(b+2.0))./1.6e+1, ...
        ((a-1.0).^2.*(a+1.0).*(b-1.0).^2)./1.6e+1+((b.*2.0-2.0).*(a-1.0).^2.*(a+1.0).*(b+2.0))./1.6e+1, ...
        ((a-1.0).^2.*(b-1.0).^2.*(b+1.0))./1.6e+1+((a.*2.0-2.0).*(a+2.0).*(b-1.0).^2.*(b+1.0))./1.6e+1, ...
        ((a-1.0).^2.*(a+2.0).*(b-1.0).^2)./1.6e+1+((b.*2.0-2.0).*(a-1.0).^2.*(a+2.0).*(b+1.0))./1.6e+1, ...
        ((a-1.0).^2.*(b-1.0).^2.*(b+1.0))./1.6e+1+((a.*2.0-2.0).*(a+1.0).*(b-1.0).^2.*(b+1.0))./1.6e+1, ...
        ((a-1.0).^2.*(a+1.0).*(b-1.0).^2)./1.6e+1+((b.*2.0-2.0).*(a-1.0).^2.*(a+1.0).*(b+1.0))./1.6e+1, ...
        (a+1.0).^2.*(b-1.0).^2.*(b+2.0).*(-1.0./1.6e+1)-((a.*2.0+2.0).*(a-2.0).*(b-1.0).^2.*(b+2.0))./1.6e+1, ...
        (a+1.0).^2.*(a-2.0).*(b-1.0).^2.*(-1.0./1.6e+1)-((b.*2.0-2.0).*(a+1.0).^2.*(a-2.0).*(b+2.0))./1.6e+1, ...
        ((a+1.0).^2.*(b-1.0).^2.*(b+2.0))./1.6e+1+((a.*2.0+2.0).*(a-1.0).*(b-1.0).^2.*(b+2.0))./1.6e+1, ...
        ((a-1.0).*(a+1.0).^2.*(b-1.0).^2)./1.6e+1+((b.*2.0-2.0).*(a-1.0).*(a+1.0).^2.*(b+2.0))./1.6e+1, ...
        (a+1.0).^2.*(b-1.0).^2.*(b+1.0).*(-1.0./1.6e+1)-((a.*2.0+2.0).*(a-2.0).*(b-1.0).^2.*(b+1.0))./1.6e+1, ...
        (a+1.0).^2.*(a-2.0).*(b-1.0).^2.*(-1.0./1.6e+1)-((b.*2.0-2.0).*(a+1.0).^2.*(a-2.0).*(b+1.0))./1.6e+1, ...
        ((a+1.0).^2.*(b-1.0).^2.*(b+1.0))./1.6e+1+((a.*2.0+2.0).*(a-1.0).*(b-1.0).^2.*(b+1.0))./1.6e+1, ...
        ((a-1.0).*(a+1.0).^2.*(b-1.0).^2)./1.6e+1+((b.*2.0-2.0).*(a-1.0).*(a+1.0).^2.*(b+1.0))./1.6e+1, ...
        ((a+1.0).^2.*(b+1.0).^2.*(b-2.0))./1.6e+1+((a.*2.0+2.0).*(a-2.0).*(b+1.0).^2.*(b-2.0))./1.6e+1, ...
        ((a+1.0).^2.*(a-2.0).*(b+1.0).^2)./1.6e+1+((b.*2.0+2.0).*(a+1.0).^2.*(a-2.0).*(b-2.0))./1.6e+1, ...
        (a+1.0).^2.*(b+1.0).^2.*(b-2.0).*(-1.0./1.6e+1)-((a.*2.0+2.0).*(a-1.0).*(b+1.0).^2.*(b-2.0))./1.6e+1, ...
        (a-1.0).*(a+1.0).^2.*(b+1.0).^2.*(-1.0./1.6e+1)-((b.*2.0+2.0).*(a-1.0).*(a+1.0).^2.*(b-2.0))./1.6e+1, ...
        (a+1.0).^2.*(b-1.0).*(b+1.0).^2.*(-1.0./1.6e+1)-((a.*2.0+2.0).*(a-2.0).*(b-1.0).*(b+1.0).^2)./1.6e+1, ...
        (a+1.0).^2.*(a-2.0).*(b+1.0).^2.*(-1.0./1.6e+1)-((b.*2.0+2.0).*(a+1.0).^2.*(a-2.0).*(b-1.0))./1.6e+1, ...
        ((a+1.0).^2.*(b-1.0).*(b+1.0).^2)./1.6e+1+((a.*2.0+2.0).*(a-1.0).*(b-1.0).*(b+1.0).^2)./1.6e+1, ...
        ((a-1.0).*(a+1.0).^2.*(b+1.0).^2)./1.6e+1+((b.*2.0+2.0).*(a-1.0).*(a+1.0).^2.*(b-1.0))./1.6e+1, ...
        (a-1.0).^2.*(b+1.0).^2.*(b-2.0).*(-1.0./1.6e+1)-((a.*2.0-2.0).*(a+2.0).*(b+1.0).^2.*(b-2.0))./1.6e+1, ...
        (a-1.0).^2.*(a+2.0).*(b+1.0).^2.*(-1.0./1.6e+1)-((b.*2.0+2.0).*(a-1.0).^2.*(a+2.0).*(b-2.0))./1.6e+1, ...
        (a-1.0).^2.*(b+1.0).^2.*(b-2.0).*(-1.0./1.6e+1)-((a.*2.0-2.0).*(a+1.0).*(b+1.0).^2.*(b-2.0))./1.6e+1, ...
        (a-1.0).^2.*(a+1.0).*(b+1.0).^2.*(-1.0./1.6e+1)-((b.*2.0+2.0).*(a-1.0).^2.*(a+1.0).*(b-2.0))./1.6e+1, ...
        ((a-1.0).^2.*(b-1.0).*(b+1.0).^2)./1.6e+1+((a.*2.0-2.0).*(a+2.0).*(b-1.0).*(b+1.0).^2)./1.6e+1, ...
```

```
        ((a-1.0).^2.*(a+2.0).*(b+1.0).^2)./1.6e+1+((b.*2.0+2.0).*(a-1.0).^2.*(a+2.0).*(b-1.0))./1.6e+1, ...
        ((a-1.0).^2.*(b-1.0).*(b+1.0).^2)./1.6e+1+((a.*2.0-2.0).*(a+1.0).*(b-1.0).*(b+1.0).^2)./1.6e+1, ...
        ((a-1.0).^2.*(a+1.0).*(b+1.0).^2)./1.6e+1+((b.*2.0+2.0).*(a-1.0).^2.*(a+1.0).*(b-1.0))./1.6e+1],[2,16]);

elseif shapefcn == 3 || shapefcn == 4
    B =  @(a,b)[((a-1.0).^2.*(a+2.0).*(b-1.0).^2.*(b+2.0))./1.6e+1,((a-1.0).^2.*(a+1.0).*(b-1.0).^2.*(b+2.0))./1.6e+1, ...
        ((a-1.0).^2.*(a+2.0).*(b-1.0).^2.*(b+1.0))./1.6e+1,((a-1.0).^2.*(a+1.0).*(b-1.0).^2.*(b+1.0))./1.6e+1, ...
        (a+1.0).^2.*(a-2.0).*(b-1.0).^2.*(b+2.0).*(-1.0./1.6e+1),((a-1.0).*(a+1.0).^2.*(b-1.0).^2.*(b+2.0))./1.6e+1, ...
        (a+1.0).^2.*(a-2.0).*(b-1.0).^2.*(b+1.0).*(-1.0./1.6e+1),((a-1.0).*(a+1.0).^2.*(b-1.0).^2.*(b+1.0))./1.6e+1, ...
        ((a+1.0).^2.*(a-2.0).*(b+1.0).^2.*(b-2.0))./1.6e+1,(a-1.0).*(a+1.0).^2.*(b+1.0).^2.*(b-2.0).*(-1.0./1.6e+1), ...
        (a+1.0).^2.*(a-2.0).*(b-1.0).*(b+1.0).^2.*(-1.0./1.6e+1),((a-1.0).*(a+1.0).^2.*(b-1.0).*(b+1.0).^2)./1.6e+1, ...
        (a-1.0).^2.*(a+2.0).*(b+1.0).^2.*(b-2.0).*(-1.0./1.6e+1),(a-1.0).^2.*(a+1.0).*(b+1.0).^2.*(b-2.0).*(-1.0./1.6e+1), ...
        ((a-1.0).^2.*(a+2.0).*(b-1.0).*(b+1.0).^2)./1.6e+1,((a-1.0).^2.*(a+1.0).*(b-1.0).*(b+1.0).^2)./1.6e+1];

elseif shapefcn == 1
    B = @(a,b)reshape([(a./4.0-1.0./4.0).*(b-1.0),0.0,0.0,(a./4.0-1.0./4.0).*(b-1.0),-(a./4.0+1.0./4.0).*(b-1.0),0.0,0.0, ...
        -(a./4.0+1.0./4.0).*(b-1.0),(a./4.0+1.0./4.0).*(b+1.0),0.0,0.0,(a./4.0+1.0./4.0).*(b+1.0),-(a./4.0-1.0./4.0).*(b+1.0), ...
        0.0,0.0,-(a./4.0-1.0./4.0).*(b+1.0)],[2,8]);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fout = f;
if ispoint == 1 %case of point load/moment
    split = BCval./(length(elem)); %redistribute the original value to all the attached elements
end
%Populate fout
syms x y
for i = 1:length(elem) %loop over all relevant elements
    x_e = nodecoord(connec(elem(i),2:5),2);
    y_e = nodecoord(connec(elem(i),2:5),3);
    dedx = 2/abs(x_e(2)-x_e(1));
    dndy = 2/abs(y_e(3)-y_e(2));
    rowpos1 = [connec(elem(i),2)*2-1,connec(elem(i),2)*2,connec(elem(i),3)*2-1,connec(elem(i),3)*2,connec(elem(i),4)*2-1,connec(elem(i),4)*2,connec(elem(i),5)*2-1,connec(
    rowpos2 = [connec(elem(i),2)*4-3,connec(elem(i),2)*4-2,connec(elem(i),2)*4-1,connec(elem(i),2)*4,connec(elem(i),3)*4-3,connec(elem(i),3)*4-2,connec(elem(i),3)*4-1,con
            connec(elem(i),4)*4-3,connec(elem(i),4)*4-2,connec(elem(i),4)*4-1,connec(elem(i),4)*4,connec(elem(i),5)*4-3,connec(elem(i),5)*4-2,connec(elem(i),5)*4-1,connec
    if ispoint == 0 %not a point load
        if ax == 0 %the case of distributed q load
            ftemp = zeros(16,1);
            if isa(BCval,'sym')
                for k = 1:p %loop over eta gauss points
                     for j = 1:p %loop over nu gauss points
                         % current values of x and y at Gauss points
                         xc = 0.25*[(1-eta(k))*(1-eta(j)),(1+eta(k))*(1-eta(j)),(1+eta(k))*(1+eta(j)),(1-eta(k))*(1+eta(j))]*x_e(:);
                         yc = 0.25*[(1-eta(k))*(1-eta(j)),(1+eta(k))*(1-eta(j)),(1+eta(k))*(1+eta(j)),(1-eta(k))*(1+eta(j))]*y_e(:);
                         J_det = (1/dedx)*(1/dndy);
                         % contribution to fout vector from current Gauss point
                         if xc < curpos{1}(1) || xc > curpos{1}(end) || yc < curpos{2}(1) || yc > curpos{2}(end)
                         else
                            Bsub = B(eta(k),eta(j));
                            %minus because of the negative sign in front of q term
                            ftemp = ftemp - double(w(k) * w(j) * J_det * Bsub' * subs(BCval,{x,y},{xc,yc}));
                         end
                     end
                end
            else
                for k = 1:p %loop over eta gauss points
                     for j = 1:p %loop over nu gauss points
                         xc = 0.25*[(1-eta(k))*(1-eta(j)),(1+eta(k))*(1-eta(j)),(1+eta(k))*(1+eta(j)),(1-eta(k))*(1+eta(j))]*x_e(:);
                         yc = 0.25*[(1-eta(k))*(1-eta(j)),(1+eta(k))*(1-eta(j)),(1+eta(k))*(1+eta(j)),(1-eta(k))*(1+eta(j))]*y_e(:);
                         J_det = (1/dedx)*(1/dndy);
                         % contribution to fout vector from current Gauss point
                         if xc < curpos{1}(1) || xc > curpos{1}(end) || yc < curpos{2}(1) || yc > curpos{2}(end)
                         else
                            J_det = (1/dedx)*(1/dndy);
                            Bsub = B(eta(k),eta(j));
                            ftemp = ftemp + (-1*BCval)*double(w(k) * w(j) * J_det * Bsub');   %12 by 1
                         end
                     end
                end
            end
            fout(rowpos2) = fout(rowpos2)+ftemp;
        elseif ax==1 %boundary parallel to x-axis (constant y surface)
            if shapefcn == 1
```

```
        ftemp = zeros(8,1);
    elseif shapefcn == 3 || shapefcn == 2
        ftemp = zeros(16,1);
    end
    yc = curpos{2}; %y-value is constant
    nu = (2*curpos{2}-y_e(1)-y_e(4))/(y_e(4)-y_e(1));
    % current values of x and y at Gauss points
    J_det = (1/dedx);
    if isa(BCval,'sym')
        for k = 1:p %loop over eta gauss points
            % contribution to fout vector from current Gauss point
            xc = 0.25*[1-eta(k),1+eta(k),1+eta(k),1-eta(k)]*x_e(:);

            if xc < curpos{1}(1) || xc > curpos{1}(end)
            else
                if shapefcn == 2
                    Bsub = B(eta(k),nu);
                    Bsub(1,:) = Bsub(1,:)*dedx;
                    Bsub(2,:) = Bsub(2,:)*dndy;
                else
                    Bsub = B(eta(k),nu);
                end
                ftemp = ftemp + double(w(k) * J_det * Bsub' * subs(BCval,{x,y},{xc,yc}));
            end
        end
    else
        for k = 1:p %loop over eta gauss points
            % contribution to fout vector from current Gauss point
            xc = 0.25*[1-eta(k),1+eta(k),1+eta(k),1-eta(k)]*x_e(:);
            if xc < curpos{1}(1) || xc > curpos{1}(end)
            else
                if shapefcn == 2
                    Bsub = B(eta(k),nu);
                    Bsub(1,:) = Bsub(1,:)*dedx;
                    Bsub(2,:) = Bsub(2,:)*dndy;
                else
                    Bsub = B(eta(k),nu);
                end
                ftemp = ftemp + double(w(k)* J_det * Bsub')*BCval;
            end
        end
    end
    if shapefcn == 1
        fout(rowpos1) = fout(rowpos1)+ftemp;
    elseif shapefcn == 3 || shapefcn == 2
        fout(rowpos2) = fout(rowpos2)+ftemp;
    end
elseif ax == 2 %boundary parallel to y-axis
    if shapefcn == 1
        ftemp = zeros(8,1);
    elseif shapefcn == 3 || shapefcn == 2
        ftemp = zeros(16,1);
    end
    xc = curpos{1}; %x-value is constant
    nu = (2*curpos{1}-x_e(1)-x_e(2))/(x_e(2)-x_e(1));
    J_det = (1/dndy);
    if isa(BCval,'sym')
        for k = 1:p %loop over eta gauss points
            % current values of x and y at Gauss points
            yc = 0.25*[1-eta(k),1+eta(k),1+eta(k),1-eta(k)]*y_e(:);
            if yc < curpos{2}(1) || yc > curpos{2}(end)
            else
                if shapefcn == 2
                    Bsub = B(nu,eta(k));
                    Bsub(1,:) = Bsub(1,:)*dedx;
                    Bsub(2,:) = Bsub(2,:)*dndy;
                else
                    Bsub = B(nu,eta(k));
                end
                % contribution to fout vector from current Gauss point
                ftemp = ftemp + double(w(k) * J_det * Bsub' * subs(BCval,{x,y},{xc,yc}));
            end
        end
```

```
            else
                for k = 1:p %loop over eta gauss points
                    yc = 0.25*[1-eta(k),1+eta(k),1+eta(k),1-eta(k)]*y_e(:);
                    if yc < curpos{2}(1) || yc > curpos{2}(end)
                    else
                         if shapefcn == 2
                             Bsub = B(nu,eta(k));
                             Bsub(1,:) = Bsub(1,:)*dedx;
                             Bsub(2,:) = Bsub(2,:)*dndy;
                         else
                             Bsub = B(nu,eta(k));
                         end

                         % contribution to fout vector from current Gauss point
                         ftemp = ftemp + (w(k)* J_det * Bsub')*BCval;
                    end
                end
            end
            if shapefcn == 1
                fout(rowpos1) = fout(rowpos1)+ftemp;
            elseif shapefcn == 3 || shapefcn == 2
                fout(rowpos2) = fout(rowpos2)+ftemp;
            end
        end
    elseif ispoint == 1 %point load/moment
        if shapefcn == 1
            ftemp = zeros(8,1);
        elseif shapefcn == 3 || shapefcn == 2 || shapefcn == 4
            ftemp = zeros(16,1);
        end
        yc = curpos{2}; %y-value is constant
        nuc = (2*yc-y_e(1)-y_e(4))/(y_e(4)-y_e(1));
        xc = curpos{1}; %x-value is constant
        etac = (2*xc-x_e(1)-x_e(2))/(x_e(2)-x_e(1));
        %Bsub = subs(B,{a,b},{etac,nuc});
        Bsub = B(etac,nuc);
        if shapefcn == 2 %case of point moment M
            Bsub(1,:) = Bsub(1,:)*dedx;
            Bsub(2,:) = Bsub(2,:)*dndy;
        end
        if shapefcn == 4  %case of point transverse load q
            ftemp = ftemp - double(Bsub' * split);
        elseif shapefcn == 3 || shapefcn == 1 || shapefcn == 2
            %case of point transverse shear Q, point in-plane load N, point moment M
            ftemp = ftemp + double(Bsub' * split);
        end
        if shapefcn == 1
            fout(rowpos1) = fout(rowpos1)+ftemp;
        elseif shapefcn == 3 || shapefcn == 2 || shapefcn == 4
            fout(rowpos2) = fout(rowpos2)+ftemp;
        end
    end
end

end
```

routine for investigating zero energy mode

```
transpts = [-1,-1;-1,1;1,-1;1,1];
syms a b ai bi
nw = (1/16)*(a+ai)^2*(a*ai-2)*(b+bi)^2*(b*bi-2);
ntx = (1/16)*ai*(a+ai)^2*(1-a*ai)*(b+bi)^2*(b*bi-2);
nty = (1/16)*bi*(a+ai)^2*(a*ai-2)*(b+bi)^2*(1-b*bi);
ntxy = (1/16)*ai*bi*(a+ai)^2*(1-a*ai)*(b+bi)^2*(1-b*bi);
for i = 1:4 %loop over 4 nodes to construct shape fcn
    %B is a 1 by 16 matrix
    B(1,i*4-3) = subs(nw,{ai,bi},{transpts(i,:)});
    B(1,i*4-2) = subs(ntx,{ai,bi},{transpts(i,:)});
    B(1,i*4-1) = subs(nty,{ai,bi},{transpts(i,:)});
    B(1,i*4) = subs(ntxy,{ai,bi},{transpts(i,:)});
```

```
end

bsub = B*rxy;

dedx = 2/2;
dndy = 2/2;
for i = 1:4 %loop over 4 nodes to construct shape fcn derivatives
    %Hw is a 3 by 16 matrix
    Hw(1,i*4-3) = diff(subs(nw,{ai,bi},{transpts(i,:)}),a,2)*dedx^2;
    Hw(1,i*4-2) = diff(subs(ntx,{ai,bi},{transpts(i,:)}),a,2)*dedx^2;
    Hw(1,i*4-1) = diff(subs(nty,{ai,bi},{transpts(i,:)}),a,2)*dedx^2;
    Hw(1,i*4) = diff(subs(ntxy,{ai,bi},{transpts(i,:)}),a,2)*dedx^2;

    Hw(2,i*4-3) = diff(subs(nw,{ai,bi},{transpts(i,:)}),b,2)*dndy^2;
    Hw(2,i*4-2) = diff(subs(ntx,{ai,bi},{transpts(i,:)}),b,2)*dndy^2;
    Hw(2,i*4-1) = diff(subs(nty,{ai,bi},{transpts(i,:)}),b,2)*dndy^2;
    Hw(2,i*4) = diff(subs(ntxy,{ai,bi},{transpts(i,:)}),b,2)*dndy^2;

    Hw(3,i*4-3) = diff(diff(subs(nw,{ai,bi},{transpts(i,:)}),a,1),b,1)*dndy*dedx*2;
    Hw(3,i*4-2) = diff(diff(subs(ntx,{ai,bi},{transpts(i,:)}),a,1),b,1)*dndy*dedx*2;
    Hw(3,i*4-1) = diff(diff(subs(nty,{ai,bi},{transpts(i,:)}),a,1),b,1)*dndy*dedx*2;
    Hw(3,i*4) = diff(diff(subs(ntxy,{ai,bi},{transpts(i,:)}),a,1),b,1)*dndy*dedx*2;
end
p = 2;
eta = [-1/sqrt(3),1/sqrt(3)];
w = [1,1];
for i = 1:p %loop over eta gauss points
     for j = 1:p %loop over nu gauss points
        Hwsub = subs(Hw,{a,b},{eta(i),eta(j)});
        strain(i,j,:) = double(Hwsub*rxy); %strain at all the gauss points
     end
end
%the strain variable should have all zero values, which means the strain at
%all the Gauss Quadrature points are zero, hence "zero energy mode".
figure,
ezsurf(bsub,[-1,1],[-1,1]);
title('CON Zero Energy Mode, int pts = 2, 1x1 mesh')
xlabel('X-Position'), ylabel('Y-Position'), zlabel('Z-Position')
```

graphing function

```
function graphing(connec,nodecoord,d,ele,f)
%Input variable:
%ele: 1- non-conforming element
%     2- conforming element

% plotting code deformed surface contour
fh(1) = figure(1);
title('Deformed Mesh')
ax(1) = axes('parent',fh(1));
fh(2) = figure(2);
title('U Displacement Map')
ax(2) = axes('parent',fh(2));
fh(3) = figure(3);
title('V Displacement Map')
ax(3) = axes('parent',fh(3));
fh(4) = figure(4);
title('W Displacement Map')
ax(4) = axes('parent',fh(4));
figure,
for i = 1:size(connec,1) %loop over elements
    x_e = nodecoord(connec(i,2:5),2);
    y_e = nodecoord(connec(i,2:5),3);

    xtrans = @(x,x_e) (2*x-x_e(1)-x_e(2))/(x_e(2)-x_e(1)); %this is a
    ytrans = @(y,y_e) (2*y-y_e(1)-y_e(4))/(y_e(4)-y_e(1)); %this is b

    %Bxy is a 2 by 8 matrix (in-plane shape functions)
    Bxy = @(a,b)reshape([(a./4.0-1.0./4.0).*(b-1.0),0.0,0.0,(a./4.0-1.0./4.0).*(b-1.0),-(a./4.0+1.0./4.0).*(b-1.0),0.0,0.0, ...
        -(a./4.0+1.0./4.0).*(b-1.0),(a./4.0+1.0./4.0).*(b+1.0),0.0,0.0,(a./4.0+1.0./4.0).*(b+1.0),-(a./4.0-1.0./4.0).*(b+1.0), ...
```

```
    0.0,0.0,-(a./4.0-1.0./4.0).*(b+1.0)],[2,8]);

if ele == 1
    %Bw is a 1 by 12 matrix
    Bw = @(a,b)[-(a./8.0-1.0./8.0).*(b-1.0).*(a+b+a.^2+b.^2-2.0),(a-1.0).^2.*(a+1.0).*(b-1.0).*(-1.0./8.0), ...
        (a-1.0).*(b-1.0).^2.*(b+1.0).*(-1.0./8.0),(a./8.0+1.0./8.0).*(b-1.0).*(-a+b+a.^2+b.^2-2.0), ...
        (a-1.0).*(a+1.0).^2.*(b-1.0).*(-1.0./8.0),((a+1.0).*(b-1.0).^2.*(b+1.0))./8.0, ...
        (a./8.0+1.0./8.0).*(b+1.0).*(a+b-a.^2-b.^2+2.0),((a-1.0).*(a+1.0).^2.*(b+1.0))./8.0, ...
        ((a+1.0).*(b-1.0).*(b+1.0).^2)./8.0,(a./8.0-1.0./8.0).*(b+1.0).*(a-b+a.^2+b.^2-2.0), ...
        ((a-1.0).^2.*(a+1.0).*(b+1.0))./8.0,(a-1.0).*(b-1.0).*(b+1.0).^2.*(-1.0./8.0)];

elseif ele == 2
%Bw is a 1 by 16 matrix
    Bw = @(a,b)[((a-1.0).^2.*(a+2.0).*(b-1.0).^2.*(b+2.0))./1.6e+1,((a-1.0).^2.*(a+1.0).*(b-1.0).^2.*(b+2.0))./1.6e+1, ...
        ((a-1.0).^2.*(a+2.0).*(b-1.0).^2.*(b+1.0))./1.6e+1,((a-1.0).^2.*(a+1.0).*(b-1.0).^2.*(b+1.0))./1.6e+1, ...
        (a+1.0).^2.*(a-2.0).*(b-1.0).^2.*(b+2.0).*(-1.0./1.6e+1),((a-1.0).*(a+1.0).^2.*(b-1.0).^2.*(b+2.0))./1.6e+1, ...
        (a+1.0).^2.*(a-2.0).*(b-1.0).^2.*(b+1.0).*(-1.0./1.6e+1),((a-1.0).*(a+1.0).^2.*(b-1.0).^2.*(b+1.0))./1.6e+1, ...
        ((a+1.0).^2.*(a-2.0).*(b+1.0).^2.*(b-2.0))./1.6e+1,(a-1.0).*(a+1.0).^2.*(b+1.0).^2.*(b-2.0).*(-1.0./1.6e+1), ...
        (a+1.0).^2.*(a-2.0).*(b-1.0).*(b+1.0).^2.*(-1.0./1.6e+1),((a-1.0).*(a+1.0).^2.*(b-1.0).*(b+1.0).^2)./1.6e+1, ...
        (a-1.0).^2.*(a+2.0).*(b+1.0).^2.*(b-2.0).*(-1.0./1.6e+1),(a-1.0).^2.*(a+1.0).*(b+1.0).^2.*(b-2.0).*(-1.0./1.6e+1), ...
        ((a-1.0).^2.*(a+2.0).*(b-1.0).*(b+1.0).^2)./1.6e+1,((a-1.0).^2.*(a+1.0).*(b-1.0).*(b+1.0).^2)./1.6e+1];
end

xbin_s = linspace(x_e(1),x_e(2),100);
ybin_s = linspace(y_e(1),y_e(4),100);
[xbin,ybin] = meshgrid(xbin_s,ybin_s);

curelem = connec(i,2:5);
for m = 1:length(ybin_s)
    for n = 1:length(xbin_s)
        xydef = Bxy(xtrans(xbin_s(n),x_e),ytrans(ybin_s(m),y_e));
        xydef = xydef*[d(curelem(1)*2-1:curelem(1)*2);d(curelem(2)*2-1:curelem(2)*2);...
                    d(curelem(3)*2-1:curelem(3)*2);d(curelem(4)*2-1:curelem(4)*2)];
        xdefplot(n,m) = xydef(1);
        ydefplot(n,m) = xydef(2);

        zdef = Bw(xtrans(xbin_s(n),x_e),ytrans(ybin_s(m),y_e));
        if ele == 1
            zdefplot(n,m) = zdef*[d(2*max(nodecoord(:,1))+curelem(1)*3-2:2*max(nodecoord(:,1))+curelem(1)*3);...
                d(2*max(nodecoord(:,1))+curelem(2)*3-2:2*max(nodecoord(:,1))+curelem(2)*3);...
                d(2*max(nodecoord(:,1))+curelem(3)*3-2:2*max(nodecoord(:,1))+curelem(3)*3);...
                d(2*max(nodecoord(:,1))+curelem(4)*3-2:2*max(nodecoord(:,1))+curelem(4)*3)];
        elseif ele == 2
            zdefplot(n,m) = zdef*[d(2*max(nodecoord(:,1))+curelem(1)*4-3:2*max(nodecoord(:,1))+curelem(1)*4);...
                d(2*max(nodecoord(:,1))+curelem(2)*4-3:2*max(nodecoord(:,1))+curelem(2)*4);...
                d(2*max(nodecoord(:,1))+curelem(3)*4-3:2*max(nodecoord(:,1))+curelem(3)*4);...
                d(2*max(nodecoord(:,1))+curelem(4)*4-3:2*max(nodecoord(:,1))+curelem(4)*4)];
        end
        magdefplot(n,m) = sqrt(xydef(1)^2 + xydef(2)^2 + zdefplot(n,m)^2);
    end
end
xdefplotd = xdefplot + xbin;
ydefplotd = ydefplot + ybin;
xdefplot = subs(xydef(1,:),{x,y},{xbin,ybin})+xbin;
ydefplot = subs(xydef(2,:),{x,y},{xbin,ybin})+ybin;
zdefplot = subs(zdef,{x,y},{xbin,ybin});

surf(double(xdefplotd)',double(ydefplotd)',double(zdefplot),'EdgeColor','none','Parent',ax(1)),
hold on,
%hold(ax(1),'on'),
surf(xbin,ybin,xdefplot','EdgeColor','none','Parent',ax(2)),
hold on,
%hold(ax(2),'on'),
surf(xbin,ybin,ydefplot','EdgeColor','none','Parent',ax(3)),
hold on,
%hold(ax(3),'on'),
surf(xbin,ybin,zdefplot','EdgeColor','none','Parent',ax(4)),
hold on,
%hold(ax(4),'on'),

%use the part below if you only want to plot the magnitude of
%deformation. You must comment all the above graphing codes.
```

```
    %surf(xbin,ybin,magdefplot','EdgeColor','none'),
    %hold on,
end
set(fh(1),'Name','Deformed Mesh');
set(ax(2), 'xTickLabel',' ')
set(ax(2), 'yTickLabel',' ')
set(ax(2), 'zTickLabel',' ')

set(ax(3), 'xTickLabel',' ')
set(ax(3), 'yTickLabel',' ')
set(ax(3), 'zTickLabel',' ')

set(ax(4), 'xTickLabel',' ')
set(ax(4), 'yTickLabel',' ')
set(ax(4), 'zTickLabel',' ')

%use the part below if you only want to label the plot of the magnitude of
%deformation. You must comment all the above graph labelling codes.
%title('Displacement Magnitude Map')
%title('z Displacement Map')
%xlabel('X-Position')
%ylabel('Y-Position')


% plotting code reaction force contour
fh(5) = figure(5);
title('X Reaction Force')
ax(5) = axes('parent',fh(5));
fh(6) = figure(6);
title('Y Reaction Force')
ax(6) = axes('parent',fh(6));
fh(7) = figure(7);
title('Z Reaction Force')
ax(7) = axes('parent',fh(7));

for i = 1:size(connec,1) %loop over elements
    x_e = nodecoord(connec(i,2:5),2);
    y_e = nodecoord(connec(i,2:5),3);

    xtrans = @(x,x_e) (2*x-x_e(1)-x_e(2))/(x_e(2)-x_e(1)); %this is a
    ytrans = @(y,y_e) (2*y-y_e(1)-y_e(4))/(y_e(4)-y_e(1)); %this is b

    %Bxy is a 2 by 8 matrix (in-plane shape functions)
    Bxy = @(a,b)reshape([(a./4.0-1.0./4.0).*(b-1.0),0.0,0.0,(a./4.0-1.0./4.0).*(b-1.0),-(a./4.0+1.0./4.0).*(b-1.0),0.0,0.0, ...
        -(a./4.0+1.0./4.0).*(b-1.0),(a./4.0+1.0./4.0).*(b+1.0),0.0,0.0,(a./4.0+1.0./4.0).*(b+1.0),-(a./4.0-1.0./4.0).*(b+1.0), ...
        0.0,0.0,-(a./4.0-1.0./4.0).*(b+1.0)],[2,8]);

    if ele == 1
        %Bw is a 1 by 12 matrix
        Bw = @(a,b)[-(a./8.0-1.0./8.0).*(b-1.0).*(a+b+a.^2+b.^2-2.0),(a-1.0).^2.*(a+1.0).*(b-1.0).*(-1.0./8.0), ...
            (a-1.0).*(b-1.0).^2.*(b+1.0).*(-1.0./8.0),(a./8.0+1.0./8.0).*(b-1.0).*(-a+b+a.^2+b.^2-2.0), ...
            (a-1.0).*(a+1.0).^2.*(b-1.0).*(-1.0./8.0),((a+1.0).*(b-1.0).^2.*(b+1.0))./8.0, ...
            (a./8.0+1.0./8.0).*(b+1.0).*(a+b-a.^2-b.^2+2.0),((a-1.0).*(a+1.0).^2.*(b+1.0))./8.0, ...
            ((a+1.0).*(b-1.0).*(b+1.0).^2)./8.0,(a./8.0-1.0./8.0).*(b+1.0).*(a-b+a.^2+b.^2-2.0), ...
            ((a-1.0).^2.*(a+1.0).*(b+1.0))./8.0,(a-1.0).*(b-1.0).*(b+1.0).^2.*(-1.0./8.0)];

    elseif ele == 2
    %Bw is a 1 by 16 matrix
        Bw = @(a,b)[((a-1.0).^2.*(a+2.0).*(b-1.0).^2.*(b+2.0))./1.6e+1,((a-1.0).^2.*(a+1.0).*(b-1.0).^2.*(b+2.0))./1.6e+1, ...
            ((a-1.0).^2.*(a+2.0).*(b-1.0).^2.*(b+1.0))./1.6e+1,((a-1.0).^2.*(a+1.0).*(b-1.0).^2.*(b+1.0))./1.6e+1, ...
            (a+1.0).^2.*(a-2.0).*(b-1.0).^2.*(b+2.0).*(-1.0./1.6e+1),((a-1.0).*(a+1.0).^2.*(b-1.0).^2.*(b+2.0))./1.6e+1, ...
            (a+1.0).^2.*(a-2.0).*(b-1.0).^2.*(b+1.0).*(-1.0./1.6e+1),((a-1.0).*(a+1.0).^2.*(b-1.0).^2.*(b+1.0))./1.6e+1, ...
            ((a+1.0).^2.*(a-2.0).*(b+1.0).^2.*(b-2.0))./1.6e+1,(a-1.0).*(a+1.0).^2.*(b+1.0).^2.*(b-2.0).*(-1.0./1.6e+1), ...
            (a+1.0).^2.*(a-2.0).*(b-1.0).*(b+1.0).^2.*(-1.0./1.6e+1),((a-1.0).*(a+1.0).^2.*(b-1.0).*(b+1.0).^2)./1.6e+1, ...
            (a-1.0).^2.*(a+2.0).*(b+1.0).^2.*(b-2.0).*(-1.0./1.6e+1),(a-1.0).^2.*(a+1.0).*(b+1.0).^2.*(b-2.0).*(-1.0./1.6e+1), ...
            ((a-1.0).^2.*(a+2.0).*(b-1.0).*(b+1.0).^2)./1.6e+1,((a-1.0).^2.*(a+1.0).*(b-1.0).*(b+1.0).^2)./1.6e+1];
    end

    xbin_s = linspace(x_e(1),x_e(2),100);
    ybin_s = linspace(y_e(1),y_e(4),100);
    [xbin,ybin] = meshgrid(xbin_s,ybin_s);

    curelem = connec(i,2:5);
```

```
    for m = 1:length(ybin_s)
        for n = 1:length(xbin_s)
            xyfrea = Bxy(xtrans(xbin_s(n),x_e),ytrans(ybin_s(m),y_e));
            xyfrea = xyfrea*[f(curelem(1)*2-1:curelem(1)*2);f(curelem(2)*2-1:curelem(2)*2);...
                        f(curelem(3)*2-1:curelem(3)*2);f(curelem(4)*2-1:curelem(4)*2)];
            xfreaplot(n,m) = xyfrea(1);
            yfreaplot(n,m) = xyfrea(2);

            zfrea = Bw(xtrans(xbin_s(n),x_e),ytrans(ybin_s(m),y_e));
            if ele == 1
                zfreaplot(n,m) = zfrea*[f(2*max(nodecoord(:,1))+curelem(1)*3-2:2*max(nodecoord(:,1))+curelem(1)*3);...
                    f(2*max(nodecoord(:,1))+curelem(2)*3-2:2*max(nodecoord(:,1))+curelem(2)*3);...
                    f(2*max(nodecoord(:,1))+curelem(3)*3-2:2*max(nodecoord(:,1))+curelem(3)*3);...
                    f(2*max(nodecoord(:,1))+curelem(4)*3-2:2*max(nodecoord(:,1))+curelem(4)*3)].*10^2;
            elseif ele == 2
                zfreaplot(n,m) = zfrea*[f(2*max(nodecoord(:,1))+curelem(1)*4-3:2*max(nodecoord(:,1))+curelem(1)*4);...
                    f(2*max(nodecoord(:,1))+curelem(2)*4-3:2*max(nodecoord(:,1))+curelem(2)*4);...
                    f(2*max(nodecoord(:,1))+curelem(3)*4-3:2*max(nodecoord(:,1))+curelem(3)*4);...
                    f(2*max(nodecoord(:,1))+curelem(4)*4-3:2*max(nodecoord(:,1))+curelem(4)*4)].*10^2;
            end

        end
    end
    surf(xbin,ybin,xfreaplot','EdgeColor','none','Parent',ax(5)),
    hold on,
    hold(ax(5),'on'),
    surf(xbin,ybin,yfreaplot','EdgeColor','none','Parent',ax(6)),
    hold on,
    hold(ax(6),'on'),
    surf(xbin,ybin,zfreaplot','EdgeColor','none','Parent',ax(7)),
    hold on,
    hold(ax(7),'on'),
end
set(ax(5), 'xTickLabel',' ')
set(ax(5), 'yTickLabel',' ')
set(ax(5), 'zTickLabel',' ')

set(ax(6), 'xTickLabel',' ')
set(ax(6), 'yTickLabel',' ')
set(ax(6), 'zTickLabel',' ')

set(ax(7), 'xTickLabel',' ')
set(ax(7), 'yTickLabel',' ')
set(ax(7), 'zTickLabel',' ')

% plotting codes node-wise
newcoord = zeros(size(nodecoord));
newcoord(:,1) = nodecoord(:,1);
for i = 1:size(nodecoord,1)
%change the "1" value to whatever scale you desire.

newcoord(i,2) = 1*d(2*i-1)+nodecoord(i,2); %xcoord
newcoord(i,3) = 1*d(2*i)+nodecoord(i,3); %ycoord
if ele == 1
    newcoord(i,4) = 1*d(2*max(nodecoord(:,1))+3*i-2); %zcoord
elseif ele == 2
    newcoord(i,4) = 1*d(2*max(nodecoord(:,1))+4*i-3); %zcoord
end
end
%plotting the deformed mesh and boundary force
figure,
for i = 1:size(connec,1)
    plot3([newcoord(connec(i,2),2),newcoord(connec(i,3),2)],[newcoord(connec(i,2),3),newcoord(connec(i,3),3)],[newcoord(connec(i,2),4),newcoord(connec(i,3),4)],'b');
    hold on,
    plot3([newcoord(connec(i,3),2),newcoord(connec(i,4),2)],[newcoord(connec(i,3),3),newcoord(connec(i,4),3)],[newcoord(connec(i,3),4),newcoord(connec(i,4),4)],'b','Handl
    hold on,
    plot3([newcoord(connec(i,4),2),newcoord(connec(i,5),2)],[newcoord(connec(i,4),3),newcoord(connec(i,5),3)],[newcoord(connec(i,4),4),newcoord(connec(i,5),4)],'b','Handl
    hold on,
    plot3([newcoord(connec(i,2),2),newcoord(connec(i,5),2)],[newcoord(connec(i,2),3),newcoord(connec(i,5),3)],[newcoord(connec(i,2),4),newcoord(connec(i,5),4)],'b','Handl

    hold on,
    plot([nodecoord(connec(i,2),2),nodecoord(connec(i,3),2)],[nodecoord(connec(i,2),3),nodecoord(connec(i,3),3)],'r');
    hold on,
```

```
    plot([nodecoord(connec(i,3),2),nodecoord(connec(i,4),2)],[nodecoord(connec(i,3),3),nodecoord(connec(i,4),3)],'r');
    hold on,
    plot([nodecoord(connec(i,4),2),nodecoord(connec(i,5),2)],[nodecoord(connec(i,4),3),nodecoord(connec(i,5),3)],'r');
    hold on,
    plot([nodecoord(connec(i,2),2),nodecoord(connec(i,5),2)],[nodecoord(connec(i,2),3),nodecoord(connec(i,5),3)],'r');
end
legend('Deformed Mesh','Undeformed Mesh','Location','Best')
title('Mesh Nodal Displacement Map')
xlabel('X-Position')
ylabel('Y-Position')
zlabel('Z-Position')

end
```