Design of Lightweight Composite Structures

Report I

Kaya Onur Dag
s111058

Department of Wind Energy
Technical University of Denmark (DTU)
April 2013

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This document so far contains a report for the exercises for the course *Design of Lightweight Composite Structures* read in the Department of Wind Energy, Technical University of Denmark.

In the appendix, most of the Matlab functions and scripts, which had been developed, can be found. In most of the figure scripts, LaTeX interpreter was used and thus it may be useful to turn the interpreter on.

Chapter 2

Micro-mechanics of Composites

2.1 Thickness of an unspecified laminate type using ROM

2.1.1 Given area masses and densities

In order to calculate the thickness of an unspecified laminate type using Rule-of-Mixtures, the area masses and densities needs to be found. With these data, thickness can be calculated via Eq. 2.1.

$$h = \sum_i \frac{W_i^*}{\rho_i} \tag{2.1}$$

where W_i^* is weight per unit area and ρ_i is the density for material i. To make the usage easier, stated equation is written in a Matlab function which can be found in Appendix A.1.1.

2.1.2 Given fibre volume fraction, densities and total area mass

For calculating the thickness of a composite, it is needed to have weight fraction which can easily be used with the Eq. 2.1. To calculate weight fractions for each material, Eq. 2.2 can be used.

$$w_i = \frac{\rho_i \upsilon_i}{\sum \rho_i \upsilon_i} \tag{2.2}$$

where υ_i is the volume fraction and w_i is the weight fraction for material i. After this point, since the weight fraction is known for all the materials, weight per unit area W_i, can be found via given *total area mass* and from the Eq. 2.1, the value for total thickness can be calculated.

The Matlab function is stated in Appendix A.1.1. Advantage of having such a Matlab function here is, for a complex layout with many materials, different volume fractions

and densities for each, total thickness can easily be calculated without having an extra for-loop from the main code.

2.2 Stiffnesses, Shear Modulus and Poisson Ration Calculation with ROM

For a UD-laminate, $E1$, $E2$ and $G12$ modulusses can be calculated with ROM. For determination of $E1$, it is assumed that strains are equal on the fibre direction for the matrix and fibres. From this assumption Eq.2.3 can be written.

$$P = \sigma_1 A_1 = \sigma_f A_f + \sigma_m A_m = E_f \varepsilon_f A_f + E_m \varepsilon_m A_m = \varepsilon_1 (E_f A_f + E_m A_m) \tag{2.3}$$

where σ_1 is the total stress for a specific volume element which consist of fibre and matrix area, A_f and A_m, at the cross-section and E_f and E_m are modulusses for fibre and matrix respectively. By substituting these with $A_{total} = A_f + A_m$, the $E1$ can be calculated as shown in Eq. 2.4.

$$E_1 = E_f \upsilon_f + E_m \upsilon_m \tag{2.4}$$

The key assumption here is having the same elongation on the fibre and matrix material. With this approximation the occured method called *parallel model / Voigt approximation* where we have E_1 as a function of υ_i.

Nevertheless, for E_2, the modulus on the transverse direction, the governing equation is based on assumption where we assume to have equal stresses. With respect to this assumption, we end up with Eq.2.5 with *serial model / Reuss approximation.* These two different approximations are shown in Fig.2.1.

$$\frac{1}{E_2} = \frac{\upsilon_f}{E_f} + \frac{\upsilon_m}{E_m} \tag{2.5}$$

For determination of shear modulus, G_{12}, we assume that the shear stress acting on the fibres and the matrix are equal. With this assumption, Eq.2.6 can be derived.

$$\frac{1}{G_{12}} = \frac{\upsilon_f}{G_f} + \frac{\upsilon_m}{G_m} \tag{2.6}$$

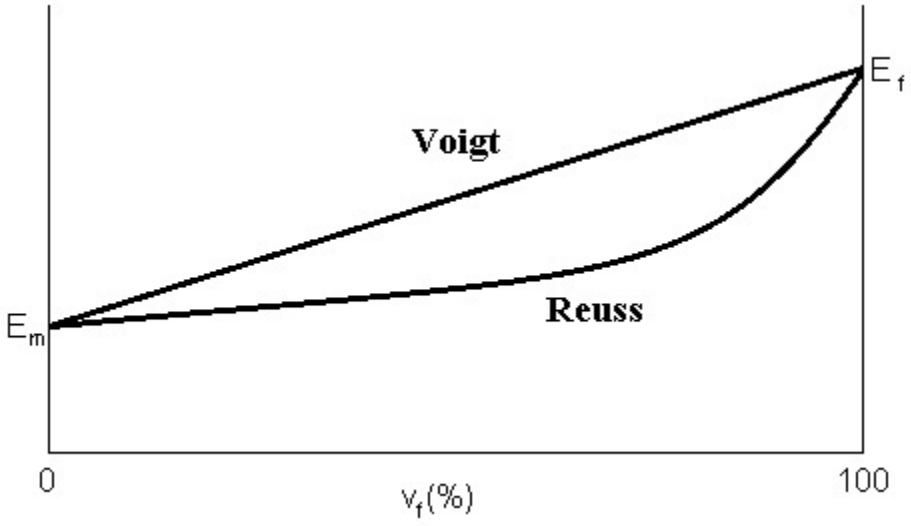


Figure 2.1: Voigt and Ruess approximations versus υ_f volume friction

We use the serial model to determine the Poisson ratio, ν_{12}. The elongation on transverse direction can be written as;

$$\Delta_2 = h\varepsilon_2 = -h\nu_{12}\varepsilon_1 \tag{2.7}$$

$$\Delta_2 = \Delta_{2f} + \Delta_{2m} = -h_f\nu_f\varepsilon_1 - h_m\nu_m\varepsilon_1 \tag{2.8}$$

and if we substitute $\frac{h_i}{h} = \upsilon_i$ we end up with Eq.2.9.

$$\nu_{12} = \upsilon_f\nu_f + \upsilon_m\nu_m \tag{2.9}$$

where υ is the volume friction and ν is the Poisson ratio.

All these four main equations are implemented in a Matlab function shown in Appendix A.1.2.

2.3 In-Plane Stiffnesses with Efficiency Factor

The efficiency factor becomes useful when we deal with composite laminates built up from fabrics and unidirectional materials. For the usage, the implementation to the R-O-M method is straight forward. For each ply, we take the efficiency value into account as stated in the Eq.2.10

$$E_1 = \sum_i \alpha_i\upsilon_i E_i \tag{2.10}$$

where α is the efficiency factor and the υ is the volume friction. For such an calculation, the Matlab function is also attached in Appendix A.1.3.

2.4 Coefficients of Thermal and Moisture Expansion for a UD-Laminate

For estimation of thermal and moisture properties the used equations are shown above. For the thermal coefficient and strain relation Eq. 2.11 can be used.

$$\varepsilon_1 = \alpha_1 \Delta T, \varepsilon_2 = \alpha_2 \Delta T \tag{2.11}$$

After the derivations, the governing equations for α_1 and α_2 can be found.

$$\alpha_1 = \frac{E_{f1}\alpha_{f1}\upsilon_f + E_m\alpha_m\upsilon_m}{E_1} \tag{2.12}$$

$$\alpha_2 = \alpha_{2f}\upsilon_f(1+\nu_{12}\frac{\alpha_{1f}}{\alpha_{2f}}) + \alpha_m\upsilon_m(1+\nu_{12m}\frac{\alpha_{1m}}{\alpha_{2m}}) - (\nu_{12f}\upsilon_f + \nu_{12m}\upsilon_m)\frac{(E\alpha)_1}{E_1} \tag{2.13}$$

where α_{1x}, α_{2x} are axial and transverse coefficients of thermal expansions respectively. The f letter stands for fibre and m for matrix properties. The ν_{12f} and ν_{12m} are fibre and matrix Poision ratios in 12-direction respectively.

For micromechanical relations for coefficient of moisture expansion β can be calculated with an assumption which states that fibres usually do not absorb any moisture. Thus the coefficients for fibres become $\beta_f = \beta_{1f} = \beta_{2f} = 0$.

At the end of derivations, for an isotropic matrix Eq.2.14 for β_1 and Eq.2.15 for β_2 can be found.

$$\beta_1 = \beta_m\frac{\upsilon_m E_m}{E_1} \tag{2.14}$$

$$\beta_2 = \beta_m\frac{\upsilon_m}{E_1}[E_{f1}\upsilon_f(1+\nu_m) + E_m(\upsilon_m - \nu_{12f}\upsilon_f)] \tag{2.15}$$

where β_{1x}, β_{2x} are axial and transverse coefficients of moisture expansions respectively. The f letter stands for fibre and m for matrix properties.

At the end, the total hygrothermal strains, which includes moisture and thermal effects, becomes as shown in Eq.2.16,Eq.2.17 and Eq.2.18.

$$\varepsilon_1^{HT} = \alpha_1\Delta T + \beta_1\Delta c \tag{2.16}$$

$$\varepsilon_2^{HT} = \alpha_2\Delta T + \beta 2\Delta c \tag{2.17}$$

$$\gamma_{12}HT = 0 \tag{2.18}$$

where ΔT and Δc are temperature and moisture content changes respectively. The function for this equation can be found in Appendix A.1.4.

2.5 Validation of the Voigt and Ruess Calculations

The example 2.1 and 2.2 from the book has been solved and for validation of the matlab function, the plot has been used from the exercise sheet. The results are compared with the Fig.21 and exact same values have been obtained. The calculation is made with 1% step size on volume friction. Obtained values can be seen from the Fig.2.2.

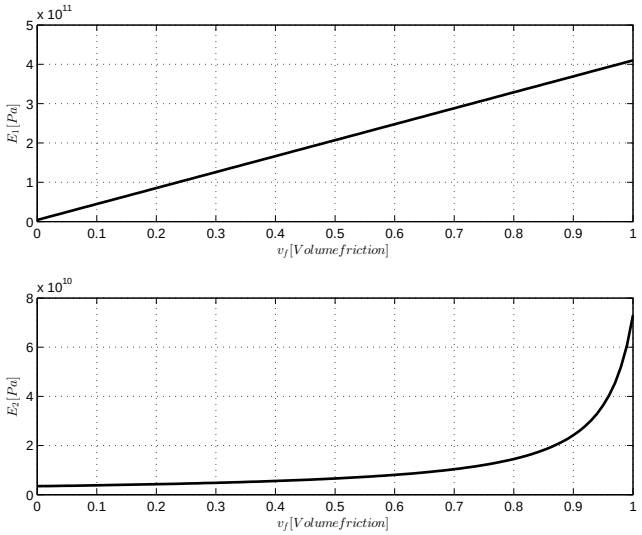


Figure 2.2: E_1 and E_2 calculations with Matlab function

Chapter 3

Mechanics of a Lamina

In this chapter calculations of a lamina is presented. In the first part, some parameters will be found via given properties of Glass-epoxy material (see the Table 3.1). Unlike the previous chapter, the lamina has already been generated thus no need to handle the stand-alone properties of fibre (glass in this case) or matrix(epoxy resin) material and calculate the properties with ROM. Furthermore, detail explanations of the generated Matlab functions are shown in the second part and their validation is presented in the last part.

3.1 Part 1

$E_1 = 54GPa$	$\nu_{12} = 0.25$	$G_{12} = 9GPa$
$E_2 = 18GPa$	$\nu_{13} = 0.25$	$G_{13} = 9GPa$
$E_3 = 18GPa$	$\nu_{23} = 0.30$	$G_{13} = 15GPa$

Table 3.1: Given glass-epoxy material properties

3.1.1 Calculation of Poisson Ratios

For the Poisson ratio calculation, Eq.3.1 is used from the lecture book.

$$\frac{\nu_{ij}}{E_i} = \frac{\nu_{ji}}{E_j} \tag{3.1}$$

where ν is the Poisson ratio and E is the modulus value for corresponding direction. Therefore, the requested Poisson ratios can be found as shown below.

$$\nu_{21} = E_2 \frac{\nu_{12}}{E_1} = 0.0833 \tag{3.2}$$

$$\nu_{31} = E_3 \frac{\nu_{13}}{E_1} = 0.0833 \tag{3.3}$$

$$\nu_{32} = E_3 \frac{\nu_{23}}{E_2} = 0.3 \tag{3.4}$$

3.1.2 Strain Responses for Given Loads

For each of these three listed load cases, the calculations has been done with a Matlab code as shown below.

```
%% Load input
sigma=[s1 s2 s3 s23 s31 s12]
%% compliance matrix generation
[ S ] = compliance3(E1,E2,E3,G12,G23,G31,nu12,nu21,nu31,nu13,nu23,nu32)
%% strain calculation
eps = S * sigma'
```

The *compliance*3 is a function which generates the 3 dimensional compliance matrix for given values. The code for this function is also provided in Appendix A.2.1.

$\sigma_1 = 20\ [MPa]$, $\sigma_2 = \sigma_3 = 0$

$\varepsilon_1 = 0.37\text{E-}3\ [m]$
$\varepsilon_2 = -0.09\text{E-}3\ [m]$
$\varepsilon_3 = -0.09\text{E-}3\ [m]$
$\varepsilon_{23} = 0$
$\varepsilon_{31} = 0$
$\varepsilon_{12} = 0$

$\sigma_2 = 20\ [MPa]$, $\sigma_1 = \sigma_3 = 0$

$\varepsilon_1 = -0.092\text{E-}3\ [m]$
$\varepsilon_2 = 1.11\text{E-}3\ [m]$
$\varepsilon_3 = -0.33\text{E-}3\ [m]$
$\varepsilon_{23} = 0$
$\varepsilon_{31} = 0$
$\varepsilon_{12} = 0$

$\sigma_1 = \sigma_2 = 20\ [MPa]$, $\sigma_3 = 0$

$\varepsilon_1 = -0.092\text{E-}3\ [m]$
$\varepsilon_2 = 1.11\text{E-}3\ [m]$
$\varepsilon_3 = -0.33\text{E-}3\ [m]$
$\varepsilon_{23} = 0$
$\varepsilon_{31} = 0$

$\varepsilon_{12} = 0$

3.1.3 Stress Responses for Given Strain

For the solution of given case, instead of a stiffness matrix, a compliance matrix is used with a backslash operator.

```
%% Given strain
eps=[1e-3 0 0 0 0 0 ]
%% compliance matrix generation
[ S ] = compliance3(E1,E2,E3,G12,G23,G31,nu12,nu21,nu31,nu13,nu23,nu32)
%% stress calculation
sigma=S\eps'
```

The obtained results for the generated stresses are shown above.

$\sigma_1 = 5.74\text{E}7\ [Pa]$
$\sigma_2 = 0.68\text{E}7\ [Pa]$
$\sigma_3 = 0.68\text{E}7\ [Pa]$
$\sigma_{23} = 0$
$\sigma_{31} = 0$
$\sigma_{12} = 0$

3.2 Part 2

There are three main Matlab functions used in this part. In order to be able generate the stiffness matrices easily, a function called *stiffness2*, as shown in Appendix A.2.2, is generated. The inputs to this function is as shown below.

```
[ Q ] = stiffness2(E1, E2, G12, nu12)
```

For the purpose of being able to use this functions in the future with laminates, a transformation function between global and local coordinate systems was needed. For this aim, a stiffness matrix transformer is generated as shown in Appendix A.2.2 and the inputs are the stiffness matrix Q at the local coordinate system and the angle θ in degrees between global and local coordinate systems. The usage is shown below.

```
[ Qg ] = stiffness_matrix_rotater(Q,theta)
```

For a given stress in global coordinates, strains can be calculated with a backslash solver via global stiffness matrix. However, in order to get the local elongations, a rotation function for strains were needed. For this purpose, functions called *rotate2_strain* for XY plane and *rotate3_strain* for XYZ plane are generated. The inputs to these functions are shown below and codes are presented in Appendix A.2.2.

3.3 Validation

The validation for the functions are done with examples from the lecture book. For stiffness matrix generation, example 3.3 is used and the calculated strain values were the same. Moreover, the validation for rotation functions are done with example 3.1 from the lecture book and the results are presented below.

In example 3.1, from a given material stiffness and compliance matrices are calculated. Furthermore, the effect of the rotation on the stiffness matrix has been calculated with 1 degree step size and plotted in the Fig.3.1 and Fig.3.2 as regeneration of the Figure 3.5 from the lecture book.

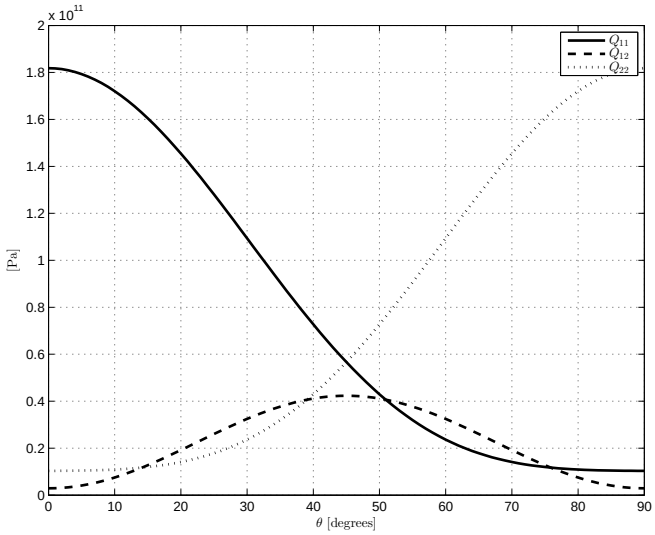


Figure 3.1: Q_{11},Q_{12},Q_{22} as a function of rotation angle θ

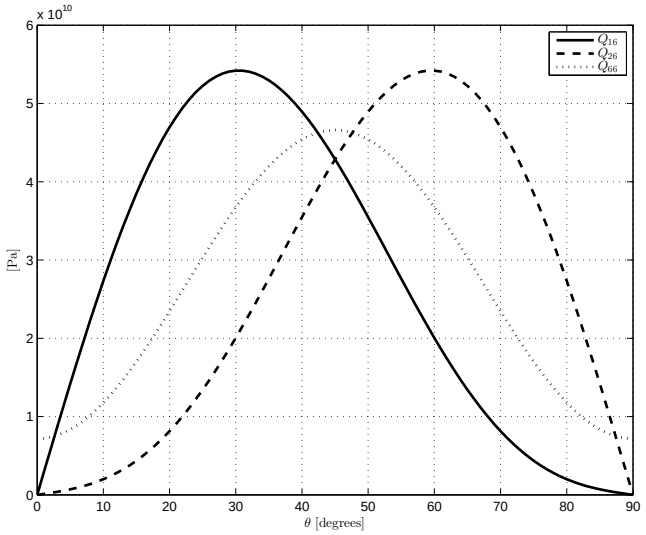


Figure 3.2: Q_{16},Q_{26},Q_{66} as a function of rotation angle θ

Chapter 4

Mechanics of a Laminate

In this chapter, mechanic properties of a laminate structure and its behavior under a specific load case, shown in Table 4, has been investigated. Detailed descriptions for calculations and coded functions are described below. The laminate structure is shown in the Fig.4.1. In order to be able to use the Matlab function in appropriate way, the numbering of the layers are adjusted and re-numbering is done from top to the bottom as also stated in the lecture book.

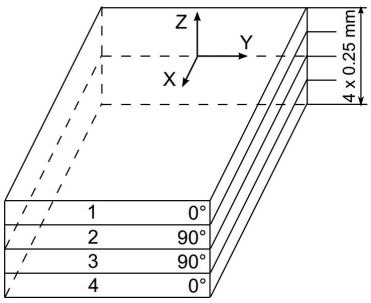


Figure 4.1: Laminate configuration

$N_x = 300N/mm$	$M_x = 50N$
$N_y = -50N/mm$	$M_y = 0N$
$N_{xy} = 25N/mm$	$M_{xy} = 0N$

Table 4.1: Specified load case

The mechanical properties of the material being used is:

$$E_1 = 46GPa \quad E_2 = 13GPa \quad G_{12} = 4.4GPa \quad \nu_{12} = 0.3$$

4.1 Determination of Stiffness Matrix

To determine the stiffness matrix for a lamina layer, the stiffness matrix generator function is used. Since each of these layers are made from the same material the

stiffness matrices are the same. Nonetheless, due to the differences between principal axes of these layers the rotation of these stiffness matrices is needed.

With the *stiffness2* function, the calculated Q_L matrix is as shown below.

$$Q_L = \begin{bmatrix} 4.72 & 0.40 & 0 \\ 0.40 & 1.33 & 0 \\ 0 & 0 & 0.44 \end{bmatrix} 10^{10}[Pa]$$

4.2 Transformation of Stiffness Matrices to Global Coordinates

To make this transformation, the stiffness matrix rotater function is used. After the calculations, the stiffness matrices on the global coordinate system are shown below.

$$Q_1 = \begin{bmatrix} 4.72 & 0.40 & 0 \\ 0.40 & 1.33 & 0 \\ 0 & 0 & 0.44 \end{bmatrix} 10^{4}[MPa]$$

$$Q_2 = \begin{bmatrix} 1.33 & 0.40 & 0 \\ 0.40 & 4.72 & 0 \\ 0 & 0 & 0.44 \end{bmatrix} 10^{4}[MPa]$$

$$Q_3 = \begin{bmatrix} 1.33 & 0.40 & 0 \\ 0.40 & 4.72 & 0 \\ 0 & 0 & 0.44 \end{bmatrix} 10^{4}[MPa]$$

$$Q_4 = \begin{bmatrix} 4.72 & 0.40 & 0 \\ 0.40 & 1.33 & 0 \\ 0 & 0 & 0.44 \end{bmatrix} 10^{4}[MPa]$$

For this calculation each single stiffness matrix on its own local coordinate system is transformed to global coordinate system with corresponding angle.

4.3 Determination of Stiffness Matrices of Laminate

The laminate is a group of arbitrarily oriented laminae from a plate or shell. Due to this orientation, to define a single specific stiffness matrix to whole laminate structure, all of the stiffness matrices of each layer need to be taken into account. This big stiffness matrix, which includes extensional stiffness, extension-bending coupling and bending stiffness matrices is shown below.

$$\begin{bmatrix} N_x \\ N_y \\ N_{xy} \\ M_x \\ M_y \\ M_{xy} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{16} & B_{11} & B_{12} & B_{16} \\ A_{12} & A_{22} & A_{26} & B_{12} & B_{22} & B_{26} \\ A_{16} & A_{26} & A_{66} & B_{16} & B_{26} & B_{66} \\ B_{11} & B_{12} & B_{16} & D_{11} & D_{12} & D_{16} \\ B_{12} & B_{22} & B_{26} & D_{12} & D_{22} & D_{26} \\ B_{16} & B_{26} & B_{66} & D_{16} & D_{26} & D_{66} \end{bmatrix} \begin{bmatrix} \varepsilon_{x0} \\ \varepsilon_{y0} \\ \gamma_{xy0} \\ \kappa_x \\ \kappa_y \\ \kappa_{xy} \end{bmatrix}$$

where A,B and D stand for *extensional*, *coupled* and *bending* stiffness matrices respectively as can also be seen from the multiplication part of the right hand side from the matrix system.

In a simple form the matrix system can written as shown below.

$$\begin{bmatrix} N \\ M \end{bmatrix} = \begin{bmatrix} A & B \\ B & D \end{bmatrix} \begin{bmatrix} \varepsilon_0 \\ \kappa \end{bmatrix}$$

For generation of A, B and D matrices, the equation stated in the lecture book used to generate the function called *ABDgenerator*. This Matlab function, which is presented in Appendix A.3.1, is based on the Eq.4.1.

$$[A, B, D] = \sum_{i=1}^{n} Q_i[(z_i - z_{i-1}), \frac{1}{2}(z_i^2 - z_{i-1}^2), \frac{1}{3}(z_i^3 - z_{i-1}^3)] \tag{4.1}$$

From the calculations the results are shown below.

$$A = \begin{bmatrix} 3.0270 & 0.4002 & 0 \\ 0.4002 & 3.0270 & 0 \\ 0 & 0 & 0.4400 \end{bmatrix} 10^4 \quad [N/mm]$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad [N]$$

$$D = \begin{bmatrix} 0.3581 & 0.0333 & 0 \\ 0.0333 & 0.1464 & 0 \\ 0 & 0 & 0.0367 \end{bmatrix} 10^4 \quad [N\ mm]$$

$$ABD = \begin{bmatrix} 3.0270 & 0.4002 & 0 & 0 & 0 & 0 \\ 0.4002 & 3.0270 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.4400 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.3581 & 0.0333 & 0 \\ 0 & 0 & 0 & 0.0333 & 0.1464 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0367 \end{bmatrix} 10^4$$

As a result of a symmetric cross-ply laminate structure, the B matrix consists of zeros. Which also means that there is no coupling between elongation and curvature.

4.4 Determination of Compliance Matrices of Laminate

The compliance matrices are the inverse form of stiffness matrices. Basically, when we multiply elongations and curvatures with a stiffness matrix, we end up with applied moments and forces. However, on the other hand, in order to reach the resulted elongations and magnitudes of bendings, a compliance matrix is needed which to be multiplied with forces and moments for reaching the goal.

The compliance matrix of shown system is calculated via taking inverse form of the ABD matrix.

$$[ABD]^{-1} = \begin{bmatrix} 0.0336 & -0.0044 & 0 & 0 & 0 & 0 \\ -0.0044 & 0.0336 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.2273 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.2853 & -0.0650 & 0 \\ 0 & 0 & 0 & -0.0650 & 0.6977 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2.7273 \end{bmatrix} 10^3$$

Nevertheless, it was preferred to use Matlab's *backslash operator* with ABD matrix instead.

4.5 Determination of Strains in Global Coordinates

As described above, with the given load case, a *backslash operator* is used with the ABD matrix and results were as shown.

$$\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \end{bmatrix} = \begin{bmatrix} 0.0103 \\ -0.0030 \end{bmatrix} \quad [mm]$$

The full result with curvatures is as shown;

$$\begin{bmatrix} \varepsilon_{x0} \\ \varepsilon_{y0} \\ \gamma_{xy0} \\ \kappa_x \\ \kappa_y \\ \kappa_{xy} \end{bmatrix} = \begin{bmatrix} 0.0103 \\ -0.0030 \\ 0.0057 \\ 0.0143 \\ -0.0032 \\ 0 \end{bmatrix}$$

4.6 Determination of Strains in Principal Lamina Directions

The main strains in global x and y axes are stated for the whole structure. However, for each lamina, the curvatures are also needs to be taken into account. The curvatures were already calculated for the structure and specified load case. A curvature value is a non-dimensional multiplier which results as an elongation due to the occurred bending with respect to the distance to the middle plane on z-direction, or in other words radius, r.

The usage of curvature κ is shown in Eq.4.2.

$$\varepsilon = \varepsilon_0 + r\kappa \tag{4.2}$$

The calculations are being made as shown for each of the lamina. For the calculations, the following Matlab code is used.

```
%inputs [if not specificly stated, Units-->pressure[Pa], length[m], ...
    Force[N]  ]
E1=46e9;
E2=13e9;
G12=4.4e9;
nu12=0.3;
n=4; %number of layers
seq=[0 90 90 0]; %sequence
h=0.25; %mm
z=[-2*h -h 0 h 2*h]; %heights of each layer
Nx=300;Ny=-50;Nxy=25; % N/mm
Mx=50;My=0;Mxy=0;     % N
Q1 = stiffness2(E1, E2, G12, nu12);
Q1=Q1/10^6; %Pascal to MPascal
%Rotate stiffness matrices to global coordinate sys.
for i = 1:n
    Q(:,:,i) = stiffness_matrix_rotater(Q1,seq(i));
end
%Calculate the ABD matrix for whole structure
[ ABD,A,B,D ] = ABDgenerator(Q,z,n);
%Calculate the strains and curvatures with backslash solver
s_xy=ABD\[Nx Ny Nxy Mx My Mxy]'; % total strains on the middle axis _xy
%Get the total strains for each lamina on their own axes.
for i = 1:n
    m=mean([z(i) z(i+1)]); %middle axes of each ply
    sum_strain_xy(:,:,i)=[s_xy(1)+s_xy(4)*m; %eps + z * curvature
        s_xy(2)+s_xy(5)*m;
        s_xy(3)+s_xy(6)*m];
    strain_12(:,:,i)=rotate2_strain(-seq(i))*sum_strain_xy(:,:,i); ...
```

```
        %xy to 12 coordinate system
29    sigma_12(:,:,i)=Q1*strain_12(:,:,i); % stresses on each ply on ...
        principle lamina coordinates
30  end
```

The total strains on each lamina were ready in the global axes. In order to transfer these information to the principal axes, the *rotate2_strain* function, which is described in previous chapter and shown in AppendixA.2.2, is used. The calculation and usage procedure can be seen inside the code block shown above.

$$\begin{bmatrix} \varepsilon_{x1} \\ \varepsilon_{y1} \\ \gamma_{xy1} \end{bmatrix} = \begin{bmatrix} 0.0050 \\ -0.0018 \\ 0.0057 \end{bmatrix} \begin{matrix} [mm] \\ [mm] \end{matrix}$$

$$\begin{bmatrix} \varepsilon_{x2} \\ \varepsilon_{y2} \\ \gamma_{xy2} \end{bmatrix} = \begin{bmatrix} -0.0026 \\ 0.0085 \\ -0.0057 \end{bmatrix} \begin{matrix} [mm] \\ [mm] \end{matrix}$$

$$\begin{bmatrix} \varepsilon_{x3} \\ \varepsilon_{y3} \\ \gamma_{xy3} \end{bmatrix} = \begin{bmatrix} -0.0034 \\ 0.0121 \\ -0.0057 \end{bmatrix} \begin{matrix} [mm] \\ [mm] \end{matrix}$$

$$\begin{bmatrix} \varepsilon_{x4} \\ \varepsilon_{y4} \\ \gamma_{xy4} \end{bmatrix} = \begin{bmatrix} 0.0157 \\ -0.0042 \\ 0.0057 \end{bmatrix} \begin{matrix} [mm] \\ [mm] \end{matrix}$$

The layers are ordered from top to bottom as shown in Fig.4.1.

4.7 Determination of Stresses for Each Lamina in Principal Lamina Directions

For calculating strains, the stiffness matrices were used. From the previous section, the local strains were found. With respect to these magnitudes, it was possible to calculate the strains via stiffness matrices.

For each lamina, the multiplication of strains with corresponding stiffness matrix has given the stress values. In addition, since whole layers were same, the stiffness matrices were same. The only issue was to have the local strains which was already calculated in the previous chapter.

```
1  for i = 1:n
```

```
    sigma_12(:,:,i)=Q1*strain_12(:,:,i); % stresses on each ply on ...
        principle lamina coordinates
end
```

The results are shown below.

$$\begin{bmatrix}\sigma_{x1}\\ \sigma_{y1}\\ \tau_{xy1}\end{bmatrix} = \begin{bmatrix}226.9000\\ -4.1155\\ 25.0000\end{bmatrix} \begin{matrix}[MPa]\\ [MPa]\\ [MPa]\end{matrix}$$

$$\begin{bmatrix}\sigma_{x2}\\ \sigma_{y2}\\ \tau_{xy2}\end{bmatrix} = \begin{bmatrix}-89.0085\\ 103.2926\\ -25.0000\end{bmatrix} \begin{matrix}[MPa]\\ [MPa]\\ [MPa]\end{matrix}$$

$$\begin{bmatrix}\sigma_{x3}\\ \sigma_{y3}\\ \tau_{xy3}\end{bmatrix} = \begin{bmatrix}-113.0744\\ 147.6184\\ -25.0000\end{bmatrix} \begin{matrix}[MPa]\\ [MPa]\\ [MPa]\end{matrix}$$

$$\begin{bmatrix}\sigma_{x4}\\ \sigma_{y4}\\ \tau_{xy4}\end{bmatrix} = \begin{bmatrix}722.1889\\ 6.1985\\ 25.0000\end{bmatrix} \begin{matrix}[MPa]\\ [MPa]\\ [MPa]\end{matrix}$$

Normally, for such a symmetric system one would expect to have the same strain and stress values on mirrored layers. Nevertheless, by taking curvatures into account, it is obvious that the strains from bending is relatively high.

4.8 Validation

The validation of the function *ABDgenerator* is done with some examples from the lecture book. The examples numbered 3.6 and 3.7 were used. In example 3.6 the A matrix was necessary. On the other hand, in example 3.7 D matrix was used to calculate the bendings. From these two examples, it was easy to conclude that the function *ABDgenerator* was working properly. The codes are attached in the AppendixA.3.2.

Chapter 5

Strength of Laminates

In this chapter, strength of the laminate structure and failure criteria are discussed. In the first part, four different failure criteria are described. The application of these methods are handled and calculations are done in the following sections. At the end, a detail explanation of last ply failure algorithm is shown.

In the Table5.1, the maximum strength properties are shown which are used for calculations below for the specified lamina.

$\hat{\sigma}_{1t}$	1000 MPa
$\hat{\sigma}_{1c}$	1000 MPa
$\hat{\sigma}_{2t}$	35 MPa
$\hat{\sigma}_{2c}$	150 MPa
$\hat{\tau}_{12}$	50 MPa
$\hat{\varepsilon}_{1t}$	$\hat{\sigma}_{1t}/E_1$
$\hat{\varepsilon}_{1c}$	$\hat{\sigma}_{1c}/E_1$
$\hat{\varepsilon}_{2t}$	$\hat{\sigma}_{2t}/E_2$
$\hat{\varepsilon}_{2c}$	$\hat{\sigma}_{2c}/E_2$
$\hat{\gamma}_{12}$	$\hat{\tau}_{12}/G_{12}$

Table 5.1: Given lamina strength properties

5.1 Failure Criteria

5.1.1 Maximum Stress Criterion

The calculated stresses for specific material, σ_1, σ_2 and τ_{12}, must be inside the limits given. In other words, the occurred stress components must satisfy following equations, where $\hat{\sigma}_{1c}$ and $\hat{\sigma}_{1t}$ are the maximum compression and tensile stresses and $\hat{\tau}_{12}$ is the maximum shear stress.

$$-\hat{\sigma}_{1c} < \sigma_1 < \hat{\sigma}_{1t} \tag{5.1}$$

$$-\hat{\sigma}_{2c} < \sigma_2 < \hat{\sigma}_{2t} \tag{5.2}$$

$$|\tau_{12}| < \hat{\tau}_{12} \tag{5.3}$$

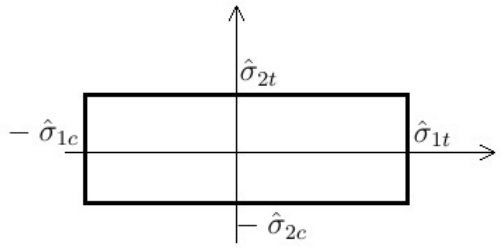


Figure 5.1: Stress limits w.r.t. maximum stress criterion

5.1.2 Maximum Strain Criterion

The occurred strains for a load, ε_1, ε_1 and γ_{12}, must be smaller than the maximums for corresponding strain type. The equation form is shown below.

$$-\hat{\varepsilon}_{1c} < \varepsilon_1 < \hat{\varepsilon}_{1t} \tag{5.4}$$

$$-\hat{\varepsilon}_{2c} < \varepsilon_2 < \hat{\varepsilon}_{2t} \tag{5.5}$$

$$|\gamma_{12}| < \hat{\gamma}_{12} \tag{5.6}$$

where $\hat{\varepsilon}_{1c}$ and $\hat{\varepsilon}_{1t}$ are the maximum compression and elongation respectively and $\hat{\gamma}_{12}$ is the maximum shear strain.

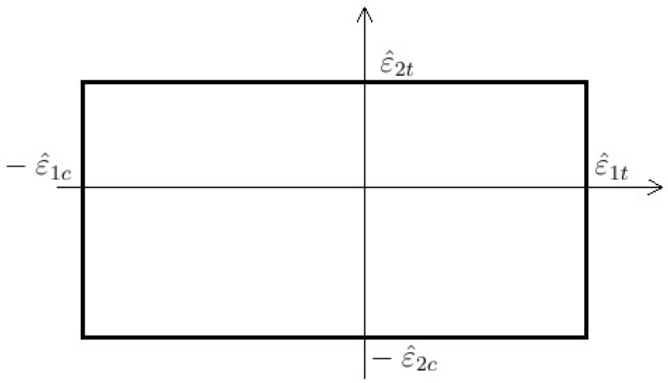


Figure 5.2: Strain boundaries w.r.t. maximum strain criterion

As can be seen from the Fig.5.1 and Fig.5.2, the coupled loadings in other words the combined stresses or strains are not being taken into account. In simple words, these two criteria state that, for a loaded material which is just about to touch one of these limits, still could handle the maximum stated strain or stress on the other direction which is not the case in reality.

In order to avoid this wrong assumption or find a better approximation for failures, methods called *Tsai-Hill* and *Tsai-Wu* are used.

5.1.3 Tsai-Hill Criterion

In this criterion, the main approximation is based on coupling all the strengths with occurred stresses. The simplified relation is shown below in the Eq.5.7 which is shorter than the original form due to neglected 3rd. dimension.

$$\frac{\sigma_1^2}{\hat{\sigma}_1^2} - \frac{\sigma_1\sigma_2}{\hat{\sigma}_1^2} + \frac{\sigma_2^2}{\hat{\sigma}_2^2} + \frac{\tau_{12}^2}{\hat{\tau}_{12}^2} = 1 \tag{5.7}$$

The approximate drawing of this criterion is shown in the Fig.5.3. In the following sections, usage of this equation is explained.

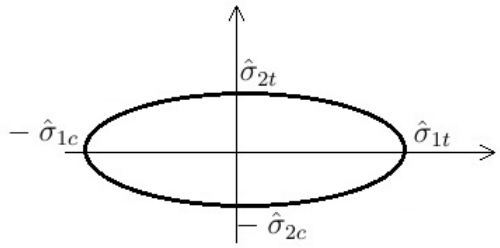


Figure 5.3: Stress limits w.r.t. Tsai-Hill criterion

5.1.4 Tsai-Wu Criterion

Same as Tsai-Hill, this model is also gives an elliptic boundaries for material strength. The simple plot of the Eq.5.8 is shown beliw in the Fig.5.4.

$$\frac{\sigma_1^2}{\hat{\sigma}_{1t}\hat{\sigma}_{1c}} + \frac{\sigma_2^2}{\hat{\sigma}_{2t}\hat{\sigma}_{2c}} + \frac{\tau_{12}^2}{\hat{\tau}_{12}^2} + 2F_{12}\sigma_1\sigma_2 + \frac{\sigma_1}{\hat{\sigma}_{1t}} - \frac{\sigma_1}{\hat{\sigma}_{1c}} + \frac{\sigma_2}{\hat{\sigma}_{2t}} - \frac{\sigma_2}{\hat{\sigma}_{2c}} = 1 \tag{5.8}$$

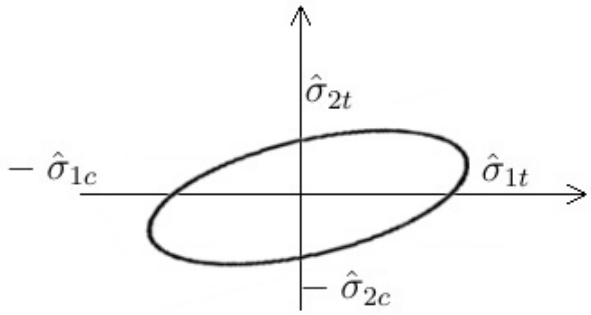


Figure 5.4: Stress limits w.r.t. Tsai-Wu criterion

where

$$F_{12} = F_{12}^{*}\sqrt{F_{11}F_{22}} \tag{5.9}$$

and

$$F_{11} = \frac{1}{\hat{\sigma}_{1t}\hat{\sigma}_{1c}} \tag{5.10}$$

$$F_{11} = \frac{1}{\hat{\sigma}_{2t}\hat{\sigma}_{2c}} \tag{5.11}$$

The F_{12}^{*} value is arbitrarily considered either -0.5 or 0. In the following parts, the usage of this equation is also shown.

5.2 Failure Index Calculations

The failure index simply means that how close is the material to the failure with respect to the criterion being chosen. In other words, when this value exceeds 1, the material fails. In this section different criteria are being used to get the failure index values for the same laminate and the load. The structure and the load case are taken from the previous chapter.

It is good to remind that the numbering of the layers are changed up-side down for all calculations as also stated at the beginning of Chapter4. Basically, the layer number one represents the very top lamina.

In the following sections, the failures are being judged in the local level. So the indice number one is corresponds to the local first axis. In other words, σ_2 for second layer is on global coordinate system's x-axis due to the 90° orientation.

5.2.1 Max. Stress

The local stress components are judged with respect to maximum allowed stress values for corresponding direction. At the end, the index calculations were made the failure index values are shown in the Table5.2 below.

Layer No	$\hat{\sigma}_{1t}$	$\hat{\sigma}_{1c}$	$\hat{\sigma}_{2t}$	$\hat{\sigma}_{2c}$	$\hat{\tau}_{12}$
1	0.22			0.02	0.5
2		0.09	**2.95**		0.5
3		0.11	**4.21**		0.5
4	0.72		0.18		0.5

Table 5.2: Failure index values for each layer and each component w.r.t. *max. stress*

Since some materials had pressure and others had tensile stresses, it was reasonable to judge them only with corresponding limitation. As a result of this some gaps in

the Table5.2 are occurred. Moreover, it is obvious that the failure has occurred on the middle planes which were oriented at 90° and were under tensile stress in the transverse direction.

5.2.2 Max. Strain

For the maximum strain criterion, the same judgments are done and results are presented in the Table5.3.

Layer No	$\hat{\varepsilon}_{1t}$	$\hat{\varepsilon}_{1c}$	$\hat{\varepsilon}_{2t}$	$\hat{\varepsilon}_{2c}$	$\hat{\gamma}_{12}$
1	0.22			0.16	0.5
2		0.12	**3.17**		0.5
3		0.16	**4.49**		0.5
4	0.72			0.37	0.5

Table 5.3: Failure index values for each layer and each component w.r.t. *max. strain*

The failure is occurred in the second and third layers same as max. stress.

5.2.3 Tsai-Hill

In this section, Tsai-Hill criterion is implemented and all of the laminae are evaluated with respect to this criterion. The failure index values are as shown below.

Layer No	Tsai-Hill
1	0.25
2	**8.96**
3	**18.04**
4	0.28

Table 5.4: Failure index values for each layer w.r.t. *Tsai-Hill*

As expected failure happened on the middle layers. The index values are higher than other two criteria since Tsai-Hill is taking both directions into account, gives a closer result to the limits of the material.

The limits of Tsai-Hill is plotted in the following figure. As can be seen that it is considered as a material failure where the maximums are reached for two directions unlike than maximum strain or maximum stress criteria.

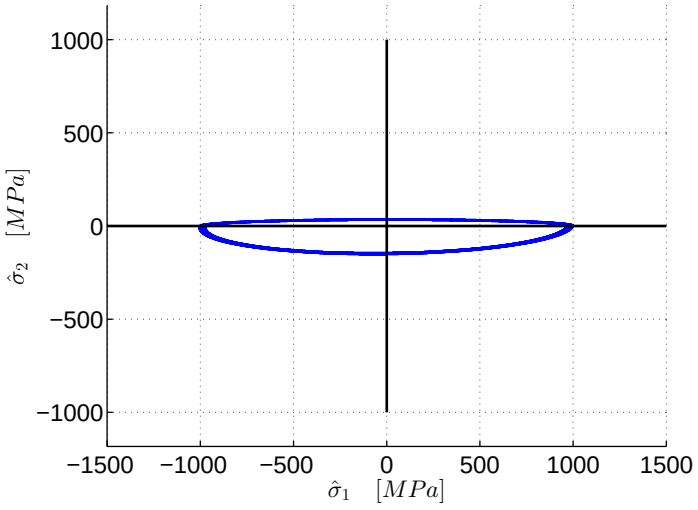


Figure 5.5: Tsai-Hill boundaries

5.2.4 Tsai-Wu

The results from Tsai-Wu criterion are presented below in the Table5.5.

Layer No	Tsai-Wu
1	0.21
2	**4.55**
3	**7.65**
4	0.91

Table 5.5: Failure index values for each layer w.r.t. *Tsai-Wu*

The criterion's limitations are plotted in Fig.5.6. The effect of F_{12}^{*} value is obvious. For this failure index calculation, the value 0 was chosen.

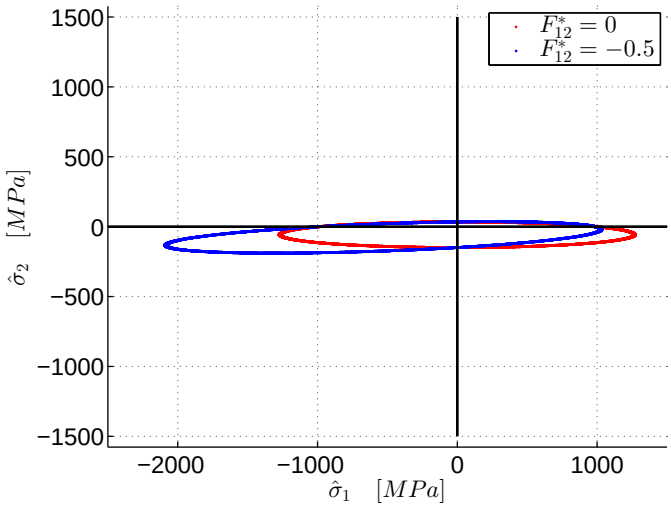


Figure 5.6: Tsai-Wu boundaries

It is important to mention that allowed material stresses can exceed the single direction max. stress of the material with Tsai-Wu which is obvious in the Fig5.6.

For the index calculations the matlab code is shown in AppendixA.4.1.

From these results, it is possible to conclude that ply failure has occurred for all 4 different criteria. Nevertheless, in order to determine if the structure is failed totally or not, a progressive failure system was needed. As described in the next section, a failure mechanism is adopted which states not to include failed lamina layers in the whole *ABD matrix* calculation.

5.3 Progressive Failure

In this section a progressive failure system is described. Basically, the main assumption was not to include failed laminae or take their stiffness as zero. It was chosen to make their fill their stiffness matrices with zeros instead of canceling them all when they fail in order to be able to keep the distance to the middle plane constant.

In other words, in each step the *ABD matrix* is being re-calculated while the load is being increased for the next step. For evaluation, Tsai-Hill criteria is used. The progressive failure can be seen from the Fig.5.7.

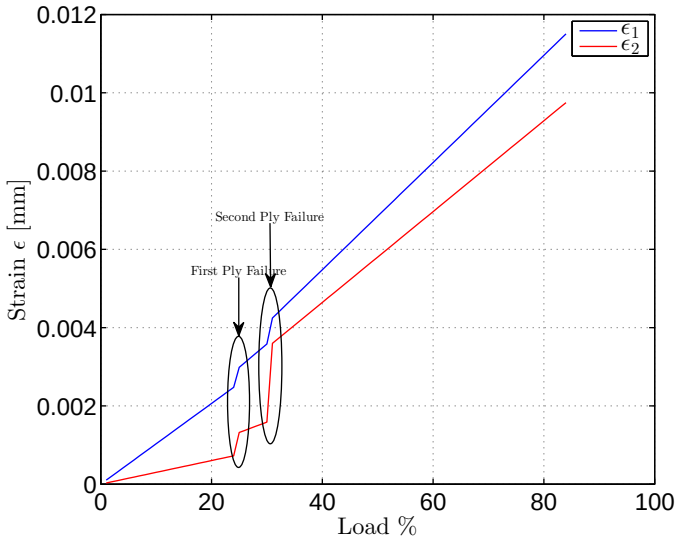


Figure 5.7: The strain behavior versus increased load

The first ply failure is occurred just below quarter load. As can be seen from the strain plot, at the time when the failure occurs, the stiffness matrix changes and due to the new, so to say 'less stiff' structure, the strain increases dramatically and the load is being carried by other layers which are still in hold.

At the end, the whole structure was failed at %84 of the total load. The very last two plies, which were the first and the last layers with 0° orientation, failed almost at the same time. The layer number 1, in other words the bottom layer, failed @%84 percentage of the load and the top layer was not able to carry the whole load by its self. Thus, the failure occurred on the structural level.

The Table5.6 below shows the failure order and the corresponding load. It is useful to remember that the order of the layers are switched due to easiness of the calculations with respect to the lecture book as described before.

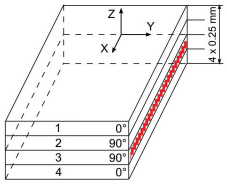
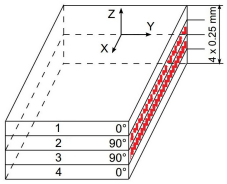
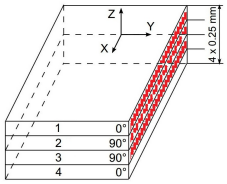
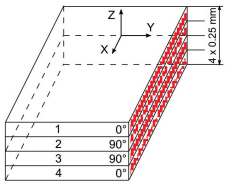
Load Level	Layer Number	
%24	3	
%30	2	
%84	1	
%84	4	

Table 5.6: Load levels and failure order

It can be concluded that the one step single failure calculation can result with wrong results. With respect to the single failure calculations, the structure was able to withstand the whole load. The middle layers were failing but the top and the bottom layers were still in the game. Nevertheless, it is proven that progressive failure calculation is needed to get realistic results if any failure exists. It would be a mistake to apply a single failure criteria with the whole load and judge it with a single iteration.

It is also good to mention that, if there is no any failure, the results from progressive and single step calculation would be the same since there would be no change in the ABD or layer-level stiffness matrices.

The Matlab code for progressive failure algorithm is provided in AppendixA.4.2.

Chapter 6

Composite Plates

In this chapter, calculations done for composite plates with respect to different layups and provided material information are presented. The dimensions of the rectangular panel are $a = 0.5mxb = 1.0m$ and the used layup orientations are $[0\ 0\ 0\ 0]_s$, $[90\ 90\ 90\ 90]_s$, $[0\ 90\ 0\ 90]_s$, $[90\ 0\ 90\ 0]_s$, and $[+45\ -45\ +45\ -45]_s$. The panel is made from carbon-epoxy material and the properties are :

$$E_1 = 46GPa \quad E_2 = 13GPa \quad G_{12} = 4.4GPa \quad \nu_{12} = 0.3$$

6.1 Maximum Deflections Under A Uniform Load

For the maximum deflection calculation, the equation 5.51a from the lecture book, shown below, is used. Derivation of this equation is based on assuming that a double Fourier sine series can represent the transverse deflection.

$$w = \sum_{n=1}^{\infty}\sum_{m=1}^{\infty} \frac{q_{mn} sin\frac{m\pi x}{a}\frac{n\pi x}{b}}{D_{11}[(\frac{m\pi}{a})^4 + \frac{2(D_{12}+2D_{66})}{D_{11}}(\frac{m\pi}{a})^2(\frac{n\pi}{b})^2 + \frac{D_{22}}{D_{11}}(\frac{n\pi}{b})^4]} \tag{6.1}$$

where w is the deflection in z direction, q_{mn} is the Fourier coefficient which becomes $\frac{16q}{mn\pi 2}$ while n and m values are both odd else $q_{mn} = 0$, D_{ij} is the element from the bending stiffness matrix of the structure and a and b are the dimensions of the rectangular shaped structure.

With respect to the Eq.6.1, the results for a uniform load $P_0 = 1kPa$ are shown below for different layups and Matlab codes are attached in AppendixA.5.1.

Layup:[0 0 0 0]s

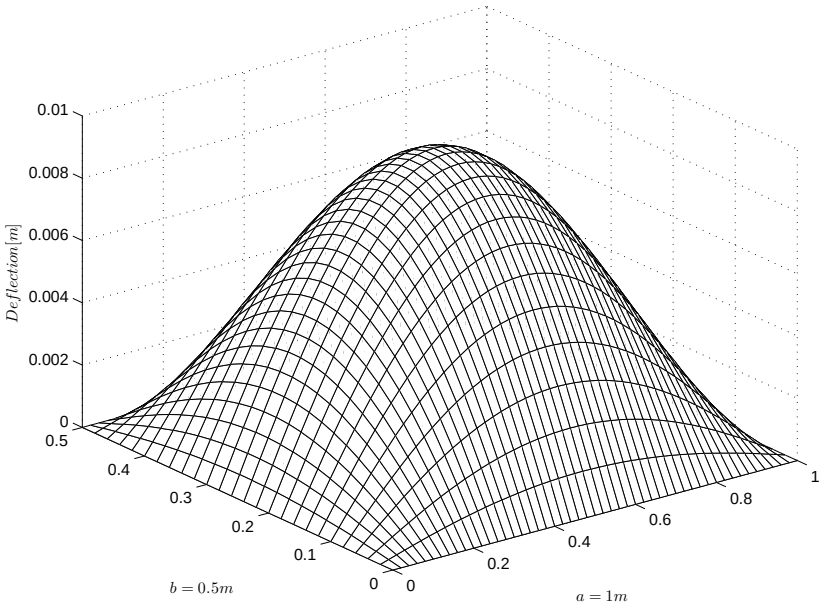


Figure 6.1: Deflection distribution. Layup:[0 0 0 0]s

Layup:[90 90 90 90]s

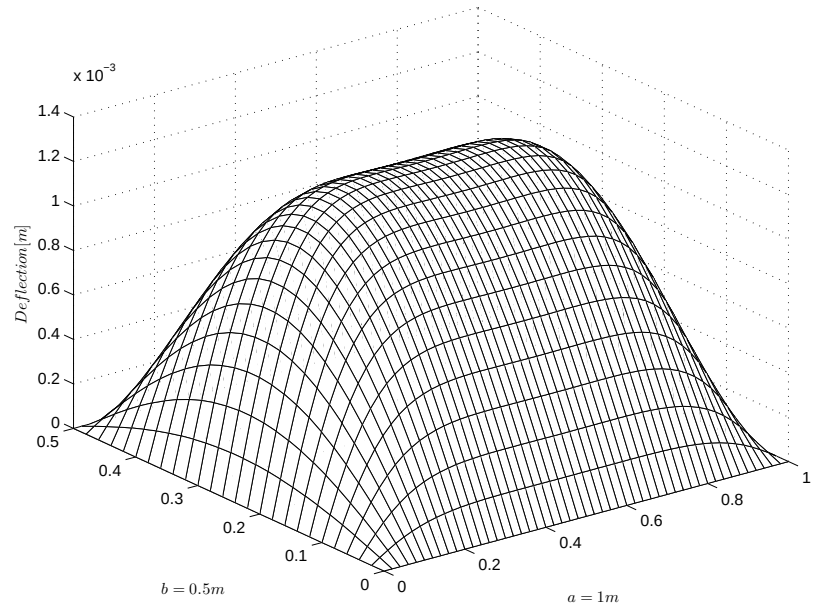


Figure 6.2: Deflection distribution. Layup:[90 90 90 90]s

Layup:[0 90 0 90]s

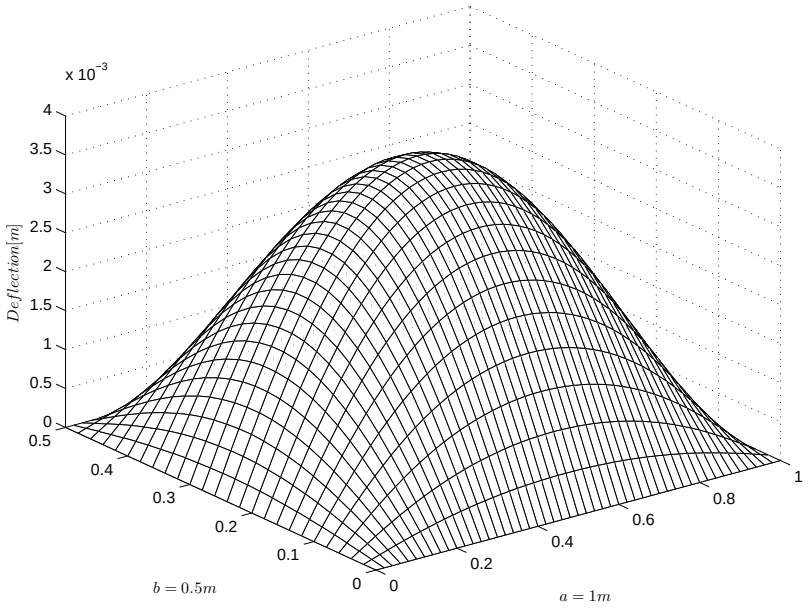


Figure 6.3: Deflection distribution. Layup:[0 90 0 90]s

Layup:[90 0 90 0]s

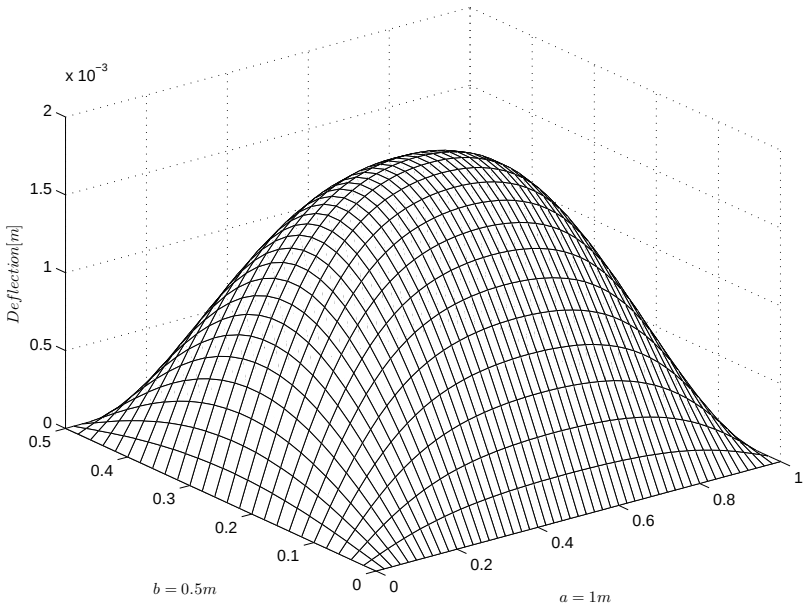


Figure 6.4: Deflection distribution. Layup:[90 0 90 0]s

Layup:[+45 -45 +45 -45]s

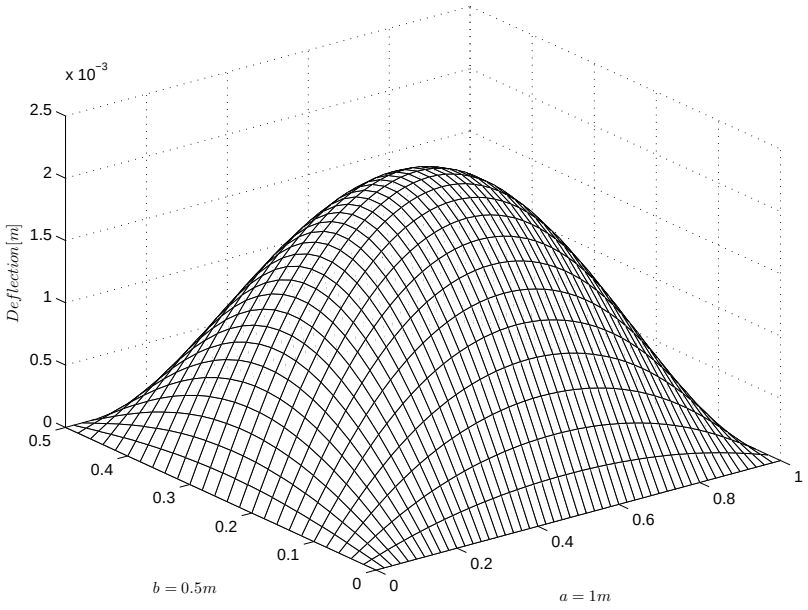


Figure 6.5: Deflection distribution. Layup:[+45 -45 +45 -45]s

The maximum deflections are shown in the Table6.1. From this part, it can be concluded that, without changing the used amount of material and its properties, the orientation difference can result with a different response.

Layout	Maximum Deflection$[mm]$
$[0\ 0\ 0\ 0]s$	0.85
$[90\ 90\ 90\ 90]s$	6.3
$[0\ 90\ 0\ 90]s$	1.3
$[90\ 0\ 90\ 0]s$	2.3
$[+45\ -45\ +45\ -45]s$	1.5

Table 6.1: Maximum deflections for different layups

6.2 Bending Moments Under A Uniform Load

In this section, the results of bending moment calculations are presented. For each layup, resulted distributions for M_x and M_y are shown below.

Layup:[0 0 0 0]s

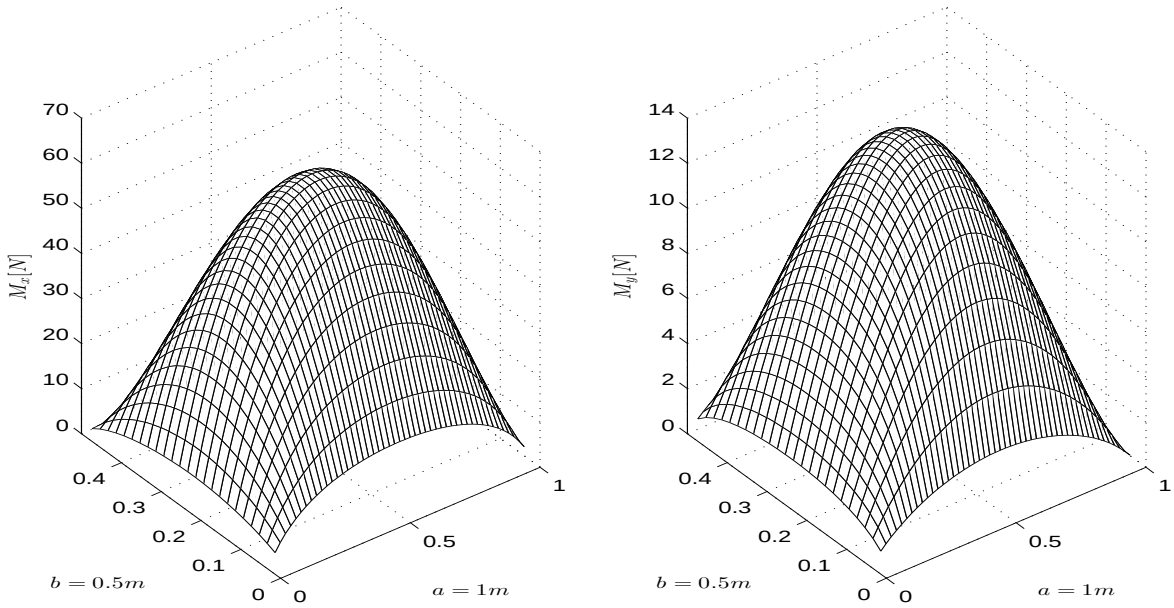


Figure 6.6: M_x (left) and M_y(right) distributions. Layup:[0 0 0 0]s.

Layup:[90 90 90 90]s

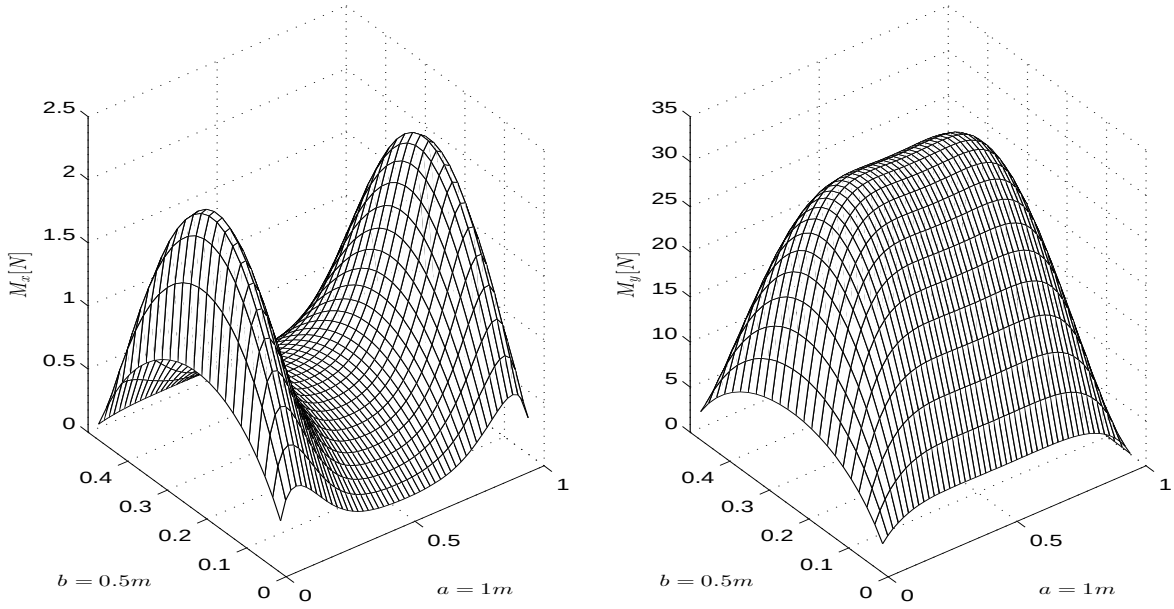


Figure 6.7: M_x (left) and M_y(right) distributions. Layup:[90 90 90 90]s

Layup:[0 90 0 90]s

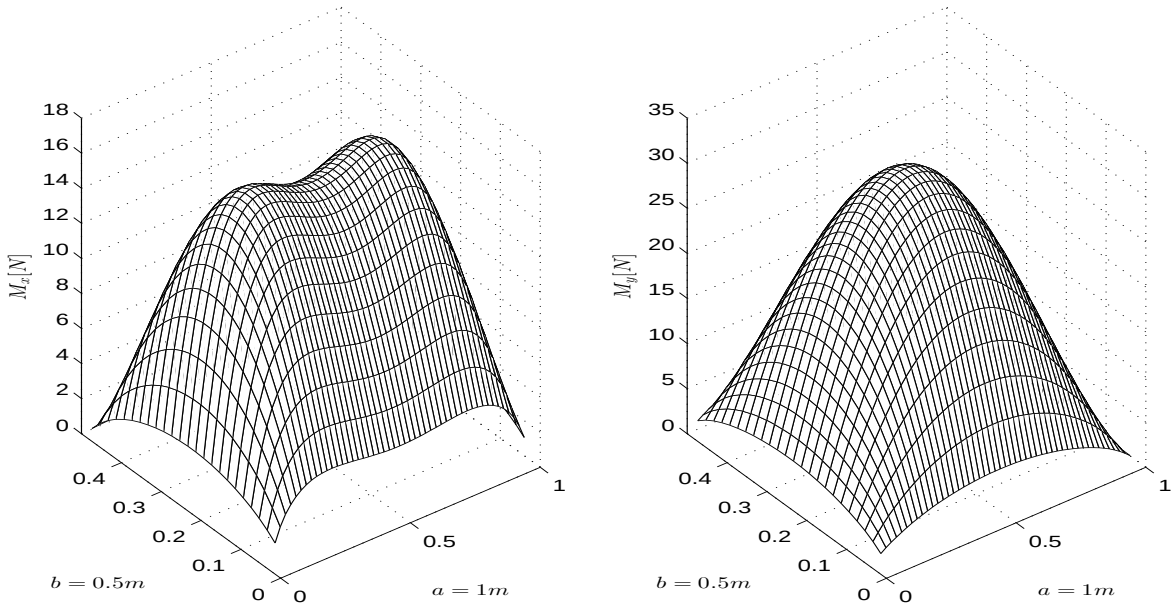


Figure 6.8: M_x (left) and M_y(right) distributions. Layup:[0 90 0 90]s

Layup:[90 0 90 0]s

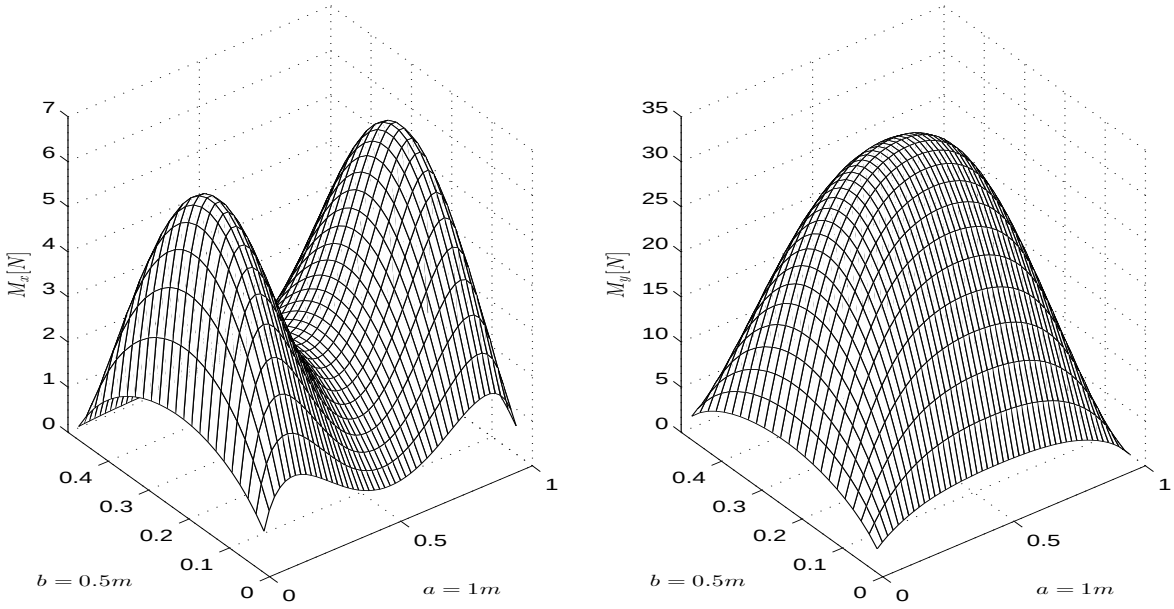


Figure 6.9: M_x (left) and M_y(right) distributions. Layup:[90 0 90 0]s

Layup:[+45 -45 +45 -45]s

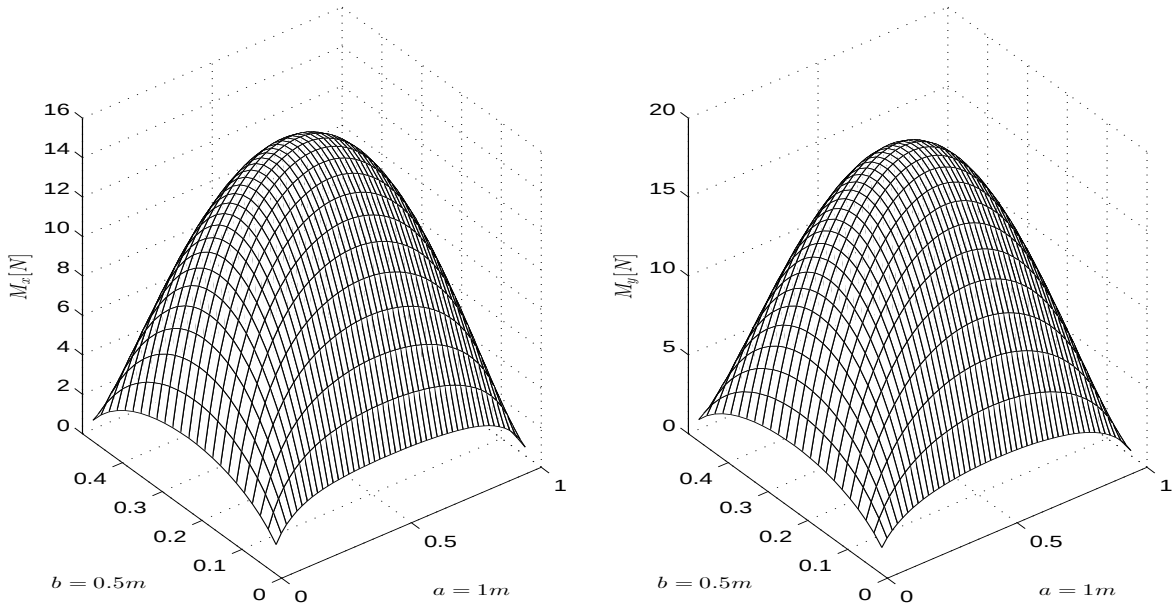


Figure 6.10: M_x (left) and M_y(right) distributions. Layup:[+45 -45 +45 -45]s

Although, one can intuitively expect to have maximum bending moments at the center of the plane, due to the orientation and stiffness matrices, it is not a necessity to have the maximum moments at the center. This can be seen from the figures presented above (i.e. Fig.6.9).

In the Table6.2 the maximum M_x and M_y bending moment values are shown for each layup. Again, the positive effect of the layer orientation is very clear.

Layout	Maximum M_x $[N]$	Maximum M_y $[N]$
$[0\ 0\ 0\ 0]s$	31.62	2.19
$[90\ 90\ 90\ 90]s$	13.7284	60.26
$[0\ 90\ 0\ 90]s$	52.20	10.46
$[90\ 0\ 90\ 0]s$	48.10	23.57
$[+45\ -45\ +45\ -45]s$	29.46	24.20

Table 6.2: Maximum bending moments for different layups

The presented codes in AppendixA.5.1 also include calculation for bending moments. The bending moment calculation is based on equation 5.11 from the lecture book where derivatives of the deflection distribution are taken numerically via Matlab.

6.3 Loads For The First Three Buckling Modes

The buckling calculation is based on Equ.6.2 shown below which gives the buckling load for corresponding mode.

$$P = \pi^2[D_{11}(\frac{m}{a})^2 + 2(D_{12} + 2D_{66})(\frac{1}{b})^2 + D_{22}(\frac{a}{m})^2(\frac{1}{b})^4] \tag{6.2}$$

where a and b are the dimensions of the rectangular shape and m and n are the number of half wakes in x and y axes respectively. In this particular case, $n = 1$ is used as a fixed value to have uni-axial buckling and for different modes, the m value is changed. In order to find the lowest modes, m values were changed through a for loop and at the end the first three buckling modes were selected and to keep it simple, the Matlab code adjusted to generate only necessary part. The results are shown in the Table6.3.

Layup	Mode number	m	n	Load $[N/mm]$
[0 0 0 0]s	1	1	1	25.54
	2	1	2	47.72
	3	1	3	94.36
[90 90 90 90]s	1	1	4	25.54
	2	1	5	26.99
	3	1	3	29.17
[0 90 0 90]s	1	1	2	47.71
	2	1	1	67.84
	3	1	3	73.98
[90 0 90 0]s	1	1	2	47.71
	2	1	3	49.54
	3	1	4	67.84
[+45 -45 +45 -45]s	1	1	2	80.92
	2	1	3	89.20
	3	1	4	107.76

Table 6.3: Buckling loads for corresponding mode and layup setting

The calculation algorithm is provided inside the given Matlab code in AppendixA.5.1.

6.4 Four Lowest Eigen Frequencies

In order to calculate the eigen frequencies the Equ.6.3 is used.

$$\rho^*\omega_{mn}^2 = D_{11}(\frac{m\pi}{a})^4 + 2(D_{12} + 2D_{66})(\frac{m\pi}{a})^2(\frac{n\pi}{b})^2 + D_{22}(\frac{n\pi}{b})^4 \tag{6.3}$$

where ρ^* is the surface density which can be calculated via Equ.6.4, ω is the corresponding eigen frequency, a and b are the dimensions of the rectangular plane and m and n are the parameters where we define the mode shape. In simple words, m is the half wave number in fiber direction for [0000] case, and n for the other.

$$\rho^* = \sum_{i=1}^{n} \rho_i h_i \tag{6.4}$$

which basically states that the $\rho*$ is sum of the densities multiplied with corresponding thickness value. To calculate the first lowest eigen modes, the equation was run in a for loop where the m and n values were varied. At the end the minimum first 4 modes were picked and converted to cycles per second $[Hz]$ from radians per second rad/s. The first four lowest eigen frequencies are shown in the Table6.4.

Layup	Mode number	m	n	Eigen Frequency $Hz[s^-1]$
[0 0 0 0]s	1	1	1	33.76
	2	1	2	89.14
	3	2	1	92.31
	4	2	2	135.07
[90 90 90 90]s	1	1	1	84.75
	2	2	1	92.31
	3	3	1	108.26
	4	4	1	135.07
[0 90 0 90]s	1	1	1	55.03
	2	2	1	92.31
	3	3	1	172.41
	4	1	2	200.14
[90 0 90 0]s	1	1	1	72.77
	2	2	1	92.31
	3	3	1	141.08
	4	4	1	220.14
[+45 -45 +45 -45]s	1	1	1	69.35
	2	2	1	120.20
	3	3	1	189.31
	4	1	2	210.72

Table 6.4: E,gen frequencies for corresponding mode and layup setting

6.5 Validation of Matlab Codes

The validation of Matlab codes are done with examples from the book. The book exercises numbered *5.1*, *5.6* and *5.7* are solved with the Matlab code shown in AppendixA.5.2 and the results were same apart from the fact that the book uses *[Newton-mm]* and *[MPa]*'s yet the code runs with *[Newton-m]* and *[Pa]* units.

Chapter 7

Finite Element Analysis

In this chapter, finite element analysis of the the same model, which was used in the previous chapter, is done. In the first part, the maximum deflections and a buckling load analysis for each layup will be presented. At the end the four lowest eigen frequencies will be shown.

The used finite element calculation based on a 2D shell model with a fine had 0.025m sized grid spacing. At the end of each part, comparisons will be made with the previous chapters. Details are showed below.

7.1 Maximum Deflections

With the same load level, 1 kPa, the deflections are shown.

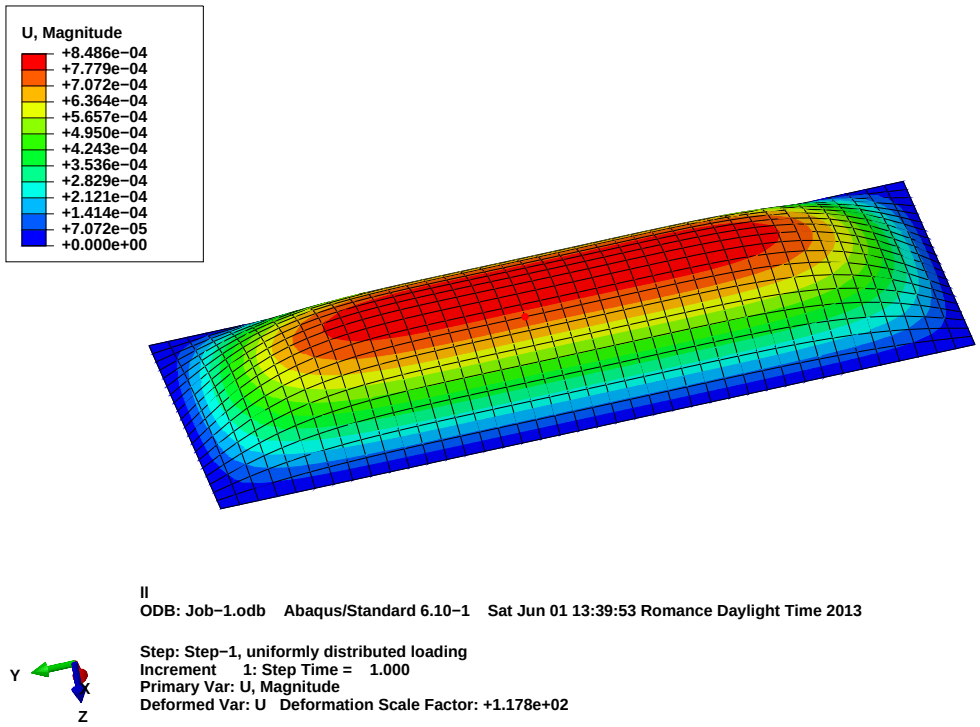


Figure 7.1: Maximum deflections Layup:[0 0 0 0]s

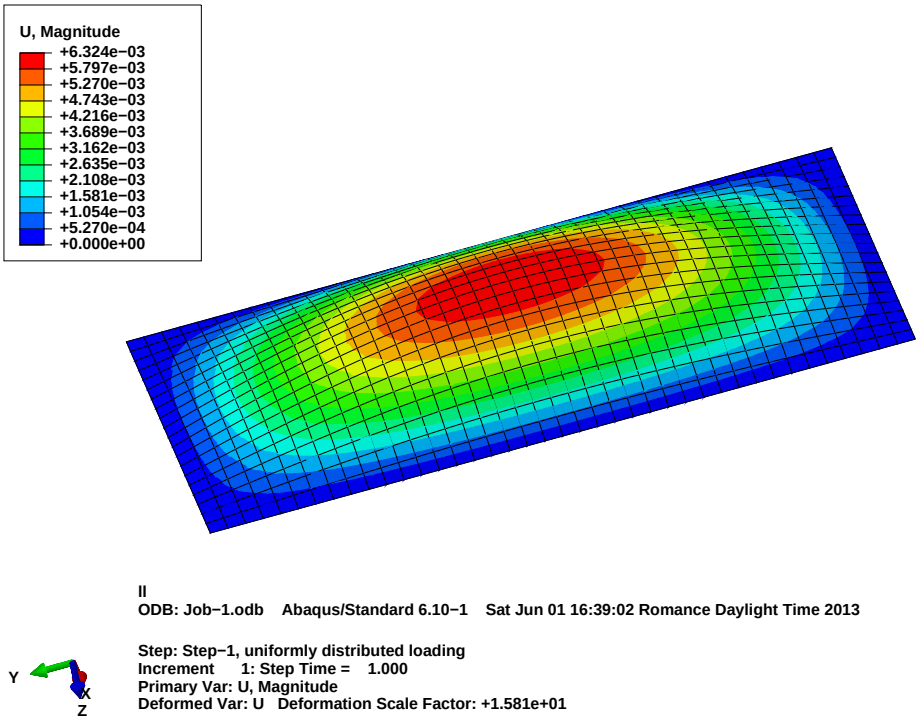


Figure 7.2: Maximum deflections Layup:[90 90 90 90]s

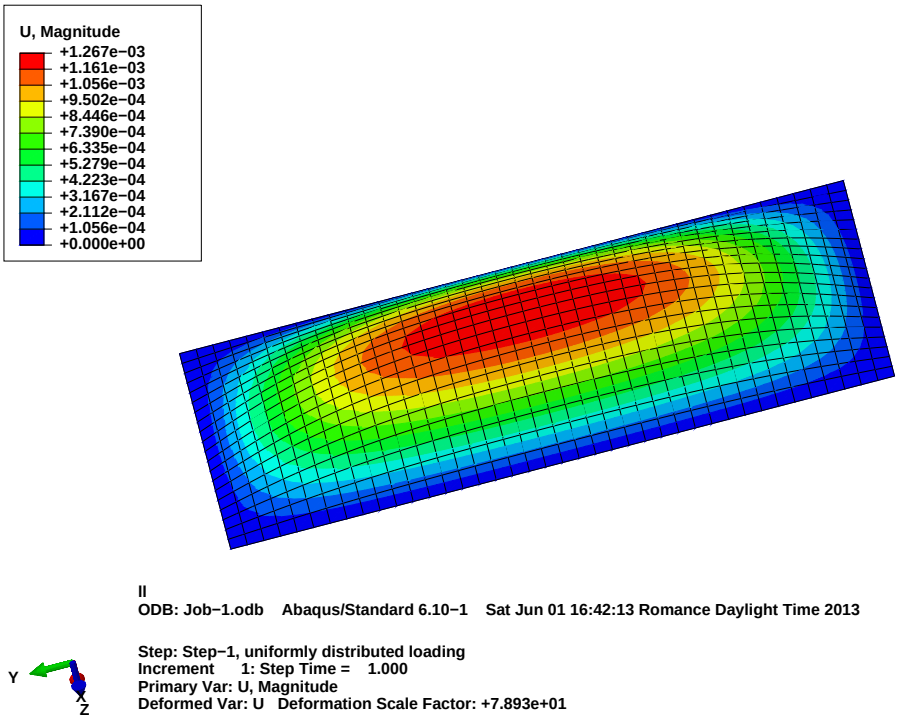


Figure 7.3: Maximum deflections Layup:[0 90 0 90]s

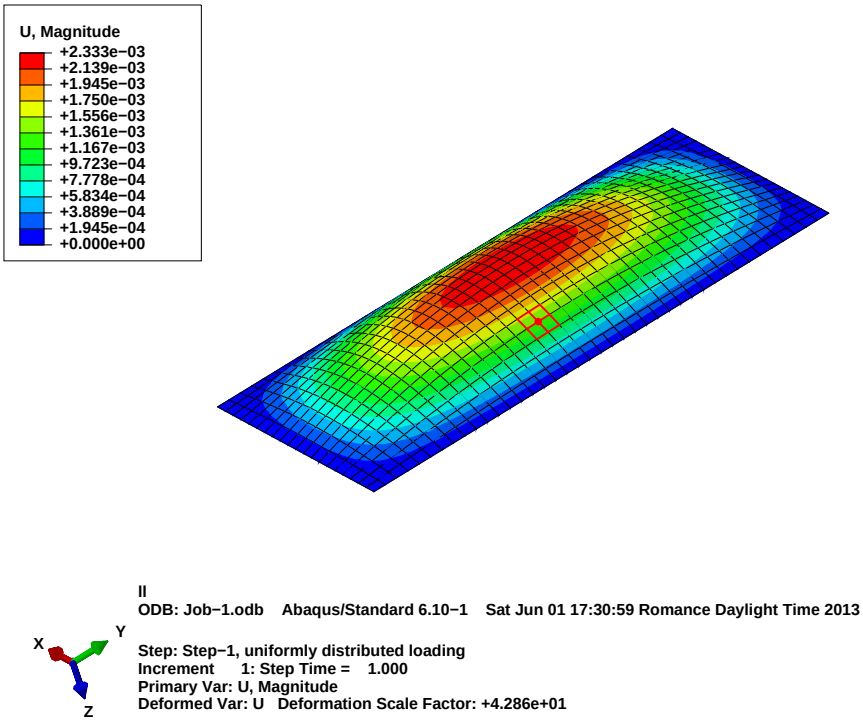


Figure 7.4: Maximum deflections Layup:[90 0 90 0]s

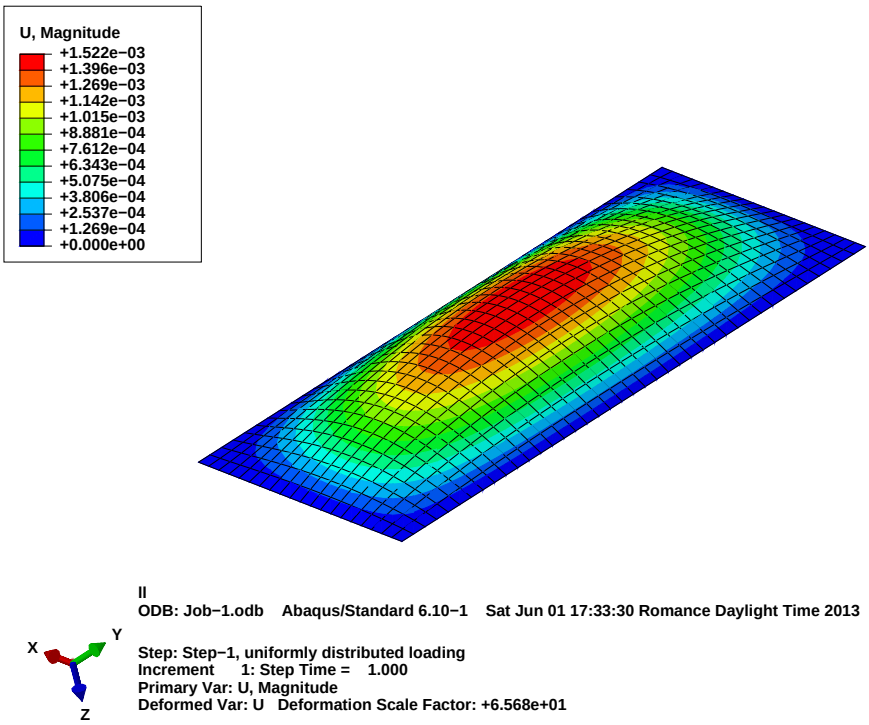


Figure 7.5: Maximum deflections Layup:[+45 -45 +45 -45]s

The results can be compared with the Table6.1 where the calculations were done via Matlab. According to the results, the Matlab calculations fit perfectly with the FE model.

7.2 Loads For The First Three Buckling Modes

In this section, the buckling behavior is investigated. For each layup, first three buckling modes are plotted and required loads are shown and compared with the previous chapter.

7.2.1 Layup:[0 0 0 0]s

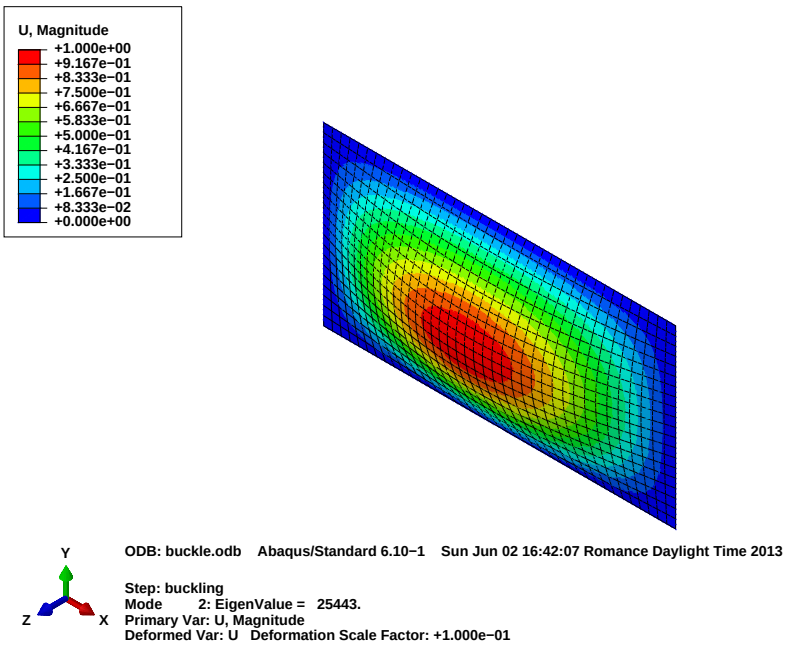


Figure 7.6: Buckling 1st mode, Layup:[0 0 0 0]s

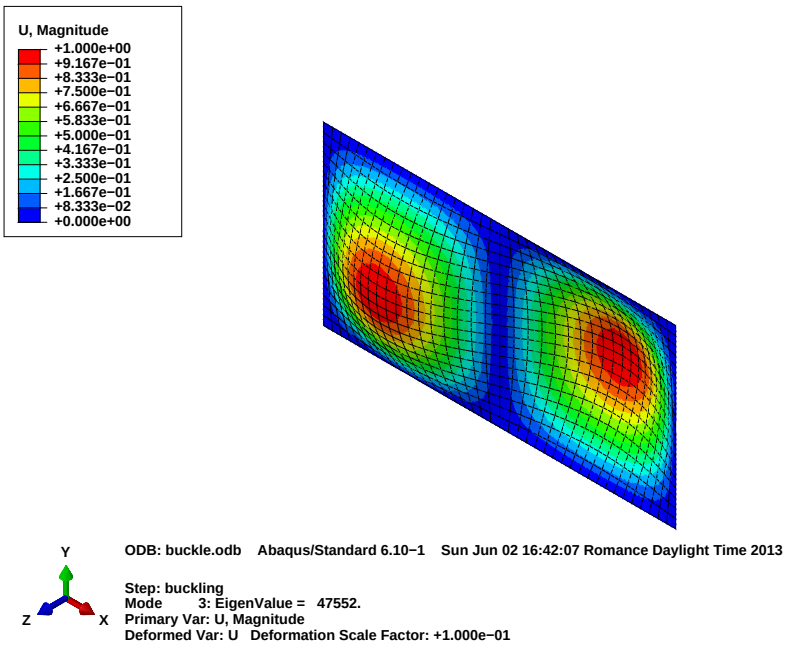


Figure 7.7: Buckling 2nd mode, Layup:[0 0 0 0]s

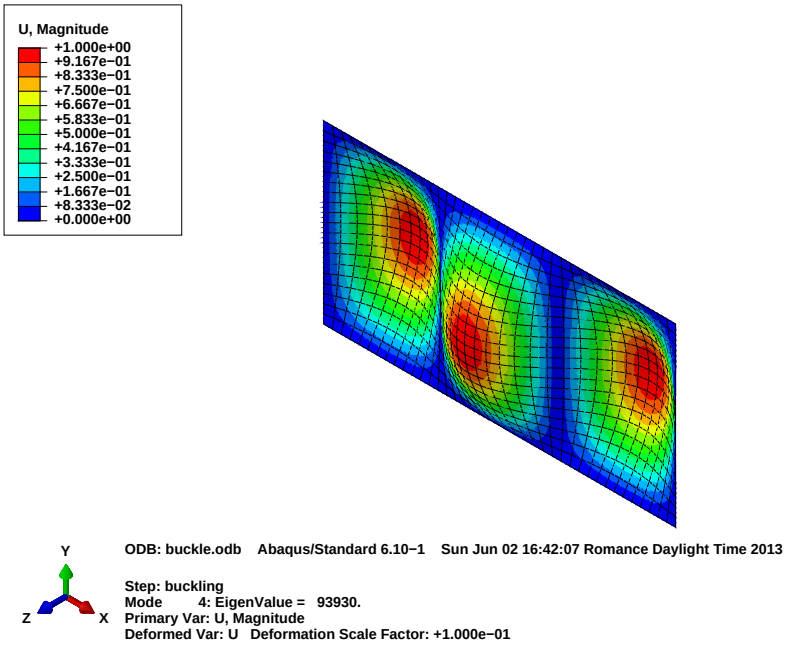


Figure 7.8: Buckling 3rd mode, Layup:[0 0 0 0]s

7.2.2 Layup:[90 90 90 90]s

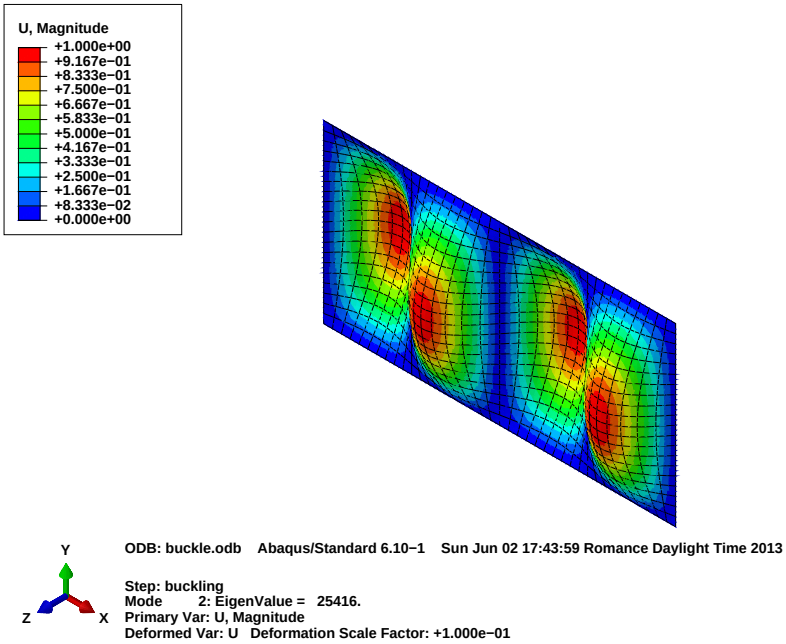


Figure 7.9: Buckling 1st mode, Layup:[90 90 90 90]s

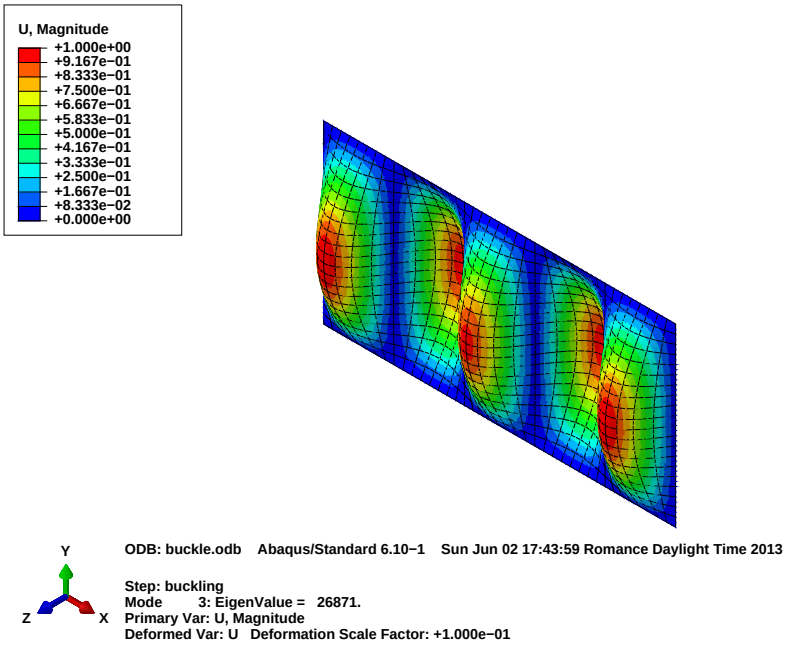


Figure 7.10: Buckling 2nd mode, Layup:[90 90 90 90]s

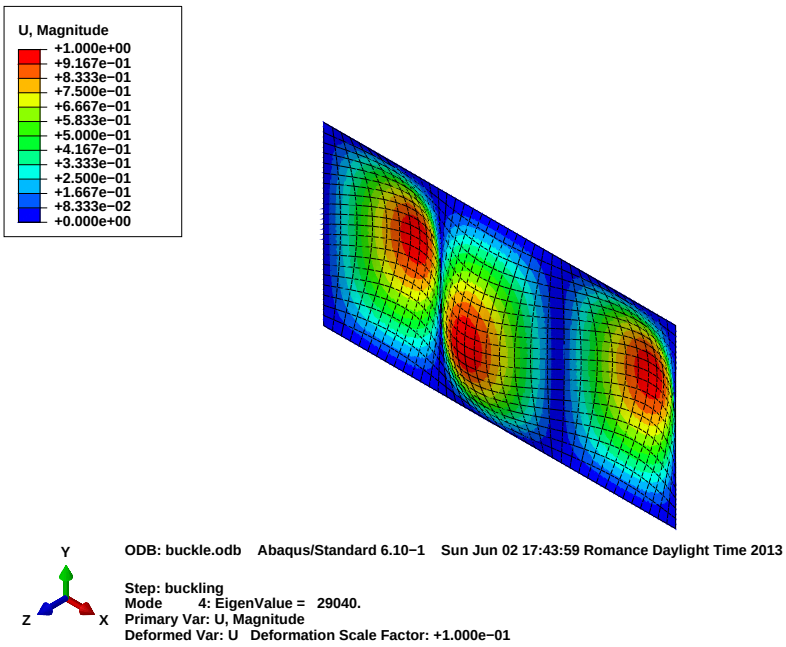


Figure 7.11: Buckling 3rd mode, Layup:[90 90 90 90]s

7.2.3 Layup:[0 90 0 90]s

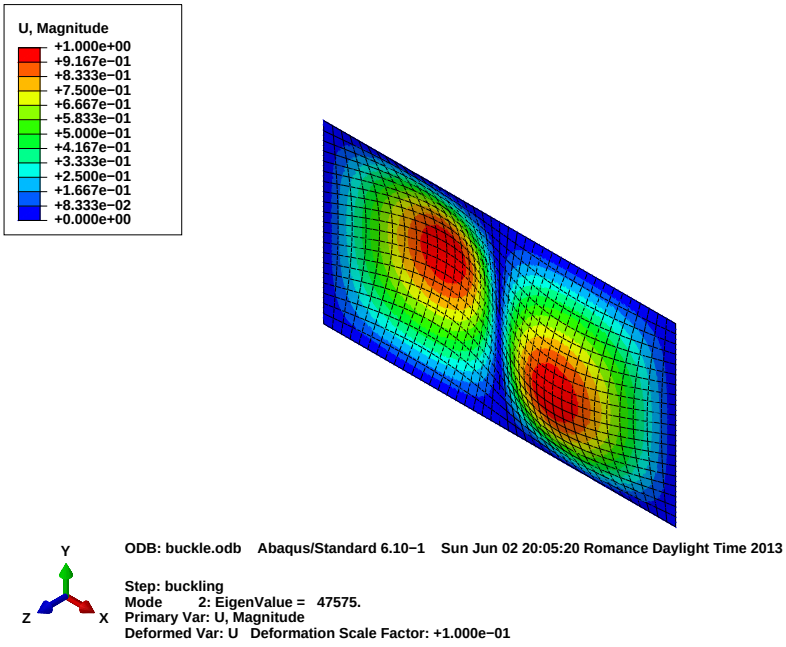


Figure 7.12: Buckling 1st mode, Layup:[0 90 0 90]s

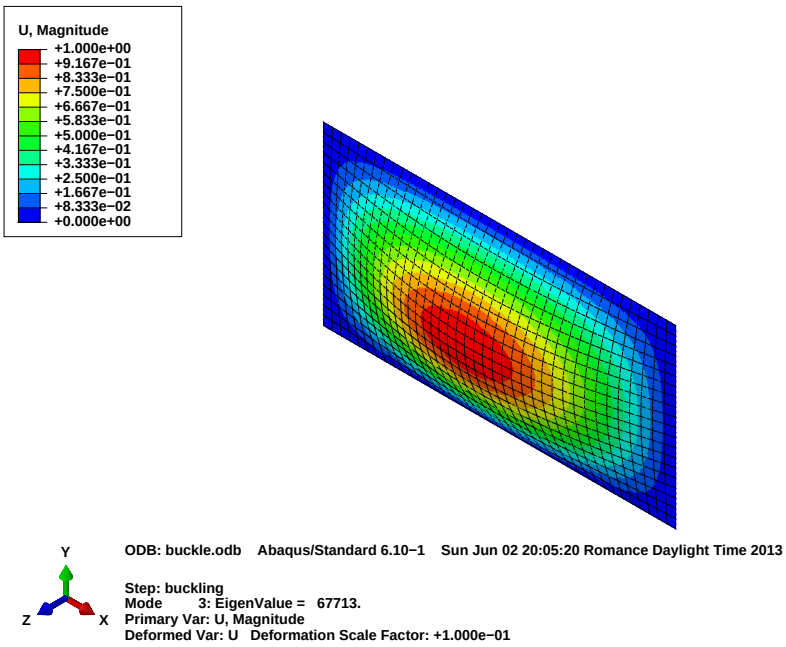


Figure 7.13: Buckling 2nd mode, Layup:[0 90 0 90]s

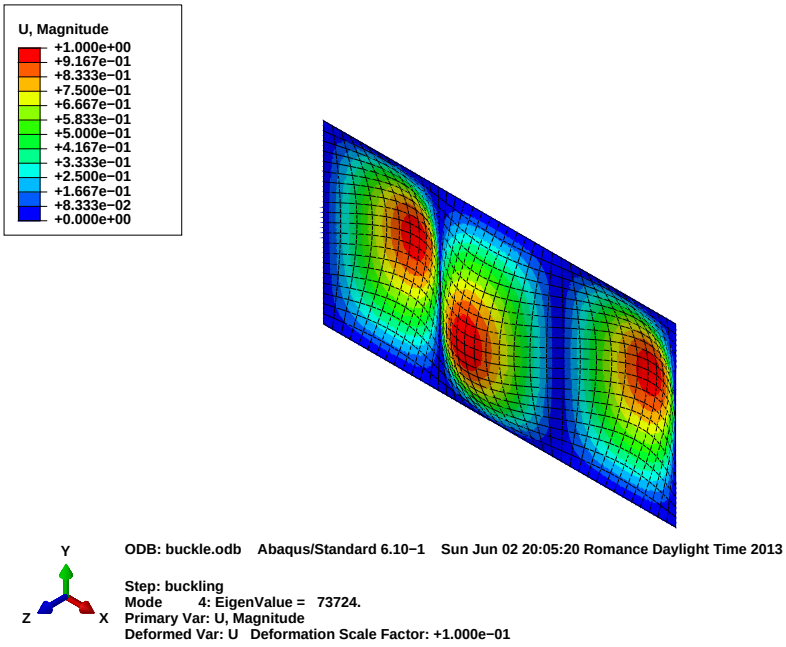


Figure 7.14: Buckling 3rd mode, Layup:[0 90 0 90]s

7.2.4 Layup:[90 0 90 0]s

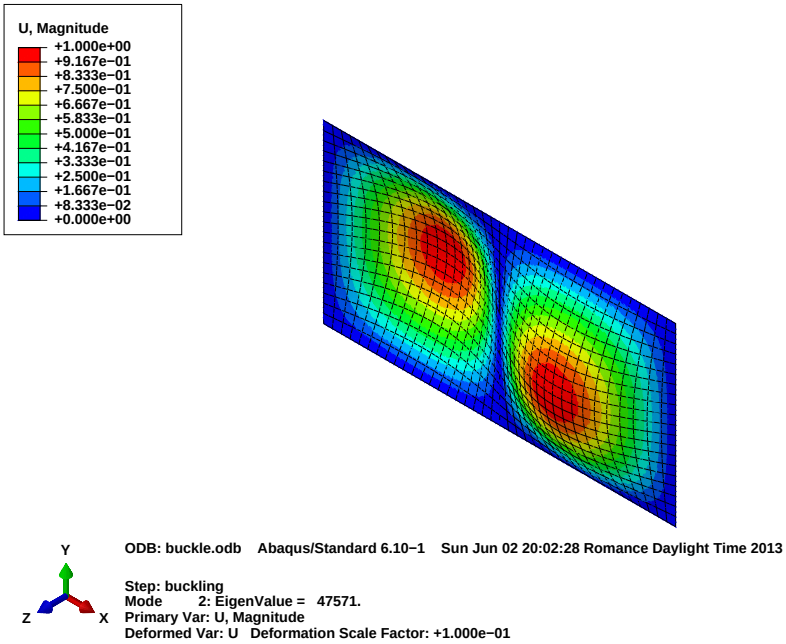


Figure 7.15: Buckling 1st mode, Layup:[90 0 90 0]s

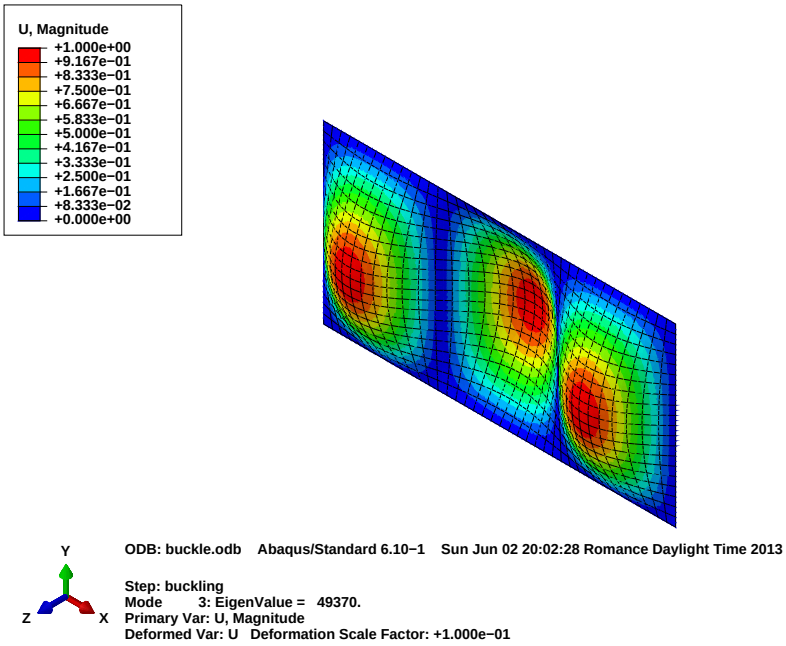


Figure 7.16: Buckling 2nd mode, Layup:[90 0 90 0]s

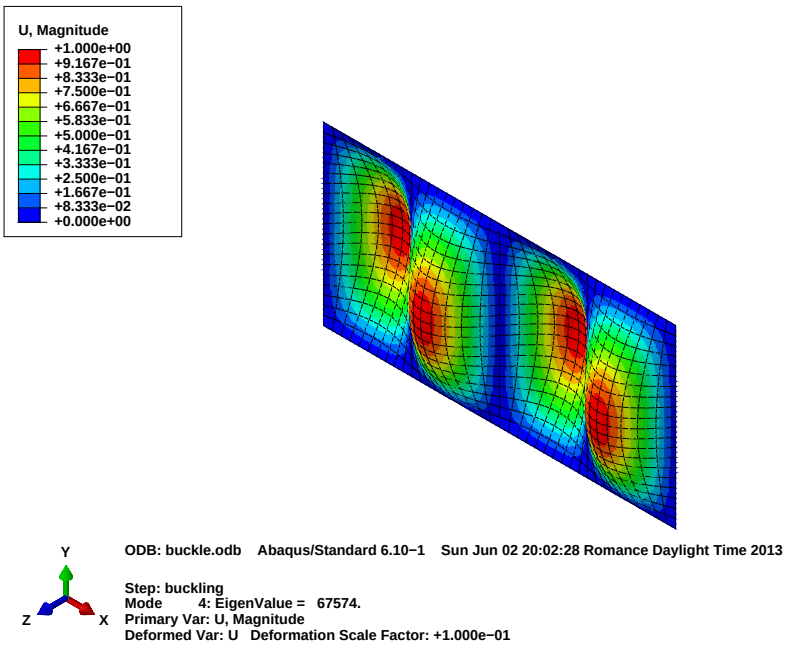


Figure 7.17: Buckling 3rd mode, Layup:[90 0 90 0]s

7.2.5 Layup:[+45 -45 +45 -45]s

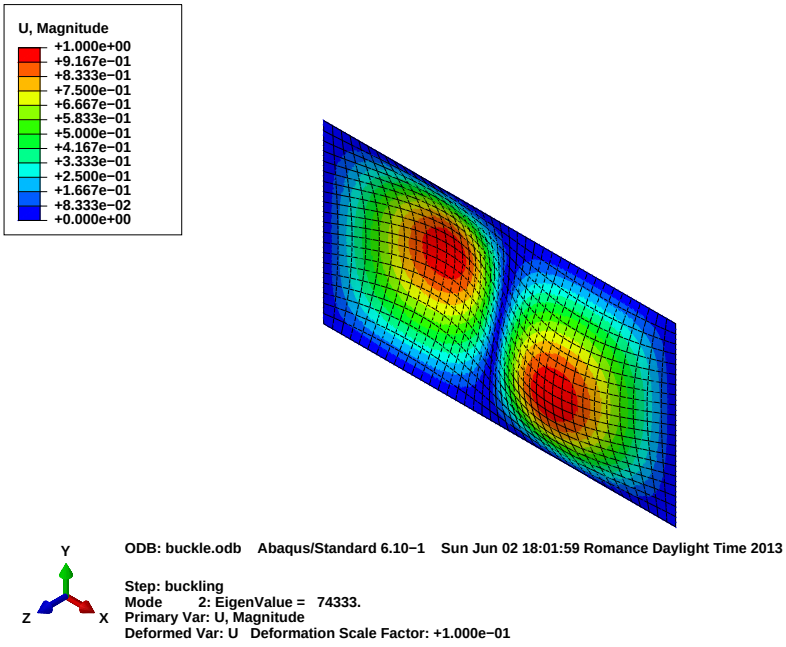


Figure 7.18: Buckling 1st mode, Layup:[+45 -45 +45 -45]s

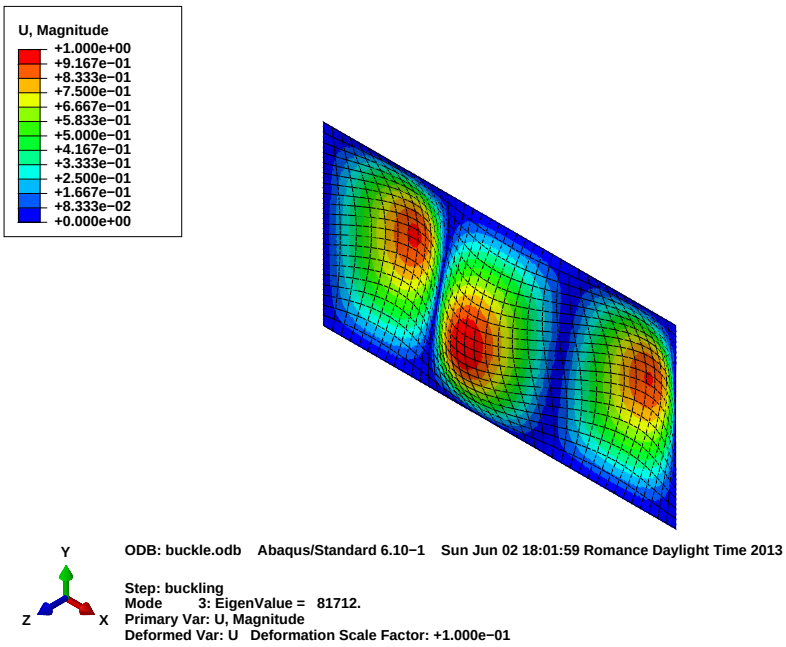


Figure 7.19: Buckling 2nd mode, Layup:[+45 -45 +45 -45]s

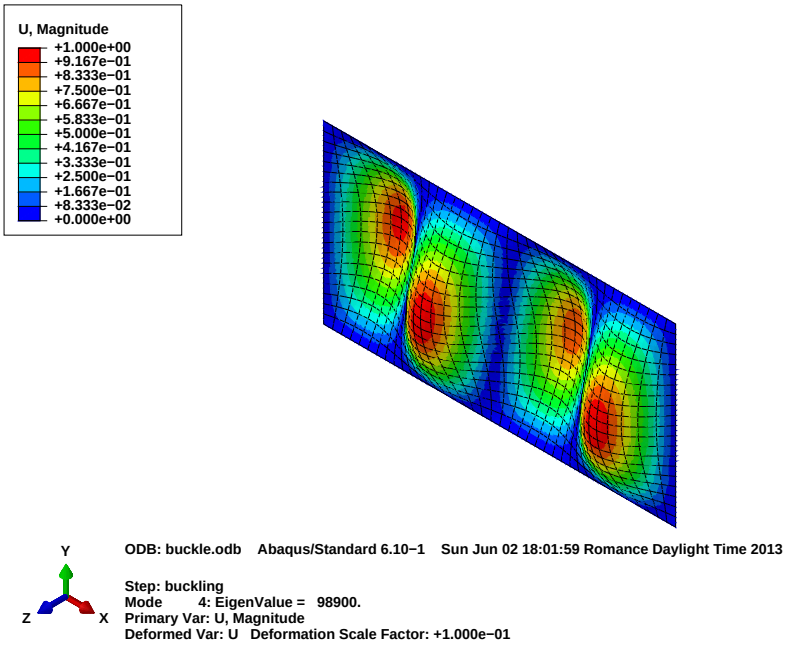


Figure 7.20: Buckling 3rd mode, Layup:[+45 -45 +45 -45]s

Eigen values can be read from the bottom-left of the images. The SI unit system was used and the unit for Eigen Values are $[N/m]$. In the images, the mode numbers

are shifted one up due to having a lower mode bacause of the boundary condition in ABAQUS setup, which was ignored.

It is concluded that the FEA results, mode shapes and corresponding loads, fit perfectly with most of the Matlab calculated results (see the Table6.3). Nevertheless, there is an obvious gap between FEA and Matlab results for the Layup setting [+45 -45 +45 -45]s. This difference can be explained by non-zero terms in D matrix, D_{16} and D_{26}, which were assumed to be zero at the derivation part of the Equ.6.2. Although the matrices from other layups also have non-zero values at these locations, the D_{26} value, especially, is relatively high due to the $+45°, -45°$ setting. More details about this can be found in the lecture book page 5.27.

7.3 Four Lowest Eigen Frequencies

7.3.1 Layup:[0 0 0 0]s

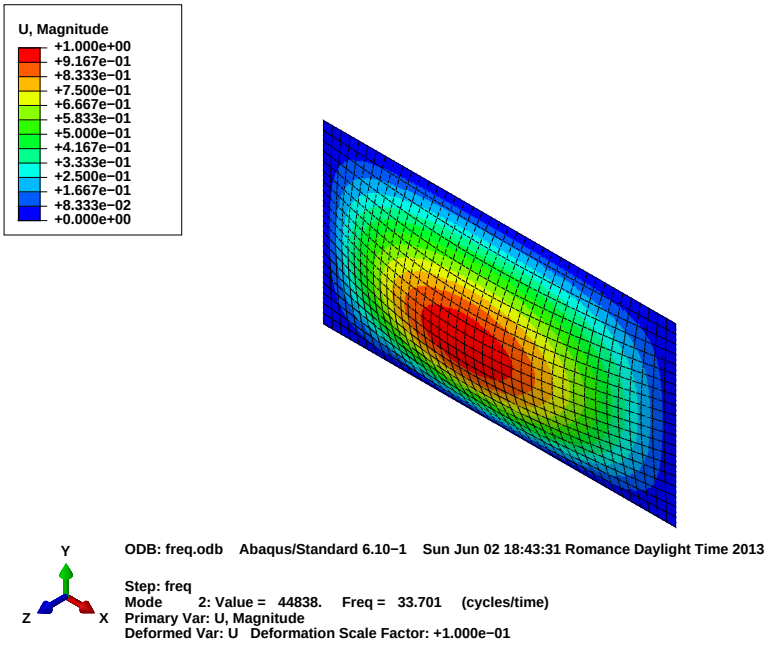


Figure 7.21: 1st eigen mode, Layup:[0 0 0 0]s

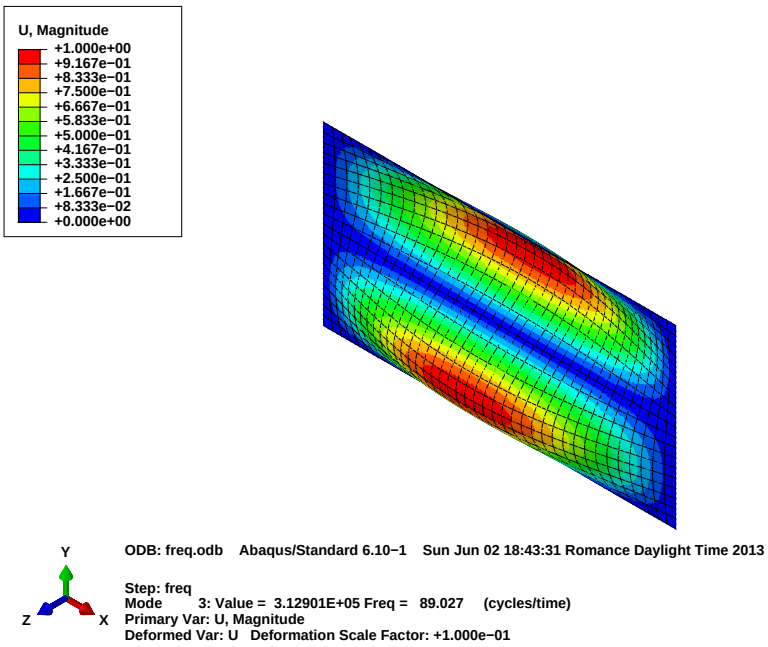


Figure 7.22: 2nd eigen mode, Layup:[0 0 0 0]s

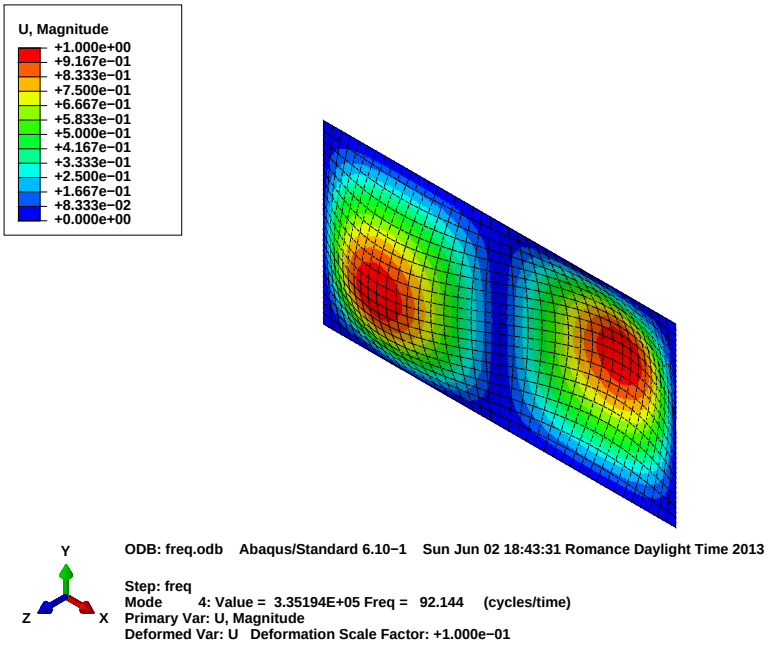


Figure 7.23: 3rd eigen mode, Layup:[0 0 0 0]s

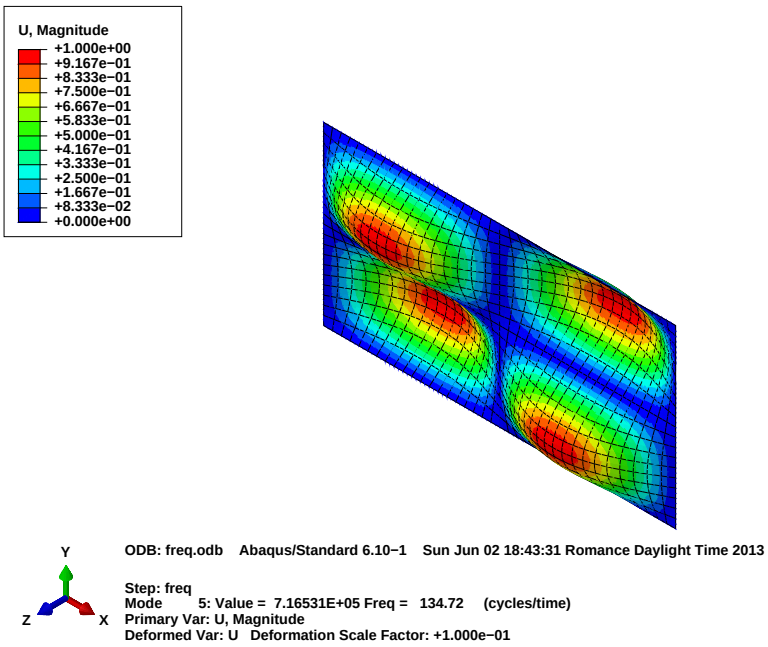


Figure 7.24: 4th eigen mode, Layup:[0 0 0 0]s

7.3.2 Layup:[90 90 90 90]s

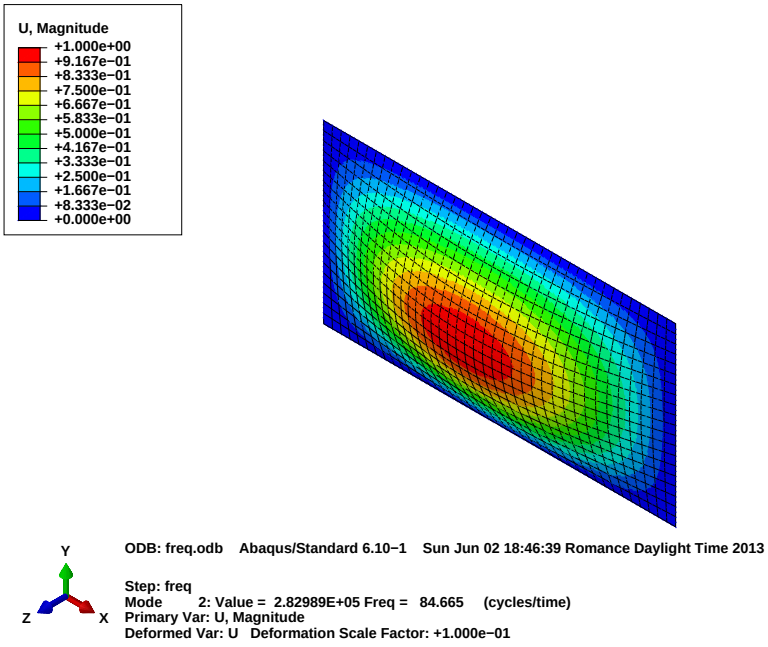


Figure 7.25: 1st eigen mode, Layup:[90 90 90 90]s

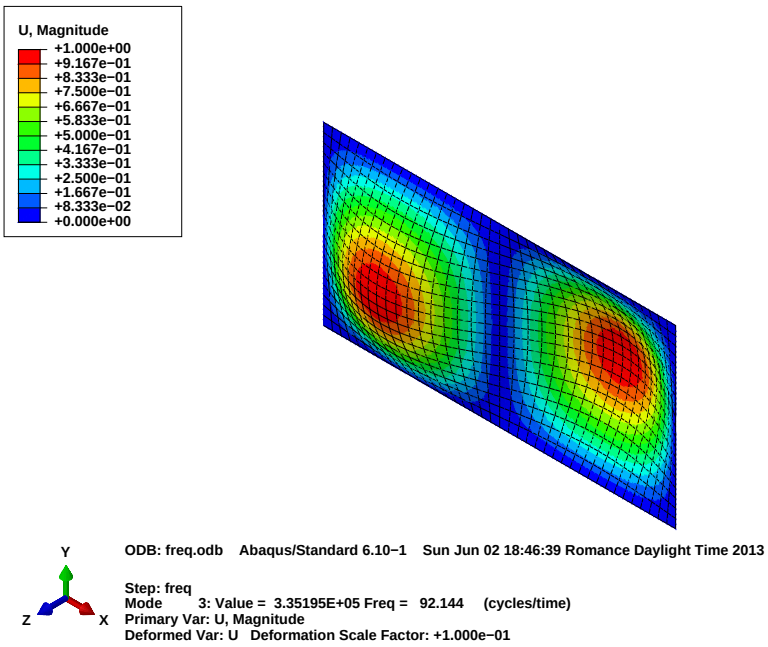


Figure 7.26: 2nd eigen mode, Layup:[90 90 90 90]s

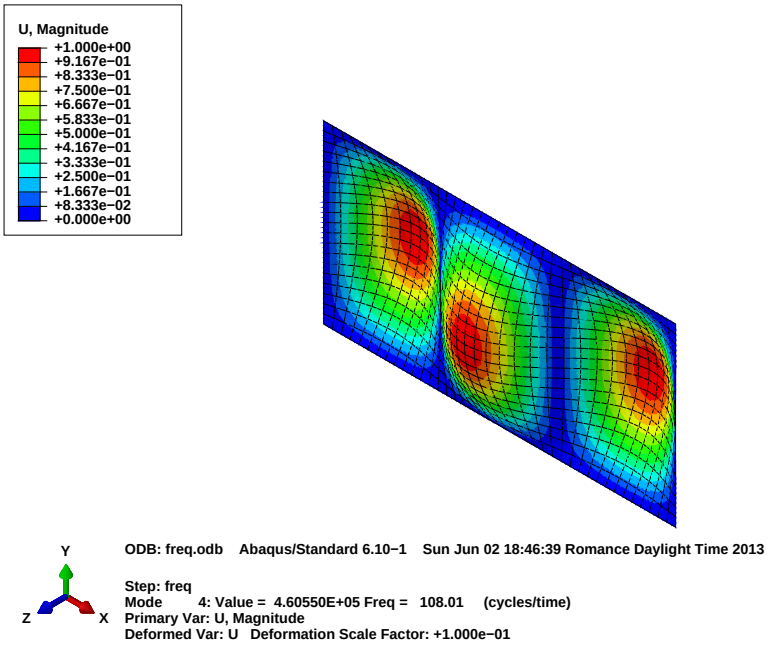


Figure 7.27: 3rd eigen mode, Layup:[90 90 90 90]s

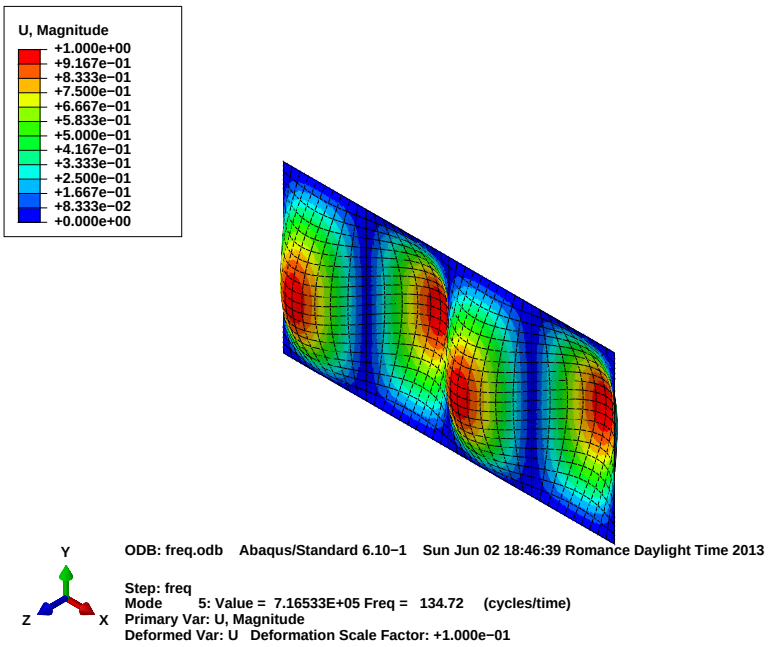


Figure 7.28: 4th eigen mode, Layup:[90 90 90 90]s

7.3.3 Layup:[0 90 0 90]s

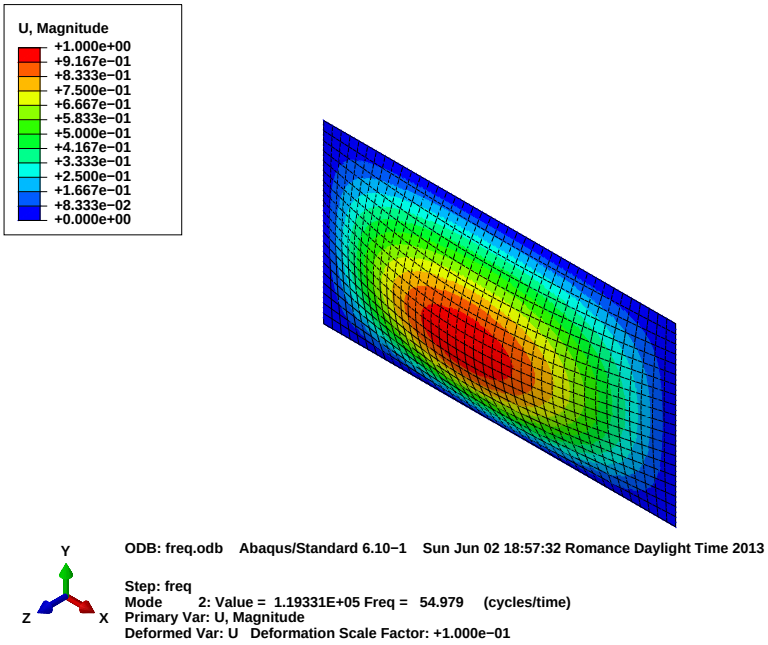


Figure 7.29: 1st eigen mode, Layup:[0 90 0 90]s

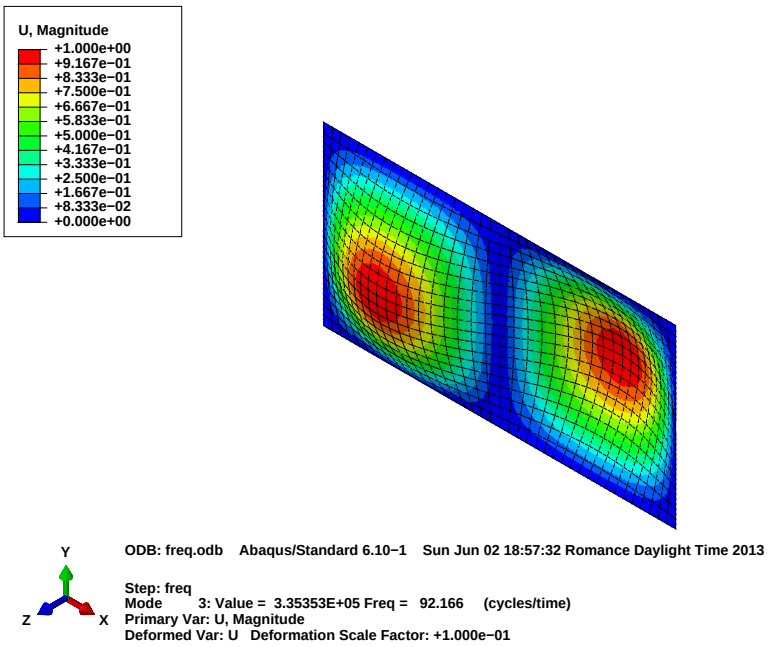


Figure 7.30: 2nd eigen mode, Layup:[0 90 0 90]s

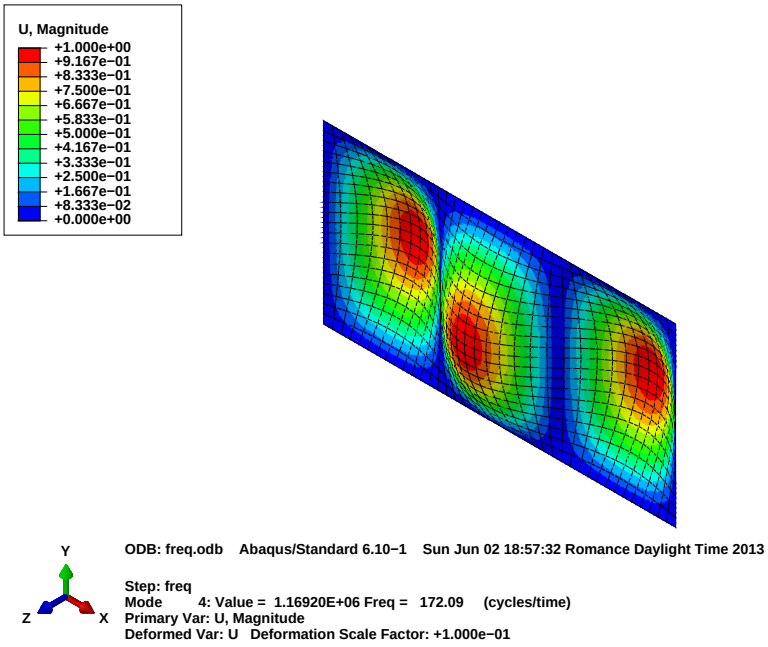


Figure 7.31: 3rd eigen mode, Layup:[0 90 0 90]s

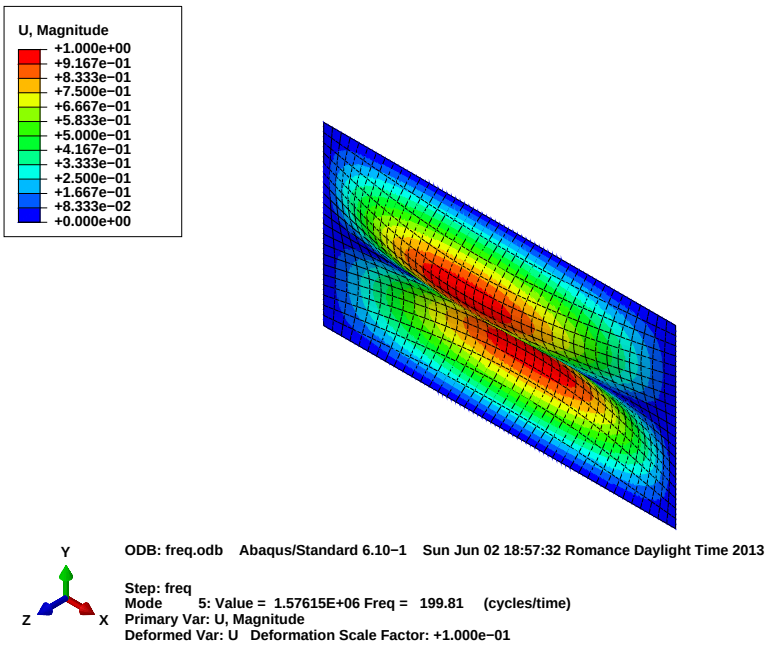


Figure 7.32: 4th eigen mode, Layup:[0 90 0 90]s

7.3.4 Layup:[90 0 90 0]s

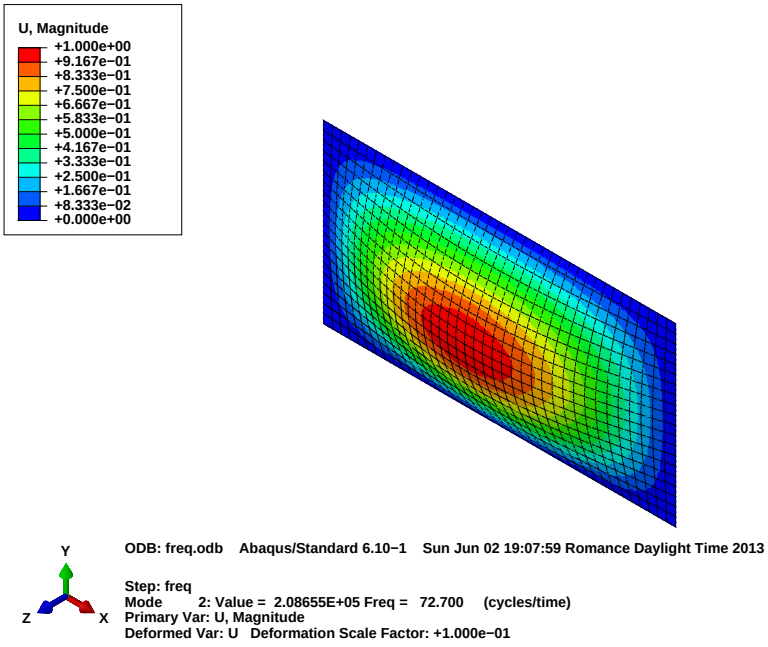


Figure 7.33: 1st eigen mode, Layup:[90 0 90 0]s

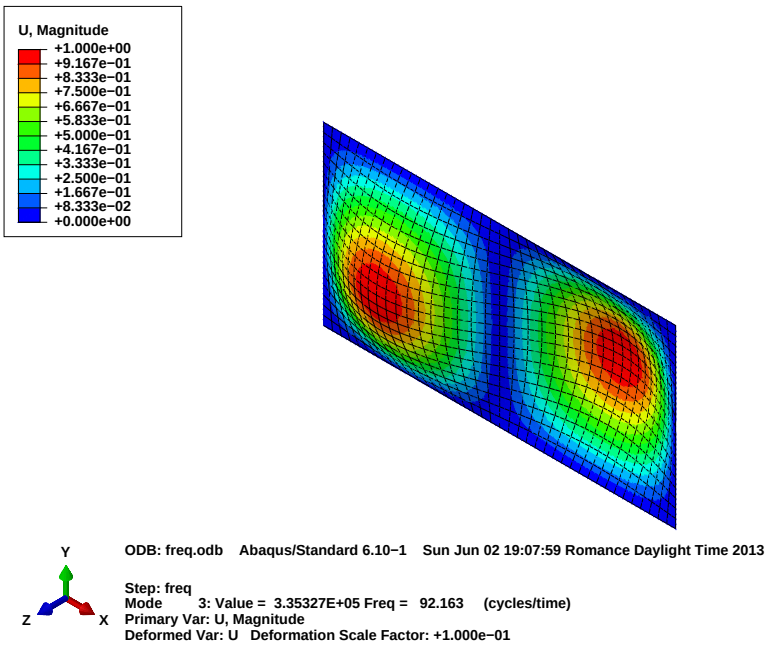


Figure 7.34: 2nd eigen mode, Layup:[90 0 90 0]s

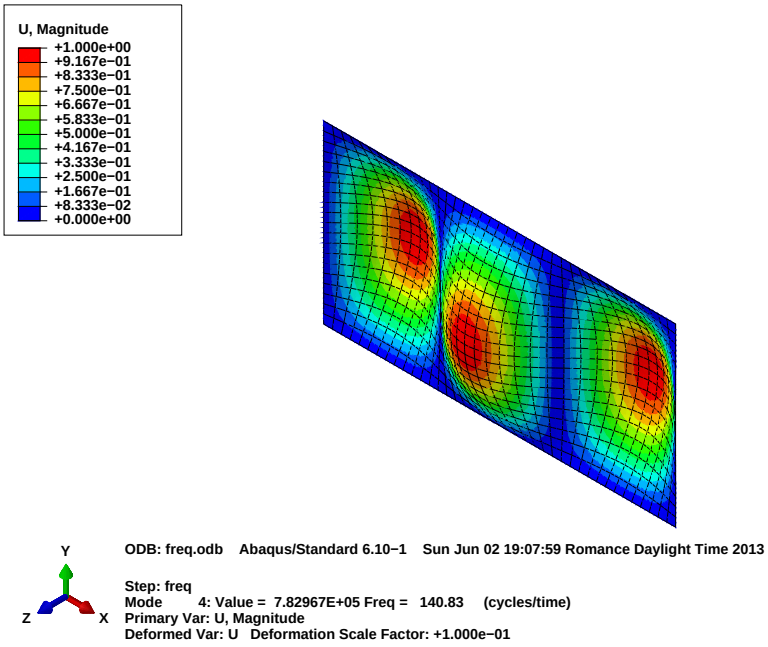


Figure 7.35: 3rd eigen mode, Layup:[90 0 90 0]s

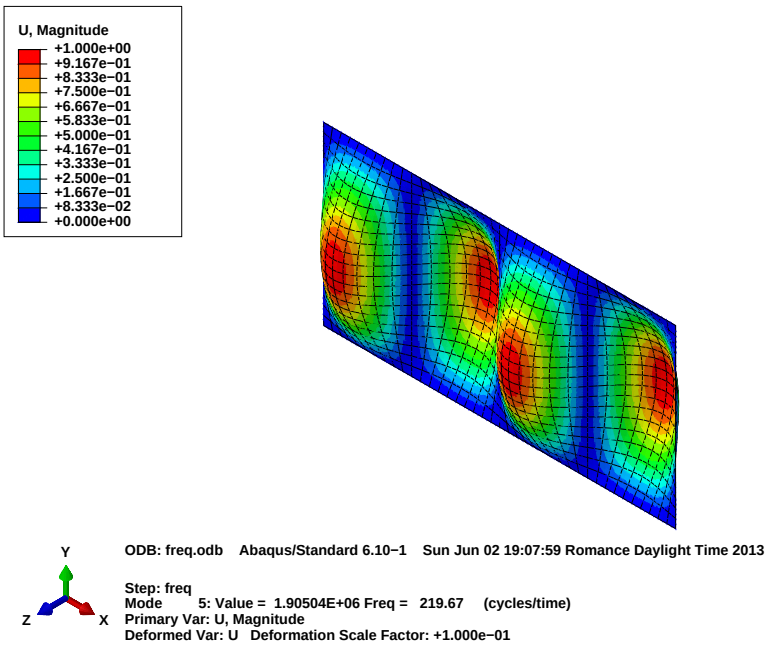


Figure 7.36: 4th eigen mode, Layup:[90 0 90 0]s

7.3.5 Layup:[+45 -45 +45 -45]s

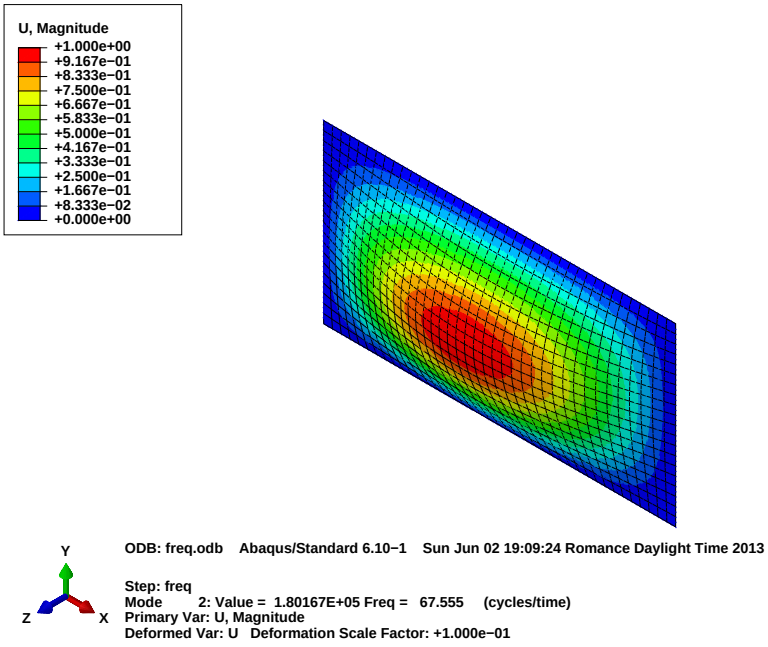


Figure 7.37: 1st eigen mode, Layup:[+45 -45 +45 -45]s

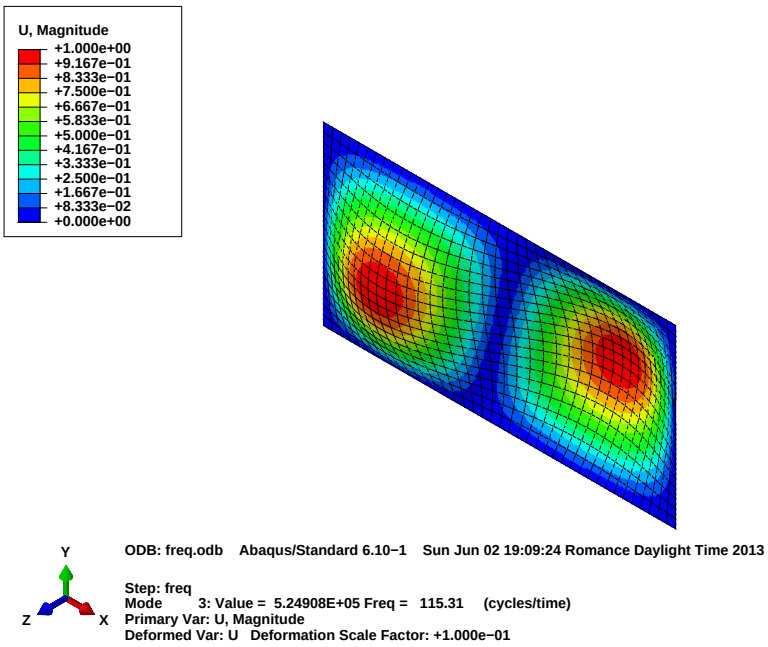


Figure 7.38: 2nd eigen mode, Layup:[+45 -45 +45 -45]s

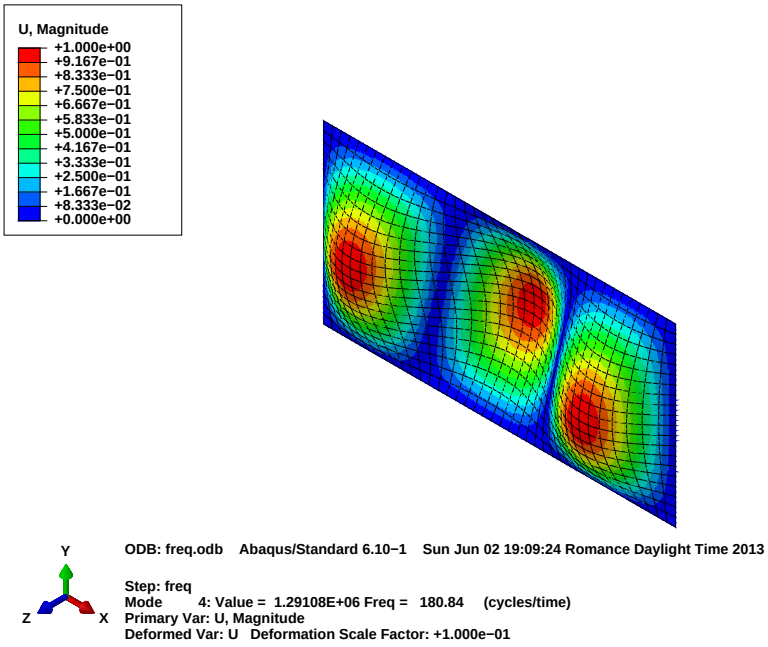


Figure 7.39: 3rd eigen mode, Layup:[+45 -45 +45 -45]s

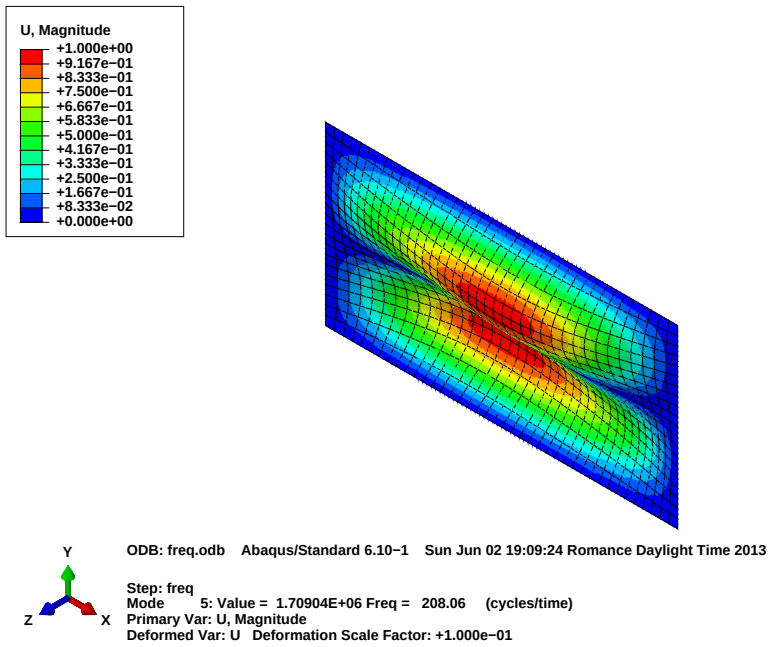


Figure 7.40: 4th eigen mode, Layup:[+45 -45 +45 -45]s

In conclusion, a good fit between Matlab calculations and FEA results is shown. The frequency values are shown as cycles per second at the bottom of each figure. It is good to remind that in some particular cases especially for higher modes the frequencies were not exactly the same. Yet, the highest difference was not more than 4%(see the Table6.4 for Matlab results).

Chapter 8

Laminate and Sandwich Structures

In the first part of this chapter, the advantages of having a sandwich structure will be shown. At first, two layers of epoxy-glass material with $t = 20mm$ thickness will be loaded at one end and other end will be fully clamped. Afterwards, a foam layer with a thickness of $40mm$ will be added between these two layers and with the same load, behavior of the structure will be investigated. At the end, the thickness of the foam layer will be increased to $120mm$ and the results will be compared with the other two cases. The comparison will be based on the deflection on y-axis U_2, σ_{11} and σ_{22} parameters.

In the second part, behavior of the last structure, which has $120mm$ foam layer between the faces, will be investigated under a specific case where roller beams are used to show the effect of shear beam deformation.

8.1 Effect of Foam Layer

As described above, the beam structures are loaded with the same load where they were fully clamped at the other end. The deformation results are shown below.

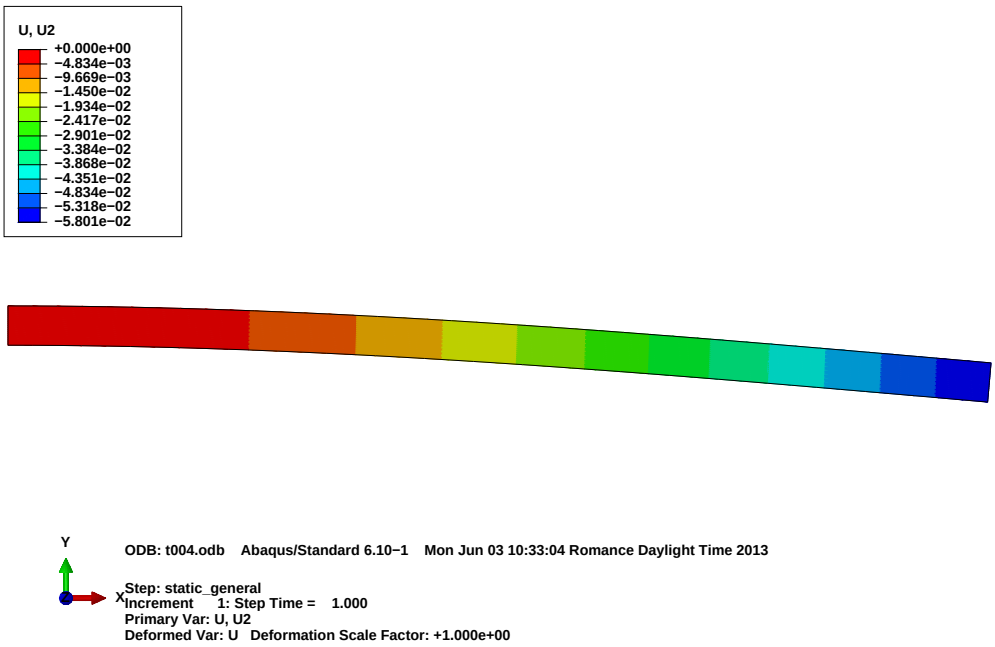


Figure 8.1: Deformation in y-axis of beam with thickness $t = 0.04m$

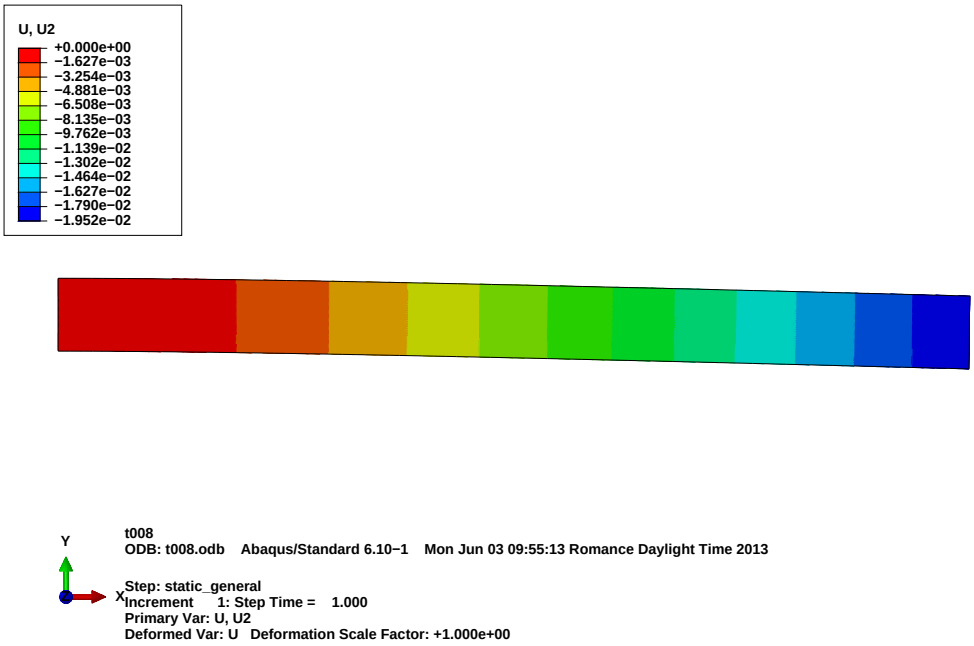


Figure 8.2: Deformation in y-axis of beam with thickness $t = 0.08m$

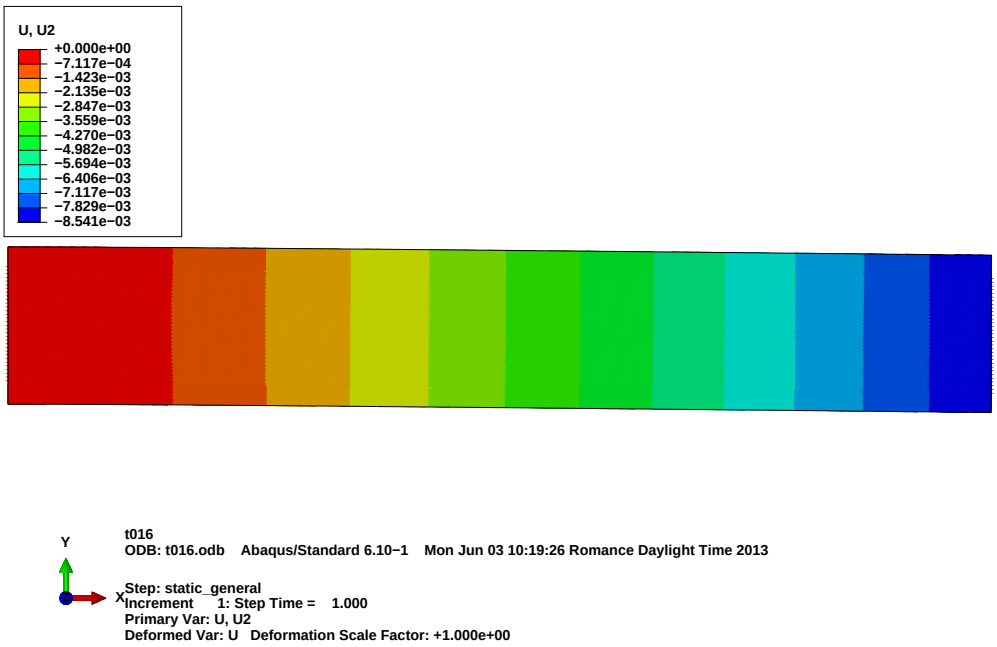


Figure 8.3: Deformation in y-axis of beam with thickness $t = 0.16m$

As can be seen from the deflection results, the foam has tremendous effect on the behavior of the structure. The results can also be read from the Table8.1 with the total mass values for each structure. The difference between the first and the last case is in the factor of 7 with the mass increase around 5%.

Total Thickness $[mm]$	Mass $[kg/m]$	Maximum Deflection in Y axis $[mm]$
40	62	58.01
80	66	19.52
160	74	8.54

Table 8.1: Maximum bending moments for different layups

Moreover, from the σ_{11} and σ_{12} plots, as shown below (see the Fig.8.4) it can be concluded that a foam layer with added tiny mass is actually causing a dramatic drop in terms of σ_{11} and σ_{12}.

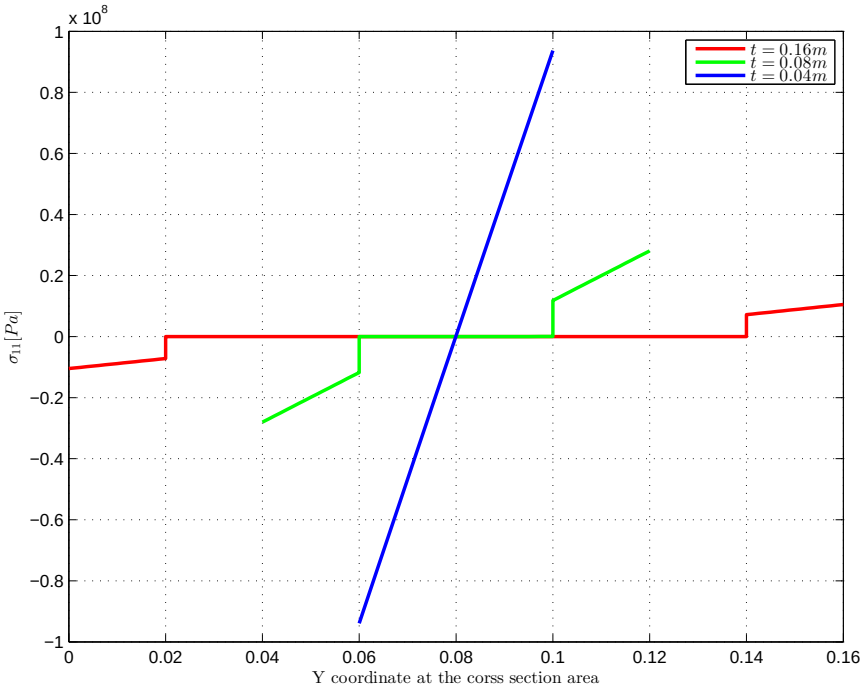


Figure 8.4: σ_{11} distribution in the cross section at the center of the beam

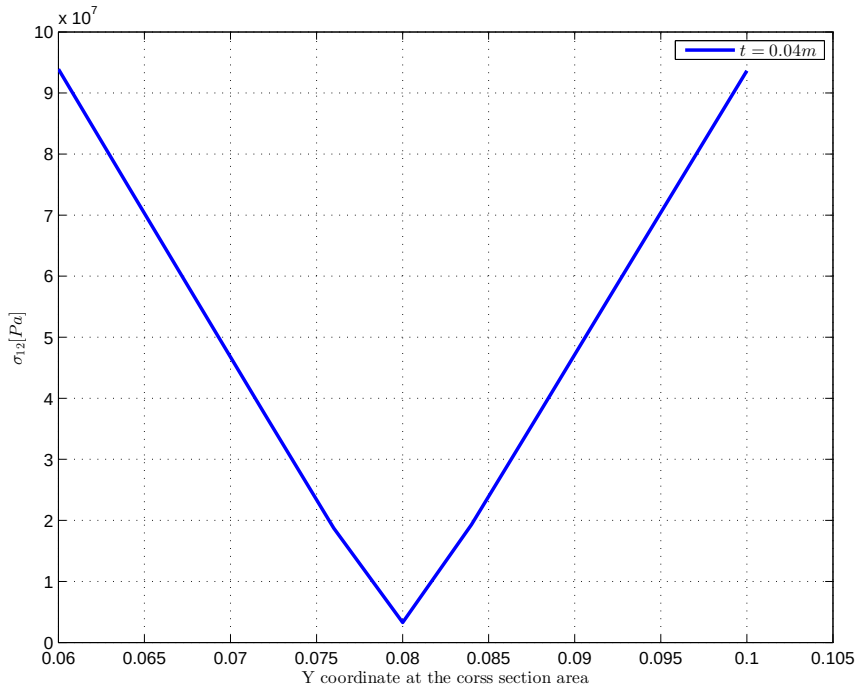


Figure 8.5: σ_{12} shear stress distribution in the cross section at the center of the beam $t = 0.04m$

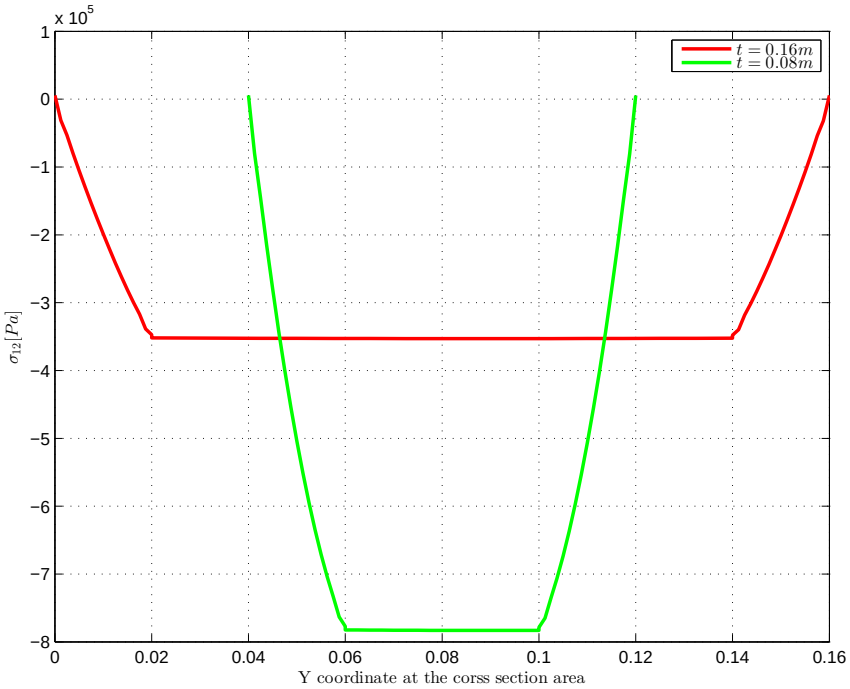


Figure 8.6: σ_{12} shear stress distribution in the cross section at the center of the beam $t = 0.08m$ and $t = 0.16m$

In the first plot, the pressure is increasing with height yet for the thicker structure, since the moment can be handled from a longer distance, the less force is required which lowers the σ_{11} value for the 3rd case.

It can also be seen that, since the stiffness of the foam is relatively small, the main load is being handled by faces where the epoxy-glass material is being used. Therefore, in the plots, the normal stress is almost zero on the foam surface. Nevertheless, the foam material is under shear stress which can be seen from the second plot. Due to this reason, it is better to select foam material considering that it needs to withstand shear stresses in the structure.

From the results, it is concluded that it is efficient to use the structure with core materials with a small increase of weight.

8.2 Bending Of The Beam

In this section, the bending of the beam with the thickest foam layer is investigated. The beam was put up on 2 roller bearings and its shape is investigated under the loading.

The loading is given via 4 points as also been done for the previous section. The illustration of this point loads are shown right hand side of the Fig.8.7. The total $50kN$

is distributed

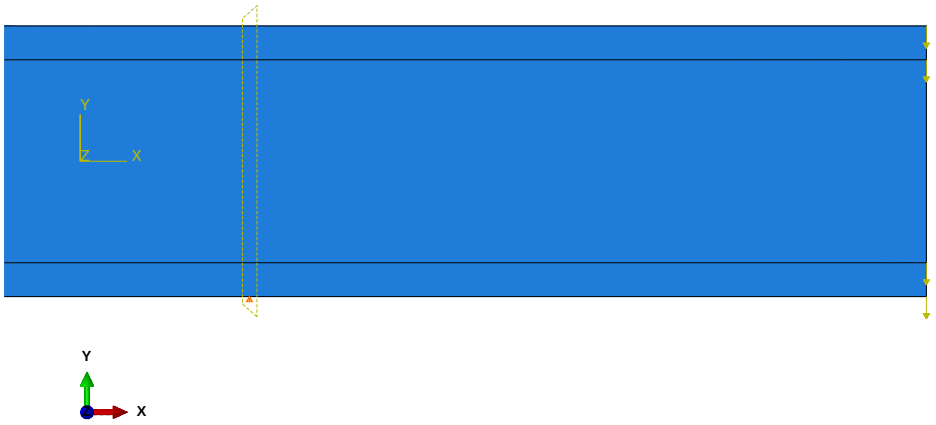


Figure 8.7: The points where forces are defined

The deformation occurred on the beam is shown in Fig.8.8.

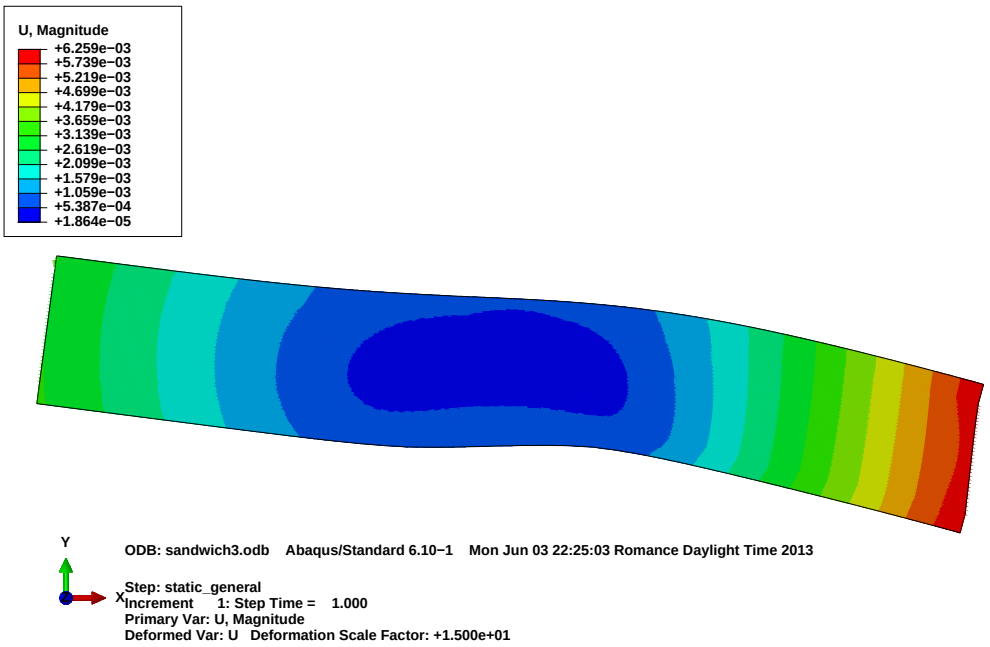


Figure 8.8: Deformation of the beam

It may not be the general expected result but due to inside-forces the structure is under shear stress and results as shown in the figure. The movement on the left hand side is on positive 'y' direction and negative on the right hand side due to applied force. Moreover, the edges of the beam don't stand parallel to the y-axis after the loading. They both forced to rotate clockwise due to the transfered shear forces inside the structure via roller bearings. And due to this behavior, the free edge rises up and edge surface becomes non-parallel to the y-axis.

Chapter 9

Sandwich Beam Subjected to Uniform Pressure

In this chapter, a beam structure will be developed. The main aim is to have a light sandwich structure which can withstand 0.1 MPa pressure with maximum $\sigma_{cr} = 100\ MPa$ critical stress and $\omega_{max} = L/50 = 20\ mm$ maximum deformation.

For the faces, the epoxy-glass material, which was used also in the previous chapters, is used. And for the core material, *H 80* type picked up from the table after some iterations.

In the first part the analytical calculations are done. The main aim was to get the results from fast calculations and make the adjustments on the FEA interface where the calculations are slow but reliable.

9.1 Analytic Solutions

In order to reach a light structure which could withstand the requirements, a Matlab script is developed which can be found in AppendixA.6.

The first one, Fig.9.1 shows if the structure is able to withstand with the given requirements. Via Matlab and analatical equations, each setup is calculated for given limitations and plotted with a 2d color plot. The axes are showing the thickness values for corresponding material and colors are for the result for specific case. Explanations for the colors are stated below the figure in the Table9.1.

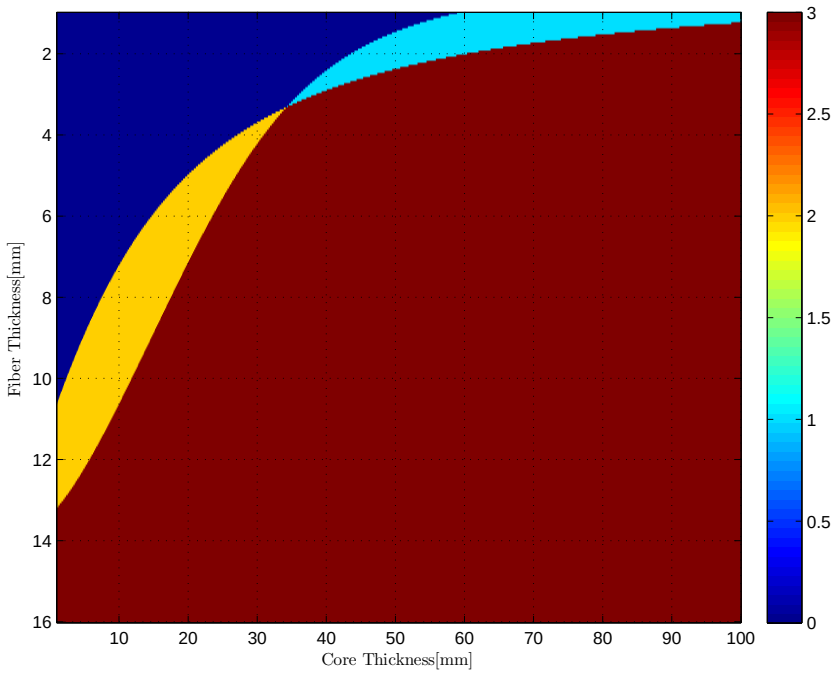


Figure 9.1: Index caluclation with varying face and core thicknesses

Color	Numeric	Details
Dark Blue	0	Both critical stress and maximum deflection are higher than the limit
Light Blue	1	Only the critical stress is higher than the limit
Yellow	2	Only the maximum deformation is higher than the limit
Brown	3	Both requirements are satisfied

Table 9.1: Details of Fig.9.1

The mass of the different setups are also calculated with the same way. The Fig.9.2 shows the weight of the each setup in kilograms. Due to lower density of the core material, the variety of the color lines are relatively low in parallel to x-axis for a fixed y or so to say, fiber thickness value. In order to have a lower mass, the higher core side of the brown area are selected. Nevertheless, in order to be secure, some extra weight and material are satisfied and as a result of this, the picked point is not very close to any of the color boundaries.

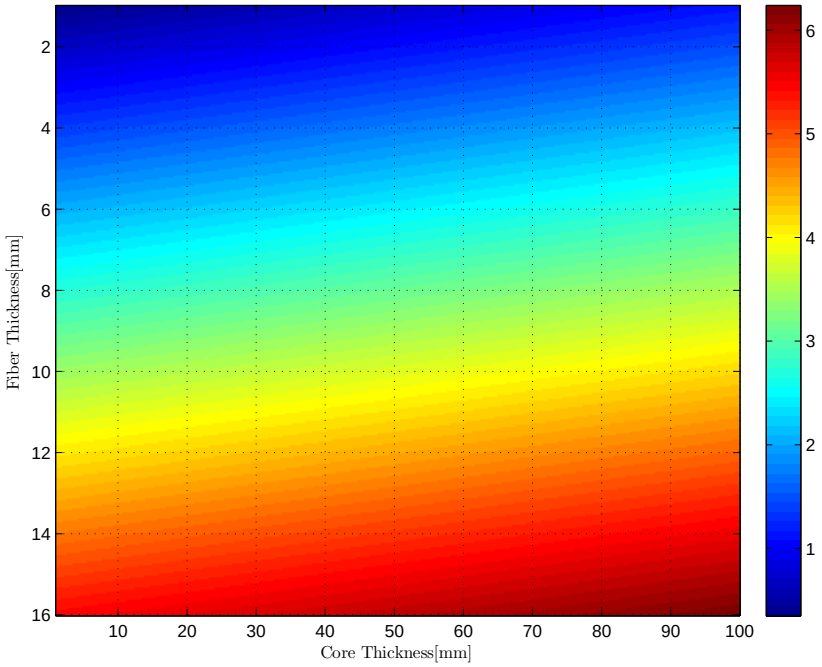


Figure 9.2: Masses with varying face and core thicknesses

From these calculations for the first step, the thicknesses for fiber and core material are decided to be 6mm and 60mm respectively.

For the next step, the deflection and maximum stress values are re-calculated for selected thickness setup. The results are shown below.

core[mm]	fiber [mm]	mass [kg]	maximum deflection[mm]	max stress [MPa]
60	6	2.52	7.37	31.4

Table 9.2: Results from analytical calculations

After this point, the picked thickness setup is transfered to ABAQUS and the structure is loaded with the same load.

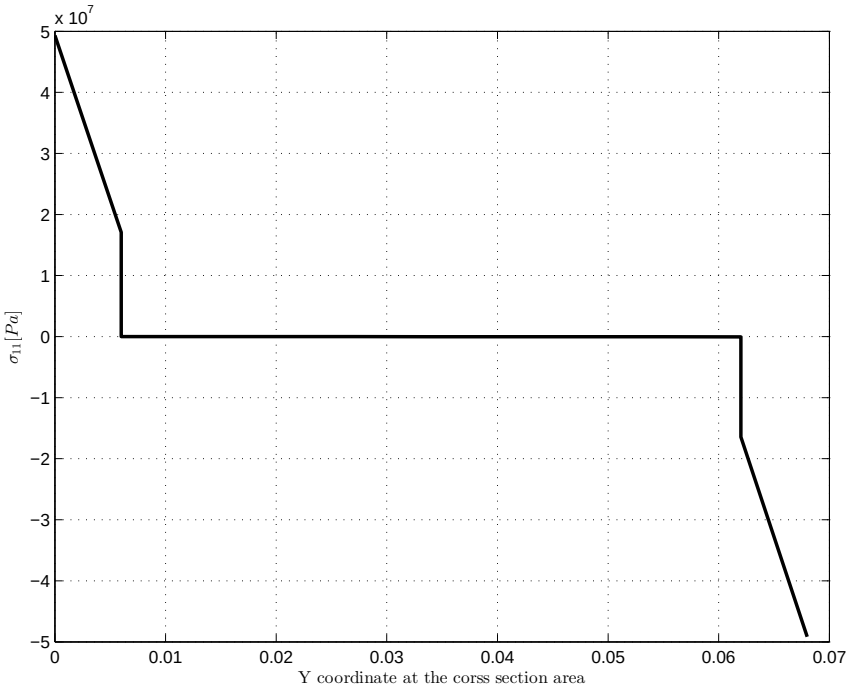


Figure 9.3: σ_{11} @ the mid-section of the beam

The stress distribution in the mid-section area is shown in the Fig.9.3 and is also shown with the results from the analytic calculation in the Table9.3.

core[mm]	fiber [mm]	mass [kg]	maximum deflection[mm]	max stress @ mid section [MPa]
60	6	2.52	7.37	31.4
60	6	2.31	13.2	49.3

Table 9.3: Results from analytical(upper) and FEA (lower) calculations

As can be seen from the Fig.9.4 the stress values are relatively high close to the boundary conditions. Although it was below the required critical limit, to decrease this stresses some more, the fiber thickness is increased from 6mm to 8mm for each face and the core thickness is kept constant. The updated result can be seen from the Fig.9.5.

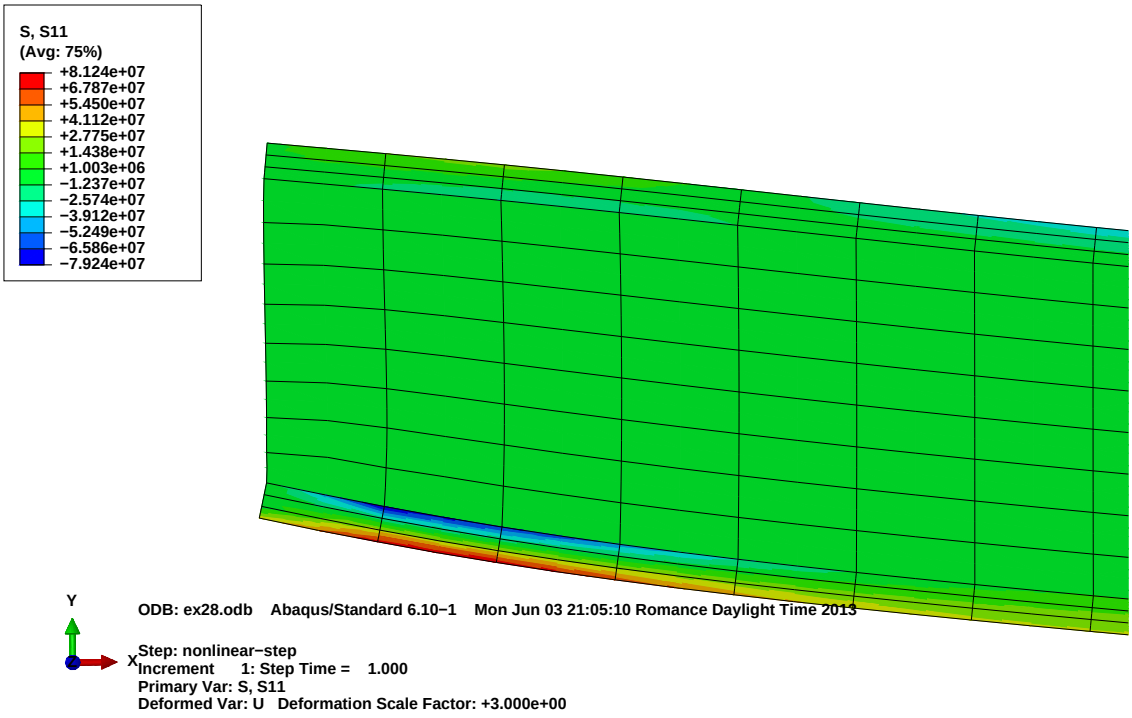


Figure 9.4: σ_{11} @ the bottom corner

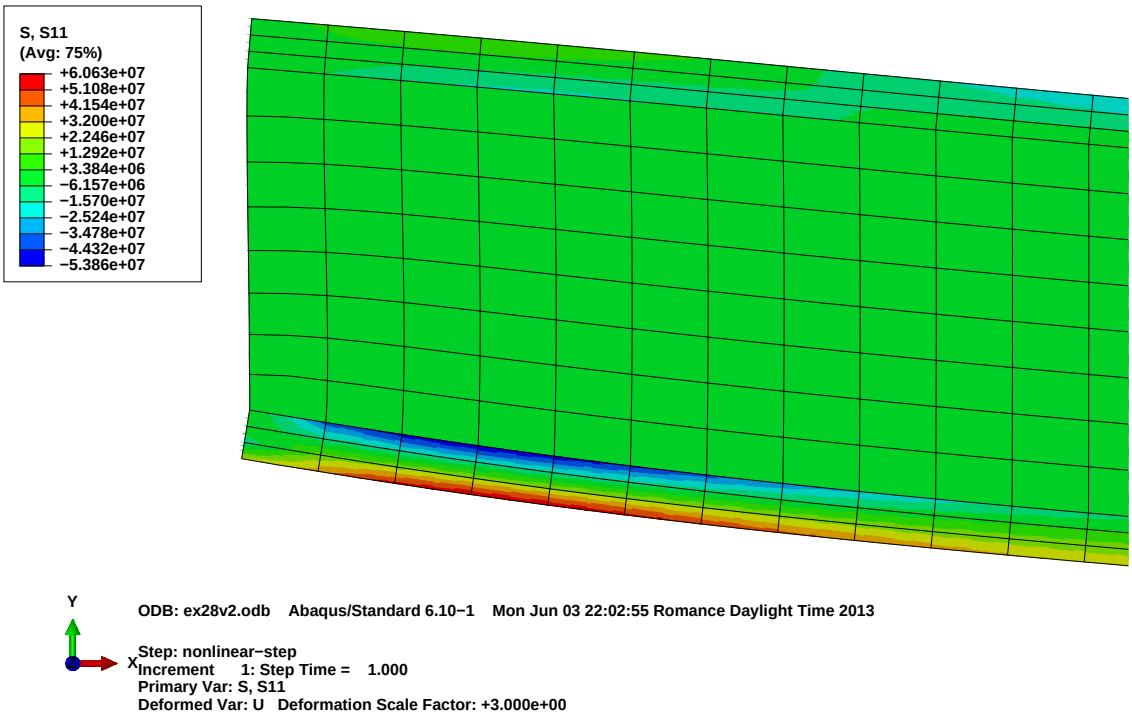


Figure 9.5: σ_{11} @ the bottom corner with thicker fiber surface

The new results from the FEA results are also shown below.

core[mm]	fiber [mm]	mass [kg]	maximum deflection[mm]	max stress @mid section [MPa]
60	8	3.01	11.08	40.31

Table 9.4: Results from FEA with increased surface thickness

Further improvements can be done for this model yet provided setup is expected to safely withstand the load with the requirements.

Appendix A

Matlab Functions

A.1 Micro-mechanics of Composites

A.1.1 Thickness of an unspecified laminate type using ROM

Given area masses and densities

```
function [ h ] = thickness_ROM( n_o_m,W,rho )
%thickness_ROM Calculate the composite's thickness
%Given : area masses and densities
%   h:thickness of composite
%   n_o_m:number of materials
%   W:weight per unit area vector [length(W)=n_o_m]
%   rho:density vector [length(rho)=n_o_m]

h=0;
for i=1:n_o_m
    h=h+W(i)/rho(i);
end
end
```

Given fiber volume fraction, densities and total area mass

```
function [ h ] = thickness_ROM2( n_o_m,v,rho,M)
%thickness_ROM2 Calculate the composite's thickness
%Given : fiber volume fraction, densities, total area mass
%   h:thickness of composite
%   n_o_m:number of materials
%   M:total mass per area
%   v:volume fraction vector [length(v)=n_o_m]
%   rho:density vector [length(rho)=n_o_m]
h=0;
for i=1:n_o_m
    w(i)=rho(i)*v(i)/sum(rho.*v); %calculate the weight fractions
    W(i)=w(i)*M; % calculate the weight per unit area for each material
```

```
13      h=h+W(i)/rho(i); % calculate the thickness and sum
14  end
15  end
```

A.1.2 Stiffnesses, Shear Modulus and Poisson Ration Calculation with ROM

```
1  function [ E1,E2,G12,v12 ] = modulus_calc( e1,e2,g12,vf,v )
2  %modulus_calc. calculates the E1,E2,G12,v12 for composite
3  %   All inputs should be 1D arrays!
4  %   vf:Volume friction vector length(v)=number of materials
5  %   e1,e2,g12:Modulusses vectors
6  %   e1(i),e2(i):Young modulusses for the material 'i'
7  %   g12(i):Shear modulusses for the material 'i'
8  %   v:Poisson ratio vector
9
10 if sum(vf)~=1 % check if volume friction vector is correct
11     error('Total volume friction can''t be more than 1 ')
12 end
13 E1=sum(e1.*vf);
14 E2=e2(1)*e2(2)/(vf(1)*e2(2)+vf(2)*e2(1));
15 G12=sum(vf./g12)^-1;
16 v12=sum(vf.*v);
17 end
```

A.1.3 In-Plane Stiffnesses with Efficiency Factor

```
1  function [ E1,E2 ] = modulus_calc_eff(e1,e2,vf,eff)
2  %modulus_calc_eff. calculates inplane stiffnesses for composite
3  %   All inputs should be 1D arrays!
4  %   vf:Volume friction vector length(v)=number of materials
5  %   e1,e2:Modulusses vectors
6  %   e1(i),e2(i):Young modulusses for the material 'i'
7  %   eff:Efficiency vector
8
9  if sum(vf)~=1 % check if volume friction vector is correct
10     error('Total volume friction can''t be more than 1 ')
11 end
12 E1=sum(eff.*e1.*vf);
13 E2=sum(vf./(e2.*eff))^-1;
14 end
```

A.1.4 Thermal and Moisture Expansion for a UD-Laminate

```
function [ eps_1,eps_2 ] = hygrothermal_strain(alfa,beta,dt,dc)
%hygrothermal_strain calculates the total strain under mosture and ...
    temp.
%change
%   All inputs should be 1D arrays!
%   alfa: CTE [axial transverse]
%   beta: Moisture expansion [axial transverse]
%   dt: Temperature change
%   dc: Change in moisture content

eps_1=alfa(1)*dt+beta(1)*dc;
eps_2=alfa(2)*dt+beta(2)*dc;
end
```

A.2 Mechanics of a Lamina

A.2.1 PART 1

Compliance3 function

```
function [ S ] = ...
    compliance3(E1,E2,E3,G12,G23,G31,nu12,nu21,nu31,nu13,nu23,nu32)
%Compliance3 matrix generator
% [ S ] = ...
    compliance3(E1,E2,E3,G12,G23,G31,nu12,nu21,nu31,nu13,nu23,nu32)
%
% |eps1 |   |   |   |sigma1 |
% |eps2 |   |   |   |sigma2 |
% |eps3 | = | S | . |sigma3 |
% |eps23|   |   |   |sigma23|
% |eps31|   |   |   |sigma31|
% |eps12|   |   |   |sigma12|

S = sparse(6,6);
S(1,1)=1/E1;
S(1,2)=-nu21/E2;
S(1,3)=-nu31/E3;
S(2,1)=-nu12/E1;
S(2,2)=1/E2;
S(2,3)=-nu32/E3;
S(3,1)=-nu13/E1;
S(3,2)=-nu23/E2;
```

```
21 S(3,3)=1/E3;
22 S(4,4)=1/G23;
23 S(5,5)=1/G31;
24 S(6,6)=1/G12;
25 end
```

A.2.2 PART 2

Stiffness matrix generator

```
1  function [ Q ] = stiffness2(E1, E2, G12, nu12)
2
3  %% Stiffness2 matrix generator
4  % [ Q ] = stiffness2(E1, E2, G12, nu12)
5  %
6  % |sigma1 |     |   |   |eps1 |
7  % |sigma2 | =   | Q | . |eps2 |
8  % |sigma12|     |   |   |eps12|
9
10 nu21=nu12*E2/E1;
11 Q = (1/(1-(nu12*nu21)))*[    E1    nu21*E1      0  ;
12                           nu12*E2    E2         0  ;
13                            0        0    G12*(1-nu12*nu21)];
14 end
```

Stiffness matrix rotater

```
1  function [ Qg ] = stiffness_matrix_rotater(Q,theta)
2
3  %Rotates the stiffness matrix
4  %[ Qg ] = stiffness_matrix_rotater(Q,theta)
5  %theta --> angle!
6
7  Q11=Q(1,1);Q12=Q(1,2);Q16=Q(1,3);
8  Q22=Q(2,2);Q26=Q(2,3);
9  Q66=Q(3,3);
10 c=cosd(theta);
11 s=sind(theta);
12 Qg11=Q11*c^4+2*(Q12+2*Q66)*c^2*s^2+Q22*s^4;
13 Qg12=(Q11+Q22-4*Q66)*c^2*s^2+Q12*(s^4+c^4);
14 Qg22=Q11*s^4+2*(Q12+2*Q66)*c^2*s^2+Q22*c^4;
15 Qg16=(Q11-Q12-2*Q66)*s*c^3+(Q12-Q22+2*Q66)*s^3*c;
16 Qg26=(Q11-Q12-2*Q66)*s^3*c+(Q12-Q22+2*Q66)*s*c^3;
17 Qg66=(Q11+Q22-2*Q12-2*Q66)*s^2*c^2+Q66*(s^4+c^4);
```

```
Qg=[Qg11 Qg12 Qg16;Qg12 Qg22 Qg26;Qg16 Qg26 Qg66];
end
```

Strain rotater XY

```
function [ T ] = rotate2_strain(theta)

%Strain axis transform matrix generator
% [ T ] = rotate2_strain(theta)
% |eps1 |    |   |   |epsx |
% |eps2 | =  | T | . |epsy |
% |eps12|    |   |   |epsxy|

T=[cosd(theta)^2                 sind(theta)^2                    ...
    sind(theta)*cosd(theta)  ;
   sind(theta)^2                 cosd(theta)^2                    ...
      -sind(theta)*cosd(theta) ;
   -2*sind(theta)*cosd(theta)    2*sind(theta)*cosd(theta)      ...
      cosd(theta)^2-sind(theta)^2];
end
```

Strain rotater XYZ

```
function [ T ] = rotate3_strain(theta)

%Strain axis transform matrix generator
% T(theta)
%
% |eps11|    |   |   |epsxx|
% |eps22|    |   |   |epsyy|
% |eps33| =  | T | . |epszz|
% |eps23|    |   |   |epsyz|
% |eps13|    |   |   |epsxz|
% |eps12|    |   |   |epsxy|

T = sparse(6,6);
T(1,1)=cosd(theta)^2;
T(1,2)=sind(theta)^2;
T(1,6)=sind(theta)*cosd(theta);
T(2,1)=sind(theta)^2;
T(2,2)=cosd(theta)^2;
T(2,6)=-sind(theta)*cosd(theta);
T(3,3)=1;
T(4,4)=cosd(theta);
T(4,5)=-sind(theta);
```

```
T(5,4)=sind(theta);
T(6,1)=-2*sind(theta)*cosd(theta);
T(6,2)=2*sind(theta)*cosd(theta);
T(6,6)=(sind(theta)^2)*(cosd(theta)^2);
end
```

A.3 Mechanics of Laminate

A.3.1 ABD Matrix Generator

```
function [ ABD,A,B,D ] = ABDgenerator(Q,z,n)
% >>>  use stiffness2 for generating the stiffness matrices
% >>>  [ Q ] = stiffness2(E1, E2, G12, nu12)
% >>>  use  stiffness_matrix_rotater for rotating the plies
% >>> [ Qg ] = stiffness_matrix_rotater(Q,theta)
%
% [ ABD,A,B,D ] = ABDgenerator(Q,z,n)
% 3D stiffness matrices Qi=Q(:,:,i)
%
% z=[z(0) z(1) z(2) ... z(n)]
% i.e. z=[-h 0 h] or z=[3*h/2 -h/2 0 h/2 3*h/2]
% n=number of laminas
%
% *start numbering laminas from top to bottom
% **if n-->odd, n=n+1, duplicate the mid ply in seq. and z vector
%
% |Nx |          |epsx |
% |Ny |          |epsy |
% |Nxy| =  ABD * |epsxy|
% |Mx |          |Kx   |
% |My |          |Ky   |
% |Mxy|          |Kxy  |

[A,B,D]=deal(zeros(3,3));
for i = 1:n
    A=A+Q(:,:,i)*(z(i+1)-z(i));
    B=B+Q(:,:,i)*1/2*(z(i+1)^2-z(i)^2);
    D=D+Q(:,:,i)*1/3*(z(i+1)^3-z(i)^3);
end
ABD=[A B;B D];
end
```

A.3.2 Validation for ABD

Example 3.6 from the lecture book - A matrix calculation

```
%% example 3_6
% SEQUENCE : [0 90 45 -45 -45 45 90 0]
addpath(genpath('C:\Users\Gonzales\Desktop\Dropbox\Fibre ...
    Composites\functions\'));
addpath(genpath('C:\Users\s111058\Desktop\Dropbox\Fibre ...
    Composites\functions\'))
E1=147e3
E2=9e3
G12=3.3e3
nu12=0.31
seq=[0 90 45 -45 -45 45 90 0]
%Calculate the stiffness matrix
Q1=stiffness2(E1, E2, G12, nu12)
%Rotate stiffnesses w.r.t. angles
for i=1:length(seq)
    Q(:,:,i)=stiffness_matrix_rotater(Q1,seq(i))
end
h=0.127
z=[-4*h -3*h -2*h -h 0 h 2*h 3*h 4*h]
n=length(seq)
%Get the ABD
[ ABD,A,B,D ] = ABDgenerator(Q,z,n)
eps_xy = ABD\([100 100 50 0 0 0]')
T_eps=rotate2_strain(45)
eps_12=T_eps*eps_xy(1:3)
Q1*eps_12
```

Example 3.7 from the lecture book - D matrix calculation due to bending load

```
%% example 3_7
% SEQUENCE :[0 90 0]
addpath(genpath('C:\Users\Administrator\Dropbox\Fibre ...
    Composites\functions'))
addpath(genpath('C:\Users\s111058\Desktop\Dropbox\Fibre ...
    Composites\functions\'))
E1=147e3
E2=9e3
G12=3.3e3
nu12=0.31
h=0.127
```

```
n=4 % number of layers + 1 due to the mid layer which is divided to ...
    two parts
seq=[0 90 90 0]
z=[-3*h/2 -h/2 0 h/2 3*h/2]
%Calculate the stiffness matrix
Q1=stiffness2(E1, E2, G12, nu12)
%Rotate stiffnesses w.r.t. angles
for i=1:n
    Q(:,:,i)=stiffness_matrix_rotater(Q1,seq(i))
end
%Get the ABD
[ ABD,A,B,D ] = ABDgenerator(Q,z,n)
inv(ABD)* [0 0 0 1 0 0]'
```

A.4 Strength of Laminates

A.4.1 Failure Criteria Calculations

Maximum Stress

```
%% 2.4
sigma_max_1t=1000; %Mpa
sigma_max_1c=1000; %Mpa
sigma_max_2t=35;   %Mpa
sigma_max_2c=150;  %Mpa
sigma_max_12=50;   %Mpa
eps_max_1t=sigma_max_1t/(E1/10^6);  %mm
eps_max_1c=sigma_max_1c/(E1/10^6);  %mm
eps_max_2t=sigma_max_2t/(E2/10^6);  %mm
eps_max_2c=sigma_max_2c/(E2/10^6);  %mm
eps_max_12=sigma_max_12/(G12/10^6); %mm
%% max stress failure index => f-->1 / FAIL!
for i = 1:n
    if -sigma_max_1c<sigma_12(1,:,i) && ...
       sigma_12(1,:,i)<sigma_max_1t && ...
       -sigma_max_2c<sigma_12(2,:,i) && ...
       sigma_12(2,:,i)<sigma_max_2t &&  ...
       abs(sigma_12(3,:,i))<sigma_max_12
        f(i)=0
    else
        f(i)=1 % failure!
    end
end
```

Maximum Strain

```
%% max strain
for i = 1:n
    if -eps_max_1c<strain_12(1,:,i) && strain_12(1,:,i)<eps_max_1t ...
       && -eps_max_2c<strain_12(2,:,i) && ...
       strain_12(2,:,i)<eps_max_2t &&  ...
       abs(strain_12(3,:,i))<eps_max_12
        f(i)=0
    else
        f(i)=1 % failure!
    end
end
```

Tsai-Hill

```
%% Tsai-Hill
for i = 1:n
% get the correct max_sigma for compression or tensile
    if sigma_12(1,:,i) < 0
        sigma_max_1=sigma_max_1c;
    else
        sigma_max_1=sigma_max_1t;
    end
    if sigma_12(2,:,i) < 0
        sigma_max_2=sigma_max_2c;
    else
        sigma_max_2=sigma_max_2t;
    end

f(i)=(sigma_12(1,:,i)^2/sigma_max_1^2 ...
    -sigma_12(1,:,i)*sigma_12(1,:,i)/sigma_max_1^2 ...
    +sigma_12(2,:,i)^2/sigma_max_2^2 ...
    +sigma_12(3,:,i)^2/sigma_max_12^2);

end
```

Tsai-Wu

```
F12_s=0; %arbitrarily select // either -0.5 or 0
F11=1/(sigma_max_1t*sigma_max_1c);
F22=1/(sigma_max_2t*sigma_max_2c);
F12=sqrt(F11*F22)*F12_s;
```

```
%calculate the index value
for i = 1:n
    f(i)=(sigma_12(1,:,i)^2/(sigma_max_1t*sigma_max_1c) ...
        +sigma_12(2,:,i)^2/(sigma_max_2t*sigma_max_2c) ...
        +sigma_12(3,:,i)^2/sigma_max_12^2 ...
        +2*F12*sigma_12(1,:,i)*sigma_12(2,:,i) ...
        +sigma_12(1,:,i)/sigma_max_1t -sigma_12(1,:,i)/sigma_max_1c ...
        +sigma_12(2,:,i)/sigma_max_2t -sigma_12(2,:,i)/sigma_max_2c);
end
```

A.4.2 Progressive Failure with Tsai-Hill

```
%% Progressive Failure
nx=300;ny=-50;nxy=25; % N/mm
mx=50;my=0;mxy=0  ;   % N
f=[0 0 0 0]; %failure vector
ff=1; %display switcher for failure type FPF/LPF
chck=[0 3]; %control switcher for failure type FPF/LPF
order=0; %Ply failure order vector
for p=1:100
    % get the new load case
    Nx=p/100*nx;Ny=p/100*ny;Nxy=p/100*nxy; % N/mm
    Mx=p/100*mx;My=p/100*my;Mxy=p/100*mxy; % N
    % re-calculate the ABD matrix
    for i = 1:n
        Q(:,:,i)= Q(:,:,i)*~f(i); %use ~f to cancel out the ...
            corresponding failed stiffness matrix
    end
    [ ABD,A,B,D ] = ABDgenerator(Q,z,n);

    s_xy=ABD\[Nx Ny Nxy Mx My Mxy]'; % total strains on the middle ...
        axis _xy
    save_s_xy(:,p)=s_xy; % save the strains for corresponding step
    % get the total strains and stresses for each lamina
    for i = 1:n
        m=mean([z(i) z(i+1)]); %middle axes of each ply
        sum_strain_xy(:,:,i)=[s_xy(1)+s_xy(4)*m; %eps + z * curvature
            s_xy(2)+s_xy(5)*m;
            s_xy(3)+s_xy(6)*m];
            %Rotate strains from xy to 12 coordinate system
        strain_12(:,:,i)=rotate2_strain(-seq(i))*sum_strain_xy(:,:,i);
        sigma_12(:,:,i)=Q1*strain_12(:,:,i); % stresses on each ply ...
            on principle lamina coordinates
    end
    f_old=f; %allocation of the failure index vector
    % apply the Tsai-Hill
    for i = 1:n
```

```
33          if sigma_12(1,:,i) < 0
34              sigma_max_1=sigma_max_1c;
35          else
36              sigma_max_1=sigma_max_1t;
37          end
38          if sigma_12(2,:,i) < 0
39              sigma_max_2=sigma_max_2c;
40          else
41              sigma_max_2=sigma_max_2t;
42          end
43
44          if (sigma_12(1,:,i)^2/sigma_max_1^2 ...
               -sigma_12(1,:,i)*sigma_12(1,:,i)/sigma_max_1^2  ...
               +sigma_12(2,:,i)^2/sigma_max_2^2  ...
               +sigma_12(3,:,i)^2/sigma_max_12^2)<1
45              f(i)=0; %passed!
46          else
47              f(i)=1; %failed!
48          end
49      end
50   % put the failures in an order
51      if max(f-f_old)>0
52          if order > 0
53              order=[order find((f-f_old)==1)];
54          else
55              order=find((f-f_old)==1);
56          end
57      end
58      if (length(find(f==1)))>chck(ff)
59
60          dp=[{'FPF'} {'LPF'}];
61
62          disp([dp{ff},' has occured @ ',num2str(p), '% load'])
63          ff=ff+1;
64      end
65      if (length(find(f==1)))>chck(2)
66          disp(['Failure order= ',num2str(order)])
67          break
68      end
69  end
70  s = get(0, 'ScreenSize');
71  figure('Position', [0 30 s(3) s(4)-96]);
72  plot([1:84],abs(save_s_xy(1,:)),'b')
73  hold on
74  plot([1:84],abs(save_s_xy(2,:)),'r')
75  grid on
76  legend('$\epsilon_{1}$','$\epsilon_{2}$')
77  xlabel('Load $\%$')
78  ylabel('Strain $\epsilon$ [mm]')
79  saveas(gcf,'pf5','epsc2')
```

A.5 Composite Plates

A.5.1 Maximum Deflections Under A Unifrom Load

Layup:[0 0 0 0]s with headers

```
clear all
clc
addpath(genpath('C:\Users\s111058\Desktop\Dropbox\Fibre ...
    Composites\functions')) %Databar computer
addpath(genpath('C:\Users\Administrator\Dropbox\Fibre ...
    Composites\functions'))
set(0,'defaulttextinterpreter','latex')
%% Material properties
E1=181e9 %[Pa]
E2=10.3e9 %[Pa]
G12=7.17e9 %[Pa]
nu12=0.28
h=0.5e-3 %[m]
%plate dimensions
a=0.5 %[m]
b=1 %[m]
%applied surface load
q=1e3 %[Pa]

%% Layup:[0 0 0 0]s
n=8 %number of layers
seq=[0 0 0 0 0 0 0 0] %sequence

z=[-4 -3 -2 -1 0 1 2 3 4]*h;
%calculate the stiffness matrix
Q1=stiffness2(E1, E2, G12, nu12)

%rotate the stiffness matrices
for i = 1:n
    Q(:,:,i) = stiffness_matrix_rotater(Q1,seq(i));
end

%calculate the ABD matrix
[ ABD,A,B,D ] = ABDgenerator(Q,z,n);

%get the required elements from D matrix
D11=D(1,1);D12=D(1,2);D22=D(2,2);D66=D(3,3);D16=D(1,3);

%generate a mesh up on the rectangular shap
x=0:0.025:.5; y=0:0.025:1; [X,Y] = meshgrid (x,y);

```

```
%calculate the max. deflection in z-axis and the bending moments
W=0;d2wdx2=0;d2wdy2=0;d2wdxy=0;
dx=0.020;dy=0.020;
x=0:dx:.5; y=0:dy:1; [X,Y] = meshgrid (x,y);
clear n
        for m=1:2:100
            m
            for n=1:2:100
                q_mn=(16.*q./(m.*n.*pi^2));
                W=W+((  q_mn*sin(m*pi*X./a).*sin(n*pi*Y./b) )  ./  ...
                    (D11*( (m*pi/a)^4 +  ...
                    (2*(D12+2*D66)/D11)*(m*pi/a)^2*(n*pi/b)^2 + ...
                    (D22/D11)*(n*pi/b)^4)));
           end
        end
mesh(X,Y,W)

%calculate the deriavatives
d2wdx2=diff((diff(W')')')'./dx^2;
d2wdy2=diff(diff(W))./dy^2;
d2wdxy=diff((diff(W')'))./(dx*dy);
d2wdx2=d2wdx2(3:end,:);
d2wdy2=d2wdy2(:,3:end);
d2wdxy=d2wdxy(2:end,2:end);

%moments
Mx=-(D11.*d2wdx2+D12.*d2wdy2+2*D16*d2wdxy);My=-D12.*d2wdx2-D22.*d2wdy2;

%maximums
W_max=max(max(W));Mx_max= max(max(Mx));My_max=max(max(My));

%Bucklings
a=1; b=0.5;
n=1;m=1; %1st. mode
P(1)=pi^2*((D11*(m/a)^2)+2*(D12+2*D66)*(1/b)^2+D22*(a/m)^2*(1/b)^4);
n=1;m=2; %2nd. mode
P(2)=pi^2*((D11*(m/a)^2)+2*(D12+2*D66)*(1/b)^2+D22*(a/m)^2*(1/b)^4);
n=1;m=3; %3th. mode
P(3)=pi^2*((D11*(m/a)^2)+2*(D12+2*D66)*(1/b)^2+D22*(a/m)^2*(1/b)^4);
P=P*10^-3 % [N/mm]

%eigen frequencies
rho=1.4*10^-3*10^6; %kg/m^3
rho_s=rho*(h*8); %kg/m^2
for n=1:5;for m=1:5;
w(m,n)=sqrt((D11*(m*pi/a)^4 +2*(D12+2*D66)*(m*pi/a)^2*(n*pi/b)^2 ...
    +D22*(n*pi/b)^4)/rho_s);
end;end
w=w/(2*pi);
%drawing deflection distribution
figure('name','[0 0 0 0]s')
```

```
mesh(X,Y,W)
xlabel('$a=1m$');ylabel('$b=0.5m$');zlabel('$Deflection[m]$')
set(gca, 'xlim', [0 .5], 'ylim', [0 1])
saveas(gca, 'p1.eps','eps');

%drawing moment distribution
X=X(3:end,3:end);
Y=Y(3:end,3:end);
figure('name','[0 0 0 0]sM')
subplot(1,2,1);mesh(X,Y,Mx)
set(gca, 'xlim', [0 0.5], 'ylim', [0 1])
xlabel('$a=1m$');ylabel('$b=0.5m$');zlabel('$M_x[N]$')
subplot(1,2,2);mesh(X,Y,My)
set(gca, 'xlim', [0 0.5], 'ylim', [0 1])
xlabel('$a=1m$');ylabel('$b=0.5m$');zlabel('$M_y[N]$')
saveas(gca, 'p1m.eps','eps');
```

Layup:[90 90 90 90]s

```
%% Layup:[90 90 90 90]s
a=0.5; b=1;
n=8 %number of layers
seq=[90 90 90 90 90 90 90 90] %sequence
z=[-4 -3 -2 -1 0 1 2 3 4]*h;
%calculate the stiffness matrix
Q1=stiffness2(E1, E2, G12, nu12)

%rotate the stiffness matrices
for i = 1:n
    Q(:,:,i) = stiffness_matrix_rotater(Q1,seq(i));
end

%calculate the ABD matrix
[ ABD,A,B,D ] = ABDgenerator(Q,z,n);

%get the required elements from D matrix
D11=D(1,1);D12=D(1,2);D22=D(2,2);D66=D(3,3);D16=D(1,3);

%generate a mesh up on the rectangular shap
x=0:0.025:1; y=0:0.025:.5; [X,Y] = meshgrid (x,y);

%calculate the max. deflection in z-axis and the bending moments

W=0;d2wdx2=0;d2wdy2=0;d2wdxy=0;
dx=0.020;dy=0.020;
x=0:dx:0.5; y=0:dy:1; [X,Y] = meshgrid (x,y);

        for m=1:2:100
```

```
30              for n=1:2:100
31                  q_mn=(16.*q./(m.*n.*pi^2));
32                  W=W+((  q_mn*sin(m*pi*X./a).*sin(n*pi*Y./b) )  ./  ...
                        (D11*( (m*pi/a)^4 +  ...
                        (2*(D12+2*D66)/D11)*(m*pi/a)^2*(n*pi/b)^2 + ...
                        (D22/D11)*(n*pi/b)^4)));
33              end
34          end
35  mesh(X,Y,W)
36
37  %calculate the deriavatives
38  d2wdx2=diff((diff(W')')')'./dx^2;
39  d2wdy2=diff(diff(W))./dy^2;
40  d2wdxy=diff((diff(W')'))./(dx*dy);
41  d2wdx2=d2wdx2(3:end,:);
42  d2wdy2=d2wdy2(:,3:end);
43  d2wdxy=d2wdxy(2:end,2:end);
44
45  %moments
46  Mx=-(D11.*d2wdx2+D12.*d2wdy2+2*D16*d2wdxy);My=-D12.*d2wdx2-D22.*d2wdy2;
47
48  %maximums
49  W_max=max(max(W));Mx_max= max(max(Mx));My_max=max(max(My));
50
51  %Bucklings
52  a=1; b=0.5;
53  n=1;m=4; %1st. mode
54  P(1)=pi^2*((D11*(m/a)^2)+2*(D12+2*D66)*(1/b)^2+D22*(a/m)^2*(1/b)^4);
55  n=1;m=5; %2nd. mode
56  P(2)=pi^2*((D11*(m/a)^2)+2*(D12+2*D66)*(1/b)^2+D22*(a/m)^2*(1/b)^4);
57  n=1;m=3; %3th. mode
58  P(3)=pi^2*((D11*(m/a)^2)+2*(D12+2*D66)*(1/b)^2+D22*(a/m)^2*(1/b)^4);
59  P=P*10^-3 % [N/mm]
60
61  %eigen frequencies
62  rho=1.4*10^-3*10^6; %kg/m^3
63  rho_s=rho*(h*8); %kg/m^2
64  for n=1:5;for m=1:5;
65  w(m,n)=sqrt((D11*(m*pi/a)^4 +2*(D12+2*D66)*(m*pi/a)^2*(n*pi/b)^2 ...
        +D22*(n*pi/b)^4)/rho_s);
66  end;end
67  w=w/(2*pi);
68  %drawing deflection distribution
69  figure('name','[90 90 90 90]s')
70  mesh(X,Y,W)
71  xlabel('$a=1m$');ylabel('$b=0.5m$');zlabel('$Deflection[m]$')
72  set(gca, 'xlim', [0 0.5], 'ylim', [0 1])
73  saveas(gca, 'p2.eps','eps');
74
75  %drawing moment distribution
76  X=X(3:end,3:end);
```

```
Y=Y(3:end,3:end);
figure('name','[90 90 90 90]sM')
subplot(1,2,1);mesh(X,Y,Mx)
set(gca, 'xlim', [0 0.5], 'ylim', [0 1])
xlabel('$a=1m$');ylabel('$b=0.5m$');zlabel('$M_x[N]$')
subplot(1,2,2);mesh(X,Y,My)
set(gca, 'xlim', [0 0.5], 'ylim', [0 1])
xlabel('$a=1m$');ylabel('$b=0.5m$');zlabel('$M_y[N]$')
saveas(gca, 'p2m.eps','eps');
```

Layup:[0 90 0 90]s

```
%% Layup:[0 90 0 90]s
a=0.5; b=1;
n=8 %number of layers
seq=[0 90 0 90 90 0 90 0] %sequence
z=[-4 -3 -2 -1 0 1 2 3 4]*h;
%calculate the stiffness matrix
Q1=stiffness2(E1, E2, G12, nu12)

%rotate the stiffness matrices
for i = 1:n
    Q(:,:,i) = stiffness_matrix_rotater(Q1,seq(i));
end

%calculate the ABD matrix
[ ABD,A,B,D ] = ABDgenerator(Q,z,n);

%get the required elements from D matrix
D11=D(1,1);D12=D(1,2);D22=D(2,2);D66=D(3,3);D16=D(1,3);

%generate a mesh up on the rectangular shap
x=0:0.025:0.5; y=0:0.025:1; [X,Y] = meshgrid (x,y);

%calculate the max. deflection in z-axis and the bending moments
W=0;d2wdx2=0;d2wdy2=0;d2wdxy=0;
dx=0.020;dy=0.020;

        for m=1:2:100
            for n=1:2:100
                q_mn=(16.*q./(m.*n.*pi^2));
                W=W+((  q_mn*sin(m*pi*X./a).*sin(n*pi*Y./b) )  ./  ...
                    (D11*( (m*pi/a)^4 +  ...
                    (2*(D12+2*D66)/D11)*(m*pi/a)^2*(n*pi/b)^2 + ...
                    (D22/D11)*(n*pi/b)^4)));
            end
        end
mesh(X,Y,W)
```

```
34
35 %calculate the deriavatives
36 d2wdx2=diff((diff(W')')')'./dx^2;
37 d2wdy2=diff(diff(W))./dy^2;
38 d2wdxy=diff((diff(W')'))./(dx*dy);
39 d2wdx2=d2wdx2(3:end,:);
40 d2wdy2=d2wdy2(:,3:end);
41 d2wdxy=d2wdxy(2:end,2:end);
42
43 %moments
44 Mx=-(D11.*d2wdx2+D12.*d2wdy2+2*D16*d2wdxy);My=-D12.*d2wdx2-D22.*d2wdy2;
45
46 %maximums
47 W_max=max(max(W));Mx_max= max(max(Mx));My_max=max(max(My));
48
49 %Bucklings
50 a=1; b=0.5;
51 n=1;m=2; %1st. mode
52 P(1)=pi^2*((D11*(m/a)^2)+2*(D12+2*D66)*(1/b)^2+D22*(a/m)^2*(1/b)^4);
53 n=1;m=1; %2nd. mode
54 P(2)=pi^2*((D11*(m/a)^2)+2*(D12+2*D66)*(1/b)^2+D22*(a/m)^2*(1/b)^4);
55 n=1;m=3; %3th. mode
56 P(3)=pi^2*((D11*(m/a)^2)+2*(D12+2*D66)*(1/b)^2+D22*(a/m)^2*(1/b)^4);
57 P=P*10^-3 % [N/mm]
58
59 %eigen frequencies
60 rho=1.4*10^-3*10^6; %kg/m^3
61 rho_s=rho*(h*8); %kg/m^2
62 for n=1:5;for m=1:5;
63 w(m,n)=sqrt((D11*(m*pi/a)^4 +2*(D12+2*D66)*(m*pi/a)^2*(n*pi/b)^2 ...
      +D22*(n*pi/b)^4)/rho_s);
64 end;end
65 w=w/(2*pi);
66 %drawing deflection distribution
67 figure('name','[0 90 0 90]s')
68 mesh(X,Y,W)
69 xlabel('$a=1m$');ylabel('$b=0.5m$');zlabel('$Deflection[m]$')
70 set(gca, 'xlim', [0 0.5], 'ylim', [0 1])
71 saveas(gca, 'p3.eps','eps');
72
73 %drawing moment distribution
74 X=X(3:end,3:end);
75 Y=Y(3:end,3:end);
76 figure('name','[0 90 0 90]sM')
77 subplot(1,2,1);mesh(X,Y,Mx)
78 set(gca, 'xlim', [0 0.5], 'ylim', [0 1])
79 xlabel('$a=1m$');ylabel('$b=0.5m$');zlabel('$M_x[N]$')
80 subplot(1,2,2);mesh(X,Y,My)
81 set(gca, 'xlim', [0 0.5], 'ylim', [0 1])
82 xlabel('$a=1m$');ylabel('$b=0.5m$');zlabel('$M_y[N]$')
83 saveas(gca, 'p3m.eps','eps');
```

Layup:[90 0 90 0]s

```
%% Layup:[90 0 90 0]s
a=0.5; b=1;
n=8 %number of layers
seq=[90 0 90 0 0 90 0 90]  %sequence
z=[-4 -3 -2 -1 0 1 2 3 4]*h;
%calculate the stiffness matrix
Q1=stiffness2(E1, E2, G12, nu12)

%rotate the stiffness matrices
for i = 1:n
    Q(:,:,i) = stiffness_matrix_rotater(Q1,seq(i));
end

%calculate the ABD matrix
[ ABD,A,B,D ] = ABDgenerator(Q,z,n);

%get the required elements from D matrix
D11=D(1,1);D12=D(1,2);D22=D(2,2);D66=D(3,3);D16=D(1,3);

%generate a mesh up on the rectangular shap
x=0:0.025:0.5; y=0:0.025:1; [X,Y] = meshgrid (x,y);

%calculate the max. deflection in z-axis and the bending moments
W=0;d2wdx2=0;d2wdy2=0;d2wdxy=0;
dx=0.020;dy=0.020;
        for m=1:2:100
            for n=1:2:100
                q_mn=(16.*q./(m.*n.*pi^2));
                W=W+((  q_mn*sin(m*pi*X./a).*sin(n*pi*Y./b) )  ./  ...
                    (D11*( (m*pi/a)^4 +  ...
                    (2*(D12+2*D66)/D11)*(m*pi/a)^2*(n*pi/b)^2 + ...
                    (D22/D11)*(n*pi/b)^4)));
           end
        end
mesh(X,Y,W)

%calculate the deriavatives
d2wdx2=diff((diff(W')')')'./dx^2;
d2wdy2=diff(diff(W))./dy^2;
d2wdxy=diff((diff(W')'))./(dx*dy);
d2wdx2=d2wdx2(3:end,:);
d2wdy2=d2wdy2(:,3:end);
d2wdxy=d2wdxy(2:end,2:end);

```

```
%moments
Mx=-(D11.*d2wdx2+D12.*d2wdy2+2*D16*d2wdxy);My=-D12.*d2wdx2-D22.*d2wdy2;

%maximums
W_max=max(max(W));Mx_max= max(max(Mx));My_max=max(max(My));

%Bucklings
a=1; b=0.5;
n=1;m=2; %1st. mode
P(1)=pi^2*((D11*(m/a)^2)+2*(D12+2*D66)*(1/b)^2+D22*(a/m)^2*(1/b)^4);
n=1;m=3; %2nd. mode
P(2)=pi^2*((D11*(m/a)^2)+2*(D12+2*D66)*(1/b)^2+D22*(a/m)^2*(1/b)^4);
n=1;m=4; %3th. mode
P(3)=pi^2*((D11*(m/a)^2)+2*(D12+2*D66)*(1/b)^2+D22*(a/m)^2*(1/b)^4);
P=P*10^-3 % [N/mm]

%eigen frequencies
rho=1.4*10^-3*10^6; %kg/m^3
rho_s=rho*(h*8); %kg/m^2
for n=1:5;for m=1:5;
w(m,n)=sqrt((D11*(m*pi/a)^4 +2*(D12+2*D66)*(m*pi/a)^2*(n*pi/b)^2 ...
    +D22*(n*pi/b)^4)/rho_s);
end;end
w=w/(2*pi);
%drawing deflection distribution
figure('name','[90 0 90 0]s')
mesh(X,Y,W)
xlabel('$a=1m$');ylabel('$b=0.5m$');zlabel('$Deflection[m]$')
set(gca, 'xlim', [0 0.5], 'ylim', [0 1])
saveas(gca, 'p4.eps','eps');

%drawing moment distribution
X=X(3:end,3:end);
Y=Y(3:end,3:end);
figure('name','[90 0 90 0]sM')
subplot(1,2,1);mesh(X,Y,Mx)
set(gca, 'xlim', [0 0.5], 'ylim', [0 1])
xlabel('$a=1m$');ylabel('$b=0.5m$');zlabel('$M_x[N]$')
subplot(1,2,2);mesh(X,Y,My)
set(gca, 'xlim', [0 0.5], 'ylim', [0 1])
xlabel('$a=1m$');ylabel('$b=0.5m$');zlabel('$M_y[N]$')
saveas(gca, 'p4m.eps','eps');
```

Layup:[+45 -45 +45 -45]s

```
%% Layup:[+45 -45 +45 -45]s
a=0.5; b=1;
n=8 %number of layers
```

```
seq=[45 -45 45 -45 -45 45 -45 45]  %sequence
z=[-4 -3 -2 -1 0 1 2 3 4]*h;
%calculate the stiffness matrix
Q1=stiffness2(E1, E2, G12, nu12)

%rotate the stiffness matrices
for i = 1:n
    Q(:,:,i) = stiffness_matrix_rotater(Q1,seq(i));
end

% calculate the ABD matrix
[ ABD,A,B,D ] = ABDgenerator(Q,z,n);

%get the required elements from D matrix
D11=D(1,1);D12=D(1,2);D22=D(2,2);D66=D(3,3);

%generate a mesh up on the rectangular shap
x=0:0.025:.5; y=0:0.025:1; [X,Y] = meshgrid (x,y);

%calculate the max. deflection in z-axis and the bending moments
W=0;d2wdx2=0;d2wdy2=0;d2wdxy=0;
dx=0.020;dy=0.020;
        for m=1:2:100
            for n=1:2:100
                q_mn=(16.*q./(m.*n.*pi^2));
                W=W+((  q_mn*sin(m*pi*X./a).*sin(n*pi*Y./b) )  ./  ...
                    (D11*( (m*pi/a)^4 +  ...
                    (2*(D12+2*D66)/D11)*(m*pi/a)^2*(n*pi/b)^2 + ...
                    (D22/D11)*(n*pi/b)^4)));
           end
        end
mesh(X,Y,W)

%calculate the deriavatives
d2wdx2=diff((diff(W')')')'./dx^2;
d2wdy2=diff(diff(W))./dy^2;
d2wdxy=diff((diff(W')')')./(dx*dy);
d2wdx2=d2wdx2(3:end,:);
d2wdy2=d2wdy2(:,3:end);
d2wdxy=d2wdxy(2:end,2:end);

%moments
Mx=-(D11.*d2wdx2+D12.*d2wdy2+2*D16*d2wdxy);My=-D12.*d2wdx2-D22.*d2wdy2;

%maximums
W_max=max(max(W));Mx_max= max(max(Mx));My_max=max(max(My));

%Bucklings
a=1; b=0.5;
n=1;m=2; %1st. mode
P(1)=pi^2*((D11*(m/a)^2)+2*(D12+2*D66)*(1/b)^2+D22*(a/m)^2*(1/b)^4);
```

```
n=1;m=3; %2nd. mode
P(2)=pi^2*((D11*(m/a)^2)+2*(D12+2*D66)*(1/b)^2+D22*(a/m)^2*(1/b)^4);
n=1;m=4; %3th. mode
P(3)=pi^2*((D11*(m/a)^2)+2*(D12+2*D66)*(1/b)^2+D22*(a/m)^2*(1/b)^4);
P=P*10^-3 % [N/mm]

%eigen frequencies
rho=1.4*10^-3*10^6; %kg/m^3
rho_s=rho*(h*8); %kg/m^2
for n=1:5;for m=1:5;
w(m,n)=sqrt((D11*(m*pi/a)^4 +2*(D12+2*D66)*(m*pi/a)^2*(n*pi/b)^2 ...
    +D22*(n*pi/b)^4)/rho_s);
end;end
w=w/(2*pi);
%drawing deflection distribution
figure('name','[+45 -45 +45 -45]s')
mesh(X,Y,W)
xlabel('$a=1m$');ylabel('$b=0.5m$');zlabel('$Deflection[m]$')
set(gca, 'xlim', [0 0.5], 'ylim', [0 1])
saveas(gca, 'p5.eps','eps');

%drawing moment distribution
X=X(3:end,3:end);
Y=Y(3:end,3:end);
figure('name','[+45 -45 +45 -45]sM')
subplot(1,2,1);mesh(X,Y,Mx)
set(gca, 'xlim', [0 0.5], 'ylim', [0 1])
xlabel('$a=1m$');ylabel('$b=0.5m$');zlabel('$M_x[N]$')
subplot(1,2,2);mesh(X,Y,My)
set(gca, 'xlim', [0 0.5], 'ylim', [0 1])
xlabel('$a=1m$');ylabel('$b=0.5m$');zlabel('$M_y[N]$')
saveas(gca, 'p5m.eps','eps');
```

A.5.2 Example 5.1/5.2/5.7 From The Lecture Book

```
%% example 5.1
% [0 90 0 90]s
clear all
clc

addpath(genpath('C:\Users\s111058\Desktop\Dropbox\Fibre ...
    Composites\functions'))
addpath(genpath('C:\Users\Administrator\Dropbox\Fibre ...
    Composites\functions'))

E1=181e9
E2=10.3e9
```

```
G12=7.17e9
nu12=0.28
h=0.5e-3

a=1000e-3
b=1000e-3
q=1e3 %[Pa]


n=8
seq=[90 90 90 90 90 90 90 90]*0
z=[-4 -3 -2 -1 0 1 2 3 4]*h;

Q1=stiffness2(E1, E2, G12, nu12)

for i = 1:n
    Q(:,:,i) = stiffness_matrix_rotater(Q1,seq(i));
end

[ ABD,A,B,D ] = ABDgenerator(Q,z,n);

D11=D(1,1);D12=D(1,2);D22=D(2,2);D16=D(1,3);D66=D(3,3);

W=0;d2wdx2=0;d2wdy2=0;d2wdxy=0;
dx=0.012;dy=0.012;
x=0:dx:1; y=0:dy:1; [X,Y] = meshgrid (x,y);

        for m=1:2:100
            for n=1:2:100
                q_mn=(16.*q./(m.*n.*pi^2));
                W=W+((  q_mn*sin(m*pi*X./a).*sin(n*pi*Y./b) )  ./  ...
                    (D11*( (m*pi/a)^4 +  ...
                    (2*(D12+2*D66)/D11)*(m*pi/a)^2*(n*pi/b)^2 + ...
                    (D22/D11)*(n*pi/b)^4)));
           end
        end
mesh(X,Y,W)

%calculate the deriavatives
d2wdx2=diff((diff(W')')')'./dx^2;
d2wdy2=diff(diff(W))./dy^2;
d2wdxy=diff((diff(W')'))./(dx*dy);
d2wdx2=d2wdx2(3:end,:);
d2wdy2=d2wdy2(:,3:end);
d2wdxy=d2wdxy(2:end,2:end);

%moments
Mx=-(D11.*d2wdx2+D12.*d2wdy2+2*D16*d2wdxy);My=-D12.*d2wdx2-D22.*d2wdy2;

%maximums
W_max=max(max(W));Mx_max= max(max(Mx));My_max=max(max(My));
```

```
59
60 %Bucklings
61 n=1;m=1; %1st. mode
62 P(1)=pi^2*((D11*(m/a)^2)+2*(D12+2*D66)*(1/b)^2+D22*(a/m)^2*(1/b)^4);
63 n=1;m=2; %2nd. mode
64 P(2)=pi^2*((D11*(m/a)^2)+2*(D12+2*D66)*(1/b)^2+D22*(a/m)^2*(1/b)^4);
65 n=1;m=3; %3th. mode
66 P(3)=pi^2*((D11*(m/a)^2)+2*(D12+2*D66)*(1/b)^2+D22*(a/m)^2*(1/b)^4);
67 P=P*10^-3 % [N/mm]
68
69 %eigen frequencies
70 rho=1.4*10^-3*10^6; %kg/m^3
71 rho_s=rho*(h*8); %kg/m^2
72 for n=1:10;for m=1:10;
73 w(m,n)=sqrt((D11*(m*pi/a)^4 +2*(D12+2*D66)*(m*pi/a)^2*(n*pi/b)^2 ...
      +D22*(n*pi/b)^4)/rho_s);
74 end;end
75 w=w/(2*pi); %convert to [Hz] from [rad/s]
76 % drawings
77 X=X(3:end,3:end);
78 Y=Y(3:end,3:end);
79 figure('name','[+45 -45 +45 -45]sM')
80 subplot(1,2,1);mesh(X,Y,Mx)
81 set(gca, 'xlim', [0 1], 'ylim', [0 0.5])
82 xlabel('$a=1m$');ylabel('$b=1m$');zlabel('$M_x[N]$')
83 subplot(1,2,2);mesh(X,Y,My)
```

A.6 Sandwich Beam Subjected To Uniform Pressure

```
1 clear all
2 clc
3 close all
4 set(0,'defaulttextinterpreter','latex')
5
6 q=0.1*10^6 %pressure [Pa]
7 L=1 %length of the beam [m]
8 x=linspace(0,1,20); %mesh in 1-D
9 E1f=54e9; %Young modulus for 1st direction / fiber
10 M=q*L*x./2-q*x.^2/2; % moment distribution
11 Gcv=[18 22 31 40 55 73 90 108]*10^6; %shear modulus from the core
12
13 rho_f=1700 %kg/m^3
14 rho_cv=[48 60 80 100 130 160 200 250] %kg/m^3
15 tfv=0.001:0.00005:0.016; %line
16 tcv=0.001:0.00005:0.1; %column
```

```
17
18
19 a=zeros(length(tfv),length(tcv));
20 mass=zeros(length(tfv),length(tcv));
21
22 %switcher for  core material
23 k=3 %core material number
24 Gc=Gcv(k); %corresponding shear
25 rho_c=rho_cv(k); %corresponding density
26
27
28 for i = 1:length(tfv) % increase the fiber thickness // line
29 for j = 1 : length(tcv) % increase the core thickness // column
30 tf=tfv(i);tc=tcv(j);
31
32 %calculate the mass
33 volume_f=tf*L*0.1*2;
34 volume_c=tc*L*0.1;
35 mass(i,j)=volume_f*rho_f+volume_c*rho_c;
36
37 d=tf+tc;
38 D=(E1f*tf*d^2)/2;
39 S=Gc*d^2/tc;
40 w=(q*L^4/(24*D))*((x./L).^4-2*(x./L).^3+(x./L))+(q/(2*S))*(L*x-x.^2); ...
       %deflections
41 ww(i,j)=max(w); % save the max deflection
42
43 sigma=max(M)*d*E1f/(2*D);
44 ss(i,j)=sigma;
45 if max(w)<L/50 && sigma < 100e6
46     a(i,j)=3; % perfect
47 elseif max(w)<L/50
48     a(i,j)=1; % max sigma is too high
49 elseif sigma < 100e6
50     a(i,j)=2; % deflection is too high
51 else
52     a(i,j)=0; % its not working at all
53 end
54 end
55 end
56
57 imagesc(tcv*1000,tfv*1000,a)
58 colorbar
59 box on
60 grid on
61 xlabel('Core Thickness[mm]')
62 ylabel('Fiber Thickness[mm]')
63 saveas(gca, 'index_a.eps','epsc2');
64
65 figure()
66 imagesc(tcv*1000,tfv*1000,mass)
```

```
67 colorbar
68 box on
69 grid on
70 xlabel('Core Thickness[mm]')
71 ylabel('Fiber Thickness[mm]')
72 saveas(gca, 'mass.eps','epsc2');
73
74 figure()
75 imagesc(tcv*1000,tfv*1000,ww)
76 colorbar
77 box on
78 grid on
79 xlabel('Core Thickness[mm]')
80 ylabel('Fiber Thickness[mm]')
81
82 figure()
83 imagesc(tcv*1000,tfv*1000,ss)
84 colorbar
85 box on
86 grid on
87 xlabel('Core Thickness[mm]')
88 ylabel('Fiber Thickness[mm]')
```