RWTH AACHEN

LEHR- UND FORSCHUNGSGEBIET KONTINUUMSMECHANIK

Mini-thesis

Implementation of a discrete finite element based model for Non-crimp fabrics in ABAQUS

SUMEDH V JOSHI

Gutachter(in):	Univ.-Prof. Dr.-Ing Mikhail Itskov
Verfasser:	Vu Ngoc Khiêm
Matrikel-Nr.:	363064
Adresse:	Kaelberweide 10, 21147 Hamburg
Email:	sumedh.joshi@rwth-aachen.de
Telefon:	+49 176 23545766

Abgabetermin: 10. August 2017

Abstract

This mini thesis describes the implementation of a discrete finite element based model for Non-Crimp fabrics (NCF's). NCF's have become an important part of the automotive, aerospace and marine industry on account of their light-weight structure and its ability to mold into any shape. This thesis is divided into the following parts. The first part discusses the implementation of a finite element mesh for the fabric using MATLAB and ABAQUS. This method introduces a novel way of representing the fibers with the help of beam element in ABAQUS. The second part discuss the implementation of a user defined material subroutine to capture all the properties of fabric. A VUMAT code is written for the same using an Intel Fortran compiler. A hyperlastic model is used for the same purpose. A comparison of results obtained from the implementation of subroutine with those from a MATLAB script is made to check the accuracy of subroutine. In the end, conclusions are drawn by comparing the experimental results and the simulation results of the bias extension test and further comments are made regarding the same.

Acknowledgement

I would like to sincerely express my gratitude to all the people who have been a part of this thesis and have guided me continuously through this exciting journey.

First and foremost, I would like to express my sincere gratitude to Mr. Vu Ngoc Khiem, my advisor at Lehr und Forschungsgebiet kontinuumsmechanik, RWTH Aachen, for his continuous support of my work, immense knowledge, motivation and especially patience. His continuous guidance and honest remarks have not only been valuable in finishing this research work, but also have helped me become a better person. He showed me the way of doing honest, righteous research, and this has been one of the most important lesson I have learned during this journey. For all his help I would like to thank him profusely.

I would also like to thank Univ.-Prof. Dr.-Ing. Mikhail Itskov, for offering me such an interesting topic of research to work on. The vast knowledge I have gained within this few months will stay with me forever.

I would like to extend my gratefulness to all my colleagues and friends, who provided me with valuable feedback and advice. Thank you all.

Finally, I would like to thank my parents and my brother for their continuous support and encouragement throughout my studies. This incredible intellectual journey would have been impossible without either of them.

Sumedh Vasant Joshi

Master's student,
RWTH Aachen University.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 INTRODUCTION

1.1 NON-CRIMP FABRICS

Non-Crimp fabrics are being widely used in the fields of automotive and aerospace industries. This is partly due to their economic production costs and their high specific stiffness ratio. Non-crimp fabrics are also highly advantageous due to their ability of forming complex geometries (1). Automotive industries mainly use engineering fabrics for the construction of chassis and other structural parts. NCF's are mostly biaxial, triaxial and quadriaxial where the fiber tows are straight albeit with different orientations which provide multi directional properties. NCF's have a high degree of drapability due to good deformation characteristics of unidirectional plies as compared to woven fabrics that are undulated. Due to their high drapability, NCF's can be molded to form various complex shapes without giving rise to wrinkles which, are a common occurrence in woven textiles. Hence NCF's are finding wide spread applications in the fields of aerospace, automotive, yachting, wind energy and complex structural components.

During manufacturing of engineering fabrics or textiles, there are some major factors which dominate like (2),

- Tensile strength along the two fiber directions
- In-plane and out of plane flexural rigidity
- Shear resistance of the sheet

Due to this, an important challenge is to use a constitutive model and a modeling approach which can perfectly capture all these properties. Thus, an important part of modeling NCF's is to extract their real material properties by using various modeling approaches. Generally, more accurate the mathematical model, more computationally expensive the solution. Hence one must always make a compromise between the model and computation cost.

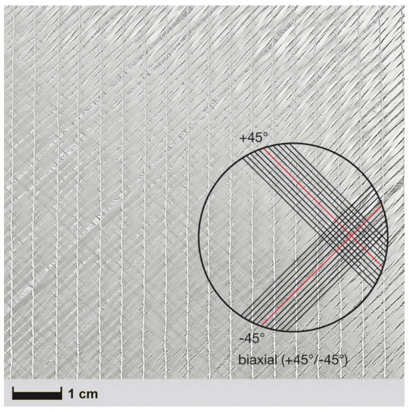


Abbildung 1.1: Biaxial non-crimp fabric

1.2 AIM and OBJECTIVES

This mini thesis consists of three parts:

- The first part focuses on the preparation of a discrete finite element based mesh for the non-crimp fabric.
- Second part consists of implementing an averaging based material model (user subroutine), which can accurately model the material properties of the fabric.
- Last part consists of running simulations for in-plane shear deformations.

2 FINITE ELEMENT MODEL

2.1 Modeling of NCF's on ABAQUS

Numerous modeling approaches, especially those which successfully model in-plane bending stiffness but neglect out of plane bending stiffness have been proposed to model engineering fabrics. The most common one is to use truss elements to represent fibers and membrane elements to capture the in-plane shear resistance. Hence, to overcome this drawback, one can use beam elements to represent the fibers (2). The introduction of beam elements allows one to easily model out of plane and in-plane bending stiffness. This approach is discussed briefly in this section.

In this thesis, beam elements are used to represent fibers. Thus, one set of Timoshenko beam elements (B31) lies at a very small distance above the membrane elements (M3D4R) and the other set of beam elements lies at a small distance below the membrane element. The corner nodes of the beam element are connected to those of membrane elements with the help of hinge connections. The hinge connections restrict relative motion of the two elements but allow for their free rotation.

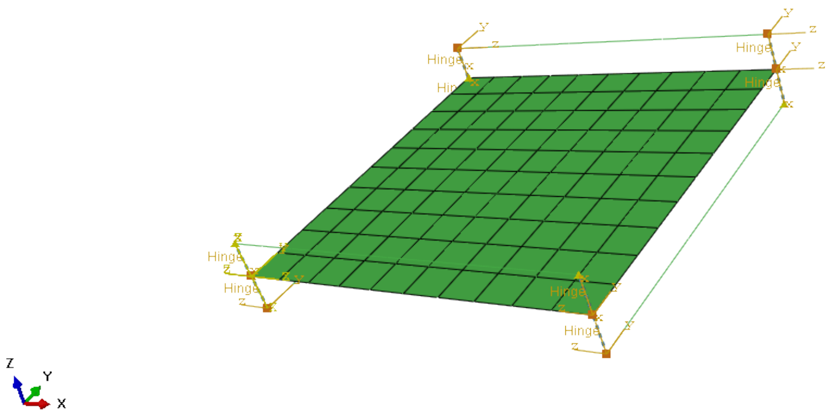


Abbildung 2.1: Unit cell of fabric

2.2 Discrete finite element mesh for NCF's

The meso-scale modeling approach is usually not preferred. It has high computational costs and also a unit cell cannot properly represent the material properties of the fabric (1). Hence a discrete finite element based approach is used which is discussed in this section.

As discussed in the previous section beam elements were used to model fibers. In this section, we discuss very briefly the modeling of a double layered NCF with the two fiber bundles aligned at 45 degrees with respect to one another. Two layers of fibres results in more strength of the non-crimp fabric. Tests were conducted on this specimen of Non-Crimp fabric using ABAQUS 6.12.1 software.

This mesh was generated by writing a code on MATLAB. This MATLAB generated input file was later used to run the simulation on ABAQUS. The alignment of fibers or the fiber direction directly affects the in-plane shear response of the non-crimp fabric. The length to width ratio also plays a part in the bending and shear response.

While running the simulations for the non-crimp fabric, the discrete elements of the fabric were modeled as shell elements. The elements can also be modeled as membrane elements. The change in the angle between the two fibre directions can also be measured.

The MATLAB code used generates the node numbers and the element numbers for the composite. This is very efficient technique as composites with very high number of discrete elements can be easily modeled. By including different test cases in the code composites with different fiber orientations can be achieved. The code generates a discrete element mesh which models the fibers as beam elements. The code has been attached in appendix for reference.

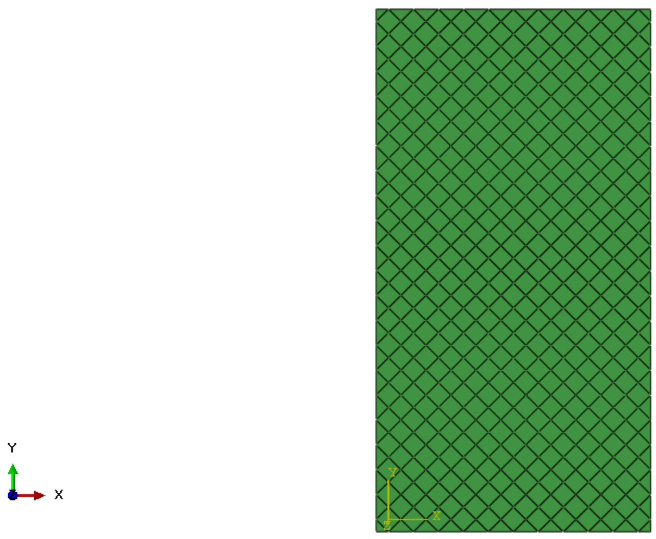


Abbildung 2.2: Finite element mesh

As shown above the fibers are arranged in a crisscross fashion to better model the behavior of the fabrics. Tests are conducted on this specimen by using a user defined material subroutine in FORTRAN and compiling it on ABAQUS.

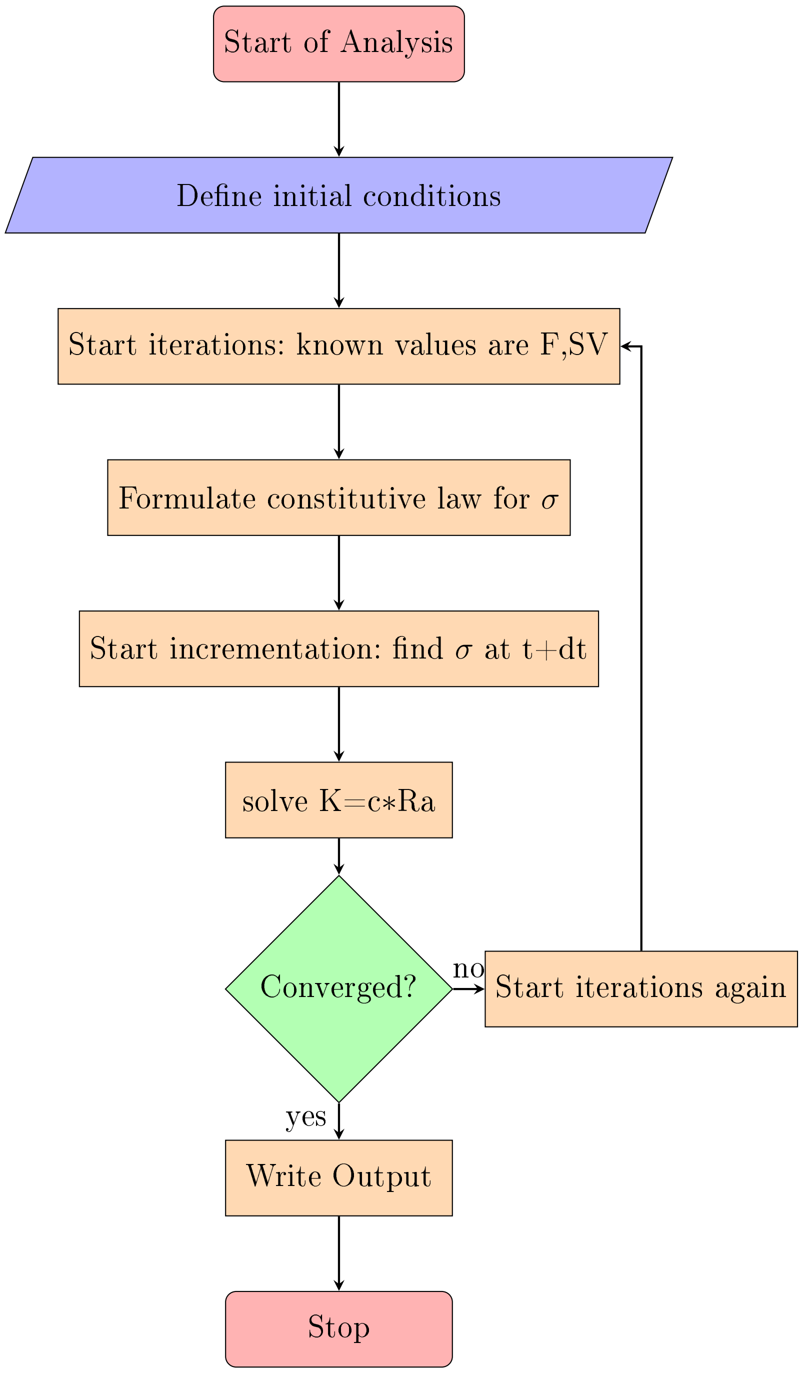
3 USER SUBROUTINE

3.1 Basics of ABAQUS solver

Commercial finite element solvers like ABAQUS are currently being used on a large scale in many engineering domains. One important drawback of these solvers is that they do not usually provide all material properties and elements that the user desires. Hence to overcome this drawback one can construct one's own material constitutive law by programming a user defined subroutine on Fortran.

The flowchart provided on the next page can help one to understand the basic functioning of an ABAQUS solve using user subroutines. Depending on the material to be modeled and the elements of the model there are a large number of subroutines available. In this mini-thesis the subroutine VUMAT ha been use to model the hyperelastic behaviour of the NCF's.

The corresponding section discusses the basics of VUMAT which is then followed by a detailed description of the material model used in this thesis.


Start of Analysis
Define initial conditions
Start iterations: known values are F,SV
Formulate constitutive law for σ
Start incrementation: find σ at t+dt
solve K=c*Ra
Converged?
no
Start iterations again
yes
Write Output
Stop

Basic flowchart for ABAQUS solver

3.2 ABAQUS Explicit solver

All commercial FEM solvers are based on either the Implicit method or the Explicit method for solving the equations of motion. Implicit solvers are usually used for static analysis while dynamic simulations use explicit solvers.

Implicit solvers are based on fully implicit 'backward difference method' applied for the time integration of the governing equations. (5)

The explicit solvers use explicit method for both time and spatial discretization. The dynamic equilibrium equations are written as,

$$\boldsymbol{Ma} = \boldsymbol{P} - \boldsymbol{I} \tag{3.2.1}$$

where $\boldsymbol{P}$ and $\boldsymbol{I}$ are the External forces and the inertial forces respectively. If the first term is small enough the equations reduce to static equations. The explicit algorithm is also known as the central difference algorithm. The lumped mass matrix allows the program to calculate the nodal accelerations easily at any given time.(5)

$$\boldsymbol{a} = \boldsymbol{M}^{-1}(\boldsymbol{P} - \boldsymbol{I}) \tag{3.2.2}$$

Neither iteration nor convergence checking is required. The advantages of ABAQUS Explicit are;

- It does not add additional degrees of freedom to the model.
- It can simulate quasi-static problem.

The stress at every point is calculated by marching through time using explicit, forward finite difference method.

$$\boldsymbol{\sigma_{t+\Delta t}} = \boldsymbol{\sigma_t} + \boldsymbol{\Delta\sigma} \tag{3.2.3}$$

The displacements are found in a similar manner by integrating the acceleration equations. For example, the displacements are calculated from the velocities as follows,

$$\boldsymbol{u}^{t+\Delta t} - \boldsymbol{u}^{t} = \boldsymbol{\Delta t u_t} \tag{3.2.4}$$

3.3 Stability requirements for EXPLICIT solvers

The explicit dynamics procedure solves every problem as a wave propagation problem. A wave propogation problem is typically represented by a hyperbolic partial differential equation. A necessary condition for the stability of these PDE's is the Courant-Friedrich-Lewy or CFL condition.The CFL condition dictates the maximum possible time step which can be used to obtain a stable solution. A stable solution is only obtained when the time step is less than the stable time step.

The CFL condition imposes a limiting condition on the time step based on the spatial time step and the largest eigenvalue of the system. The CFL condition states that,'The numerical domain of influence should be larger than the physical domain of influence'. The physical domain of influence is defined by the characteristic curves of the solution, while the numerical domain is decided by the spatial and temporal time steps. Mathematically, the CFL condition is given as,

$$\frac{\boldsymbol{\Delta t}}{\boldsymbol{\Delta x}} \leq \frac{\mathbf{d}\boldsymbol{t}}{\mathbf{d}\boldsymbol{x}} \tag{3.3.1}$$

In general the CFL condition is given as,

$$\boldsymbol{\lambda}\frac{\boldsymbol{\Delta t}}{\boldsymbol{\Delta x}} \leq \mathbf{1} \tag{3.3.2}$$

where, $\boldsymbol{\lambda}$ indicates the largest eigenvalue of the system.

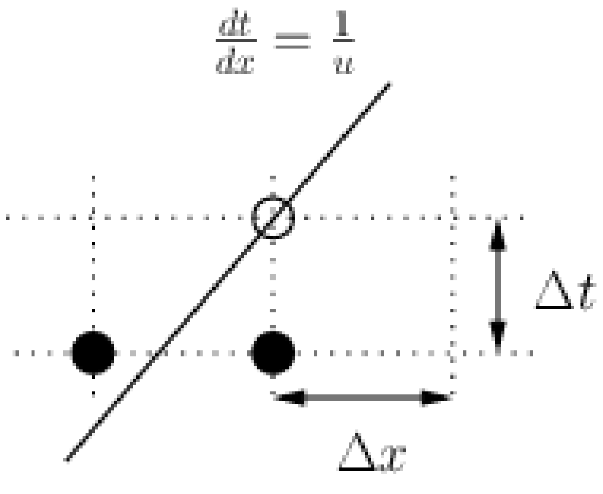


Abbildung 3.1: Stable CFL condition

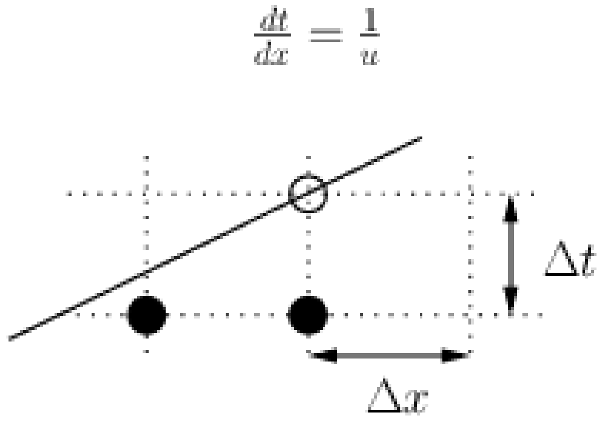


Abbildung 3.2: Unstable CFL condition

As stated before the explicit solver solves every problem as a wave propagation problem. In solid mechanics problems, the largest eigenvalue is represented by $\boldsymbol{\omega_{max}}$. This is the largest frequency of the propagating pressure wave or stress wave. Hence the stable time increment is the minimum time that a dilatational wave takes to move across any discrete element in the body.(8) To avoid extracting eigenvalues, a more practical estimate of the stability limit is made using dilatational wave speed c_d and the characteristic length L_e of the smallest element domain. This is given by the following formula,

$$\boldsymbol{\Delta t = min \frac{L_e}{c_d}} \tag{3.3.3}$$

The damping of stress shock waves, (shock waves are a common phenomena in non-linear hyperbolic PDE's) can be achieved by increasing the density.(8) This is called as 'mass scaling'. The two figures below describe a stable and unstable CFL condition.

3.4 Basics of VUMAT subroutine

The VUMAT subroutine is used to define mechanical constitutive behavior of a materi-al.VUMAT is called on blocks of material points for which the constitutive law has been stated.

The VUMAT subroutine need the Cauchy stress tensor $\boldsymbol{\sigma}$ to be in corrotational form. This form is called as objective stress rate. In a corrotational frame the basis system rotates with the material. All stress and strain tensor quantities are defined relative to this frame of reference.(7)

Cauchy stress rate $\boldsymbol{\sigma}$ is used to define stress in ABAQUS. This stress is based on the deformed coordinate system. This stress can be depicted as;

$$\sigma = \sigma_{ij} e_i \otimes e_j \tag{3.4.1}$$

where, $e_i (i = 1, 2, 3)$ represents a rectangular coordinate system which rotates with the deformed material. As stress generally depends on history it is necessary to build a constitutive equation in incremental form. Thus differentiating above equation with respect to time yields;

$$\dot{\sigma} = \dot{\sigma}_{ij}\, e_i \otimes e_j + \sigma_{ij}\, \dot{e}_i \otimes e_j + \sigma_{ij}\, e_i \otimes \dot{e}_j \tag{3.4.2}$$

The second and third terms in the above equation account for change due to the motion of coordinate system. This stress rate takes a special form, called the Green-Naghadi strain rate for ABAQUS explicit. This stress rate is modeled when wrriting a VUMAT. It is only stated below, without giving ny derivation (7);

$$\dot{\boldsymbol{\sigma}} = \mathring{\boldsymbol{\sigma}} + \boldsymbol{\Omega\sigma} + \boldsymbol{\sigma\Omega^T} \tag{3.4.3}$$

If a stress rate other than the Green-Naghdi rate is used, then there is a need to rotate the tensor state variables. This is given by;

$$\boldsymbol{\sigma_{t+\Delta t}} = \boldsymbol{\Delta R} \cdot \boldsymbol{\sigma_t} \cdot \boldsymbol{\Delta R^T} + \boldsymbol{\Delta \sigma} \tag{3.4.4}$$

Hyperelastic constitutive models in VUMAT should be defined in a corotational coordinate system in which the basis system rotates with the material. This is most effectively accomplished by formulating the hyperelastic constitutive model in terms of the stretch tensor $\boldsymbol{U}$, instead of in terms of the deformation gradient, $\boldsymbol{F}$. The polar decomposition of deformation gradient is given by;

$$\boldsymbol{F} = \boldsymbol{R} \cdot \boldsymbol{U} \tag{3.4.5}$$

Hence, as the deformation gradient already contains the rotational tensor, there is no need to again multiply the stress tensors by rotation tensors. This exact approach has been used in this thesis.

3.5 Strain Energy function for Non-Crimp Fabric

In this thesis a hyper-elastic material model has been used to model the material behavior. The corresponding strain energy function for this material is given by;

$$\Psi(\mathbf{C}) = \kappa_1 f(\bar{I}_a) + \mu_1 g(\bar{J}_a) \tag{3.5.1}$$

where κ_1 and μ_1 are material constants. Furthermore,

$$\bar{I}_a = tr(\boldsymbol{C L_a}) \quad and \quad \bar{J}_a = tr(\boldsymbol{C^{-1} L_a}) \tag{3.5.2}$$

therein $\boldsymbol{C}$ is the right Cauchy Green tensor.

$$\boldsymbol{L_a} = w_a \boldsymbol{L_1} + \frac{(1-w_a)}{3}\boldsymbol{I} \tag{3.5.3}$$

where

$$\boldsymbol{L_1} = \boldsymbol{M_1} \otimes \boldsymbol{M_1} \tag{3.5.4}$$

Here $\boldsymbol{M_1}$ is unit vector in the fiber direction in the undeformed configuration. w_a is a material constant.

3.6 Second Piola-Kirchoff stress tensor

The second Piola-Kirchoff stress tensor can be derived from the strain energy function by using the following identity (9);

$$\mathbf{S} = 2\frac{\partial \Psi}{\partial \mathbf{C}} \tag{3.6.1}$$

where $\boldsymbol{S}$ is the first Piola-Kirchoff stress tensor. The differentiation can be done by using the laws of tensor algebra. This is shown below. We will consider the two terms of equation 3.5.1

The two terms need to be differentiated here are;

$$tr(\boldsymbol{C}\boldsymbol{L_a}) = w_a tr(\boldsymbol{C}\boldsymbol{L_1}) + \frac{1-wa}{3} tr(\boldsymbol{C}) \tag{3.6.2}$$

$$tr(\boldsymbol{C^{-1}}\boldsymbol{L_a}) = w_a tr(\boldsymbol{C^{-1}}\boldsymbol{L_1}) + \frac{1-wa}{3} tr(\boldsymbol{C^{-1}}) \tag{3.6.3}$$

Using the tensor rules of differentiation as stated below (9)

$$tr(\mathbf{CL})_{,\mathbf{C}} = \mathbf{L} \quad and \quad \mathbf{C}^{-1}{}_{,\mathbf{C}} = -\mathbf{C}^{-1} \otimes \mathbf{C}^{-1} \tag{3.6.4}$$

and tensor identities;

$$\mathbf{Y} : \mathbf{A} \otimes \mathbf{B} = \mathbf{A}^T \mathbf{Y} \mathbf{B}^T \tag{3.6.5}$$

Knowing that $\mathbf{C}$ is a symmetric tensor, we get the following expression for $\mathbf{S}$;

$$\boldsymbol{S} = \kappa_1 w_a \boldsymbol{L_1} + \kappa_1 \frac{1-w_a}{3}\boldsymbol{I} - \mu_1 w_a \boldsymbol{C^{-1}}\boldsymbol{L_1}\boldsymbol{C^{-1}} + \mu_1 \frac{w_a - 1}{3}\boldsymbol{C^{-1}}\boldsymbol{I}\boldsymbol{C^{-1}} \tag{3.6.6}$$

3.7 Corotational Cauchy stress tensor

In VUMAT user subroutine we need the corrotational Cauchy stress tensor. The relation between the second Piola-Kirchoff stress tensor and Cauchy stress tensor is given by (9);

$$\sigma = J^{-1}\boldsymbol{F}\boldsymbol{S}\boldsymbol{F}^{\boldsymbol{T}} \tag{3.7.1}$$

Here we assume the material is incompressible and hence J=1;
Using the polar decomposition of the deformation gradient as given by equation

$$\sigma = J^{-1}\boldsymbol{R}\boldsymbol{U}\boldsymbol{S}\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{R}^{\boldsymbol{T}} \tag{3.7.2}$$

Thus, substituting the expression for $\boldsymbol{S}$ in the equation above,

$$\sigma = J^{-1}\boldsymbol{R}\boldsymbol{U}(\kappa_1 w_a \boldsymbol{L_1} + \kappa_1\frac{1-w_a}{3}\boldsymbol{I} - \mu_1 w_a \boldsymbol{C^{-1}}\boldsymbol{L_1}\boldsymbol{C^{-1}} + \mu_1\frac{w_a-1}{3}\boldsymbol{C^{-1}}\boldsymbol{I}\boldsymbol{C^{-1}})\boldsymbol{U^T}\boldsymbol{R^T} \tag{3.7.3}$$

The corrotational tensor is given by;

$$\boldsymbol{\sigma_{corot}} = \boldsymbol{R^T}\boldsymbol{\sigma}\boldsymbol{R} \tag{3.7.4}$$

Hence from equation (3.7.4) we get,

$$\sigma_{\boldsymbol{corot}} = \kappa_1 w_a \boldsymbol{U}\boldsymbol{L_1}\boldsymbol{U^T} + \kappa_1\frac{1-w_a}{3}\boldsymbol{U^2} - \mu_1 w_a \boldsymbol{U^{-1}}\boldsymbol{L_1}\boldsymbol{U^{-1}} + \mu_1\frac{w_a-1}{3}\boldsymbol{U^{-1}}\boldsymbol{I}\boldsymbol{U^{-1}} \tag{3.7.5}$$

In this mini thesis we assume that the material is incompressible Hence the constraint for incompressibility needs to be applied. This condition is given by J = 1 (9). Hence The strain energy function is written as;

$$\Psi(\boldsymbol{C}) = \Psi(\boldsymbol{C}) - p(J-1) \tag{3.7.6}$$

where the hydrostatic pressure p acts a Lagrange multiplier to enforce the incompressibility condition. Accounting for this term gives us the final expression for the corrotational Cauchy stress tensor as;

$$\sigma_{\boldsymbol{corot}} = \kappa_1 w_a \boldsymbol{U}\boldsymbol{L_1}\boldsymbol{U^T} + \kappa_1 \frac{1-w_a}{3}\boldsymbol{U^2} - \mu_1 w_a \boldsymbol{U^{-1}}\boldsymbol{L_1}\boldsymbol{U^{-1}} + \mu_1 \frac{w_a - 1}{3}\boldsymbol{U^{-1}}\boldsymbol{I}\boldsymbol{U^{-1}} - p\boldsymbol{I} \tag{3.7.7}$$

The above equation is used while writing a VUMAT subroutine. The above equation describes the average based material model for the non-crimp fabrics. Note that the co-rotational stress tensor is expresses in terms of the principal stretch tensor $\boldsymbol{U}$. This is an important characteristic of VUMAT subroutine.

4 CONSISTENCY CONDITIONS FOR INCOMPRESSIBLE HYPERELASTIC MATERIAL

4.1 Introduction

Every user material constitutive law follows the linear elastic law at the start of the simulation (at t = 0). Hence the material law should be linearized for the first iteration. This process of linearizing a given material law is called as linearization. Consistency with linear elastic materials is sometimes required for hyperelastic materials, in order to determine certain material constants. These usually include the Lame constants. These conditions can be found out by comparing Hooke's law with linearized hyperelasticity at small strains.

The stress strain equations for linear elastic material is given by,

$$\boldsymbol{\sigma = \lambda tr(\epsilon) + 2\mu\epsilon} \tag{4.1.1}$$

The following form should be obtained for stress-strain relation of hyperelastic material. Note that, for incompressible materials, this law is a different and is stated later on. This process is called linearization (10).

4.2 Derivation

The first step in the derivation is finding the fourth order tangent modulus tensor. The tangent modulus tensor for any constitutive equation is given by;

$$\boldsymbol{\mathcal{C}} = 2\frac{\partial \mathbf{S}}{\partial \mathbf{C}} \tag{4.2.1}$$

For linearization, the right cauchy green tensor reduces to the identity tensor. That is, $\mathbf{C} = \mathbf{I}$. Hence at time t=0, the material follows linear elastic law.Thus making the appropriate substitutions, for the given material law we get the following expression for tangent modulus (at the initial time step);

$$\boldsymbol{\mathcal{C}} = 2w_a\mu_1(\mathbf{I}\otimes\boldsymbol{L_1}+\boldsymbol{L_1}\otimes\mathbf{I}) - \frac{4}{3}\mu_1(w_a-1)\mathbf{I}\otimes\mathbf{I} \tag{4.2.2}$$

In order to find the cauchy stress tensor, we map the tangent modulus with the infinitesimal strain tensor. We use the following identity;

$$\boldsymbol{\sigma} = \boldsymbol{\mathcal{C}} : \boldsymbol{\epsilon} \tag{4.2.3}$$

while deriving this we use the following two identities (10);

$$\boldsymbol{tr(\epsilon) = 0} \tag{4.2.4}$$

$$\boldsymbol{\mathcal{I} : \epsilon = tr(\epsilon) I} \tag{4.2.5}$$

where, $\mathcal{I}$ is the fourth order super symmetric identity tensor also called as the projection tensor. The non super-symmetric identity tensor when mapped with the cauchy strain tensor yields the following result.

$$\mathtt{I} : \boldsymbol{\epsilon} = \boldsymbol{\epsilon} \tag{4.2.6}$$

Note that the non super -symmetric identity tensor is given by (9);

$$\mathbf{I}\otimes\mathbf{I} = \mathtt{I} \tag{4.2.7}$$

Hence on mapping the tangent modulus with the cauchy strain tensor, yields the following expression for the cauchy stress tensor.

$$\boldsymbol{\sigma} = \boldsymbol{\alpha_1}(\boldsymbol{\epsilon L_1} + \boldsymbol{L_1 \epsilon}) + \boldsymbol{\alpha_2 \epsilon} - \boldsymbol{pI} \tag{4.2.8}$$

where the constants α_1 and α_2 are given by;

$\alpha_1 = 2w_a\mu_1$ and $\alpha_2 = \frac{4}{3}\mu_1(w_a - 1)$

Note that for this derivation, the following identity has also been used (9):

$$\mathbf{A} \otimes \mathbf{B} : \mathbf{X} = \mathbf{AXB} \tag{4.2.9}$$

The corresponding equation for the linear elastic incompressible material is given by;

$$\boldsymbol{\sigma} = -\boldsymbol{pI} + \boldsymbol{\mu\epsilon} \tag{4.2.10}$$

Comparing the above equation with equation (4.2.8) we get the following value for the lame parameter;

$$\mu = (2\alpha_1 L_{ij} + \alpha_2) \tag{4.2.11}$$

Hence in the user subroutine the above derived equation for cauchy stress tensor is used at time t=0. This process is called linearization.

5 RESULTS

In this section the results of user subroutine and various simulations are discussed. The results are divided into tw0 sections;

- First we discuss the accuracy of the user subroutine. This is achieved by comparing the results for stress obtained by user subroutine to those by a symbolic MATLAB code.
- In the second section the results of the user subroutine VUMAT on the non-crimp fabrics is discussed. The results are discussed later.

Before discussing the results, a very important point concerning the incompressibility of material needs to be addressed. This assumption is very important in determining the out of plane component of the stretch tensor.

5.1 Assumption of Incompressibility

For the sake of computational efficiency, NCF's can be assumed to be an incompressible material (1). For shell elements, the out of plane stress $\sigma_{33} = 0$. But the value of the stretch tensor component is actually unknown and can be calculated under the assumption of incompressibility. This is explained in the following section.

The assumption of incompressibility provides an easy way to calculate $U_{3,3}$. The constraint of incompressibility is given by $det(U) = 1$. As we are dealing with the condition of plane stress, the following assumptions can be made regarding the various components of the stretch tensor.

$$U(1,3) = U(3,1) = U(2,3) = U(3,2) = 0 \tag{5.1.1}$$

Thus we have the following format for the stretch tensor;

$$U_{ij} = \begin{pmatrix} U_{11} & U_{12} & 0 \\ U_{21} & U_{22} & 0 \\ 0 & 0 & U_{33} \end{pmatrix}$$

Applying the constraint of incompressibility we get the following equation for the value of $U_{3,3}$;

$$det(U) = 1 = U_{3,3} * (U_{1,1} * U_{2,2} - U_{1,2}^2) \tag{5.1.2}$$

$$U_{3,3} = \frac{1}{(U_{1,1} * U_{2,2} - U_{1,2}^2)} \tag{5.1.3}$$

This is the condition which has been used in this mini thesis to find the value of U_{33}. By using this condition, one can the find out the value of the hydrostatic pressure, 'p'. This is explained in the next section. A point worth noting here is that there is also another way of finding the value of $U_{3.3}$. This is by assuming a predefined value of 'p' for near incompressibility and then using this value to find that of $U_{3,3}$. This approach has not been used in this thesis and hence is not elaborated further.

5.2 Hydrostatic Pressure term

Another important point worth noting is the definition of the hydrostatic pressure. For incompressible material we need the correct definition of hydrostatic pressure. This is obtained by applying the correct boundary conditions. The boundary condition used in this thesis is that the stress in the normal direction is zero, i.e $\sigma_{3,3} = 0$. We will use the value of $U_{3,3}$, calculated in the previous section to define the value of 'p'

The derivation is carried out by equating the stress in the normal direction to be zero. Hence equation (3.7.7) in component form reads;

$$\begin{aligned} \sigma_{\boldsymbol{ijcorot}} = \kappa_1 w_a(\boldsymbol{ULU^T})_{ij} + \kappa_1 \frac{1-w_a}{3}\boldsymbol{U^2_{ij}} - \mu_1 w_a(\boldsymbol{U^{-1}LU^{-1}})_{ij} \\ +\mu_1 \frac{w_a - 1}{3}(\boldsymbol{U^{-2}})_{ij} - p\mathbf{I} \end{aligned} \tag{5.2.1}$$

For plane stress condition, the terms containing the structural tensors do not contribute anything to the normal stress component and are zero. Hence equating the remaining terms of the Right hand side of equation(5.2.1) to zero ($\sigma_{33} = 0$) and substituting

$i = j = 3$; we get;

$$0 = \kappa_1 \frac{1 - w_a}{3} U_{33}^2 + \mu_1 \frac{w_a - 1}{3} (U^{-2})_{33} - p \tag{5.2.2}$$

Hence we get the following expression for the hydrostatic pressure 'p';

$$p = \kappa_1 \frac{1 - w_a}{3} (U_{33}^2) + \mu_1 \frac{w_a - 1}{3} (U^{-2})_{33} \tag{5.2.3}$$

Substituting this value of p, in the corrotational cauchy stress tensor equation ensures that the stress in the normal direction vanishes for each iteration.

5.3 MATLAB and FORTRAN results comparison

An important task in material modeling is to check the validity of the user subroutine. That is, we need to check whether the derived expression for the Cauchy stress is correct. This can be done by comparing the results with a Matlab code for a given deformation gradient $\boldsymbol{F}$. The next paragraph briefly explains the methodology of MATLAB code.

For the codes the following user inputs are given, the fiber orientation angle, the material constants κ, μ and w_a and a certain value of the deformation gradient.

An important distinction needs to be made here. While differentiating symmetric tensors we have to deal with the problem of non-uniqueness (9). Indeed in this case, $\boldsymbol{M_{ij}} = \boldsymbol{M_{ji}}$ for $(i \neq j = 1, 2.....n)$ so that only $n(n+1)/2$ among all n^2 components of the tensor argument $\boldsymbol{M} \; \epsilon \; \boldsymbol{Sym}^n$ are independent. Hence in these cases the directional derivative does nnot yield a unique result. This problem is solved by symmetrizing the tensor. thus, the derivative is only taken with respect to the independent components

Writing the strain energy in the component form;

$$\Psi((Csym)_{ij}) = \kappa_1 tr((Csym)_{ij} L_{aij}) + \mu_1 tr((Csym)_{ij}^{-1} L_{aij}) - p(det(Csym) - 1) \tag{5.3.1}$$

where;

$$(Csym)ij = \frac{1}{2}(C_{ij} + C_{ij}^T) \tag{5.3.2}$$

We differentiate this equation to obtain the second Piola - Kirchoff stress tensor.Hence we denote the second Piola - Kirchoff stress tensor in component form as;

$$S_{ij} = 2\frac{\partial \Psi}{\partial C_{ij}} \tag{5.3.3}$$

Hence we have accounted for non-uniqueness by using above expressions. next we discuss a little bit about the vectors in the fiber direction. These vector is given by;

$$\boldsymbol{M_1} = \cos\alpha \boldsymbol{e_1} + \sin\alpha \boldsymbol{e_2} \tag{5.3.4}$$

Hence the structural tensor takes the following forms;

$$L_{1ij} = \begin{pmatrix} \cos^2\alpha & \cos\alpha\sin\alpha & 0 \\ \cos\alpha\sin\alpha & \sin^2\alpha & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

The basis for above mentioned tensors is the orthogonal one which rotates with the deformed material.

The corrotional Cauchy stress tensor is then obtained from the second-Piola Kirchoff stress tensor. This is done by using the formulae discussed in the previous sections.Thus, the values of second-Piola Kirchoff stress tensors after running codes on
MATLAB and FORTRAN were obtained. These are compared in the result table shown below.

A point worth noting here is that, the results above are for the plane stress case. Hence the value of S_{13} and S_{23} is zero.

Stress component	MATLAB result	Fortran result	Error
S_{11}	-2001.162982794145	-2001.16296552493	1.7265e-5
S_{22}	-776.679157424189	-776.679150679183	6.820e-6
S_{33}	0.00000	8.993e-7	8.993e-7
S_{12}	1248.402696674738	1248.40268584637	1.082e-5
S_{13}	0.00000000000000	0.00000000000000	0.0000
S_{23}	0.00000000000000	0.00000000000000	0.0000

Tabelle 5.1: Comparison of results

5.4 Simulation results of Bias extension test

The bias extension test is a simple test to determine in-plane shear properties of fabric composites. The standard analysis of this test is based on two assumptions; the inextensibility of fibers and rotation at the yarn crossovers without slippage. Thus this test is used to investigate the shear stiffness of fabrics. The bias extension test can also be used to study how the fibers reorient themselves after the tensile force is applied.

The fibers are initially aligned at an angle of 45 degrees to the direction of tensile load. The angle between two directions of fibers is 90 degrees. The specimen used had the dimensions of W=100mm and H=50mm. The next section briefly describes the boundary conditions of the bias extension test.

The figure below shows the strip of material used to conduct the bias extension test.

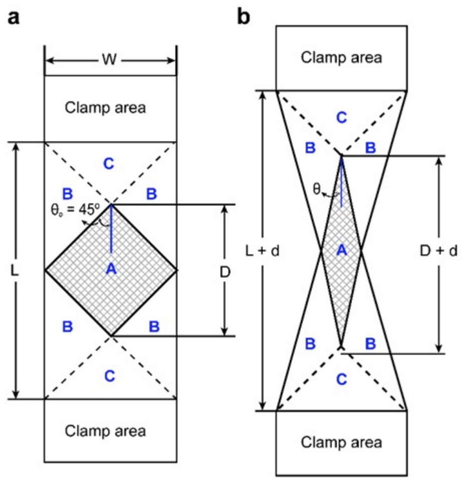


Abbildung 5.1: Bias extension test set-up

The specimen is divided into three main regions. Region C (below part) is completely fixed (encastre boundary condition is applied). To the region C (upper part) a displacement in the y direction is applied while all other displacements and rotations are set to zero. The middle region denoted by A is in a state of pure shear. These three regions exhibit distinct behavior as shown in the results of simulation.The bias extension test can be carried out both experimentally and by running numerical simulations. Here only the numerical approach is discussed.

The experiment was run on the commercial finite element solver ABAQUS.The user material subroutine VUMAT was used for the same. Numerical results of the shear angles in bias extensions obtained by FE-simulations is illustrated next.

5.5 Simulation results

The results are carried out for two different loading conditions. The displacement in y direction is 25mm and 50mm respectively. As can be seen from the results, the fibers reorient themselves as shown. As the displacement in the y direction increases, the fibers are deformed further. This experiment thus demonstrates the in plane shear properties of the fibers. The above shown result describes the in-plane shear component of Cauchy stress. This result is not exactly accurate. This indicates that the subroutine can be improved and the material constants should be checked. It figuratively matches the results obtained by the experimental bias extension test as shown above. The following sections discuss the results in an elaborate manner.

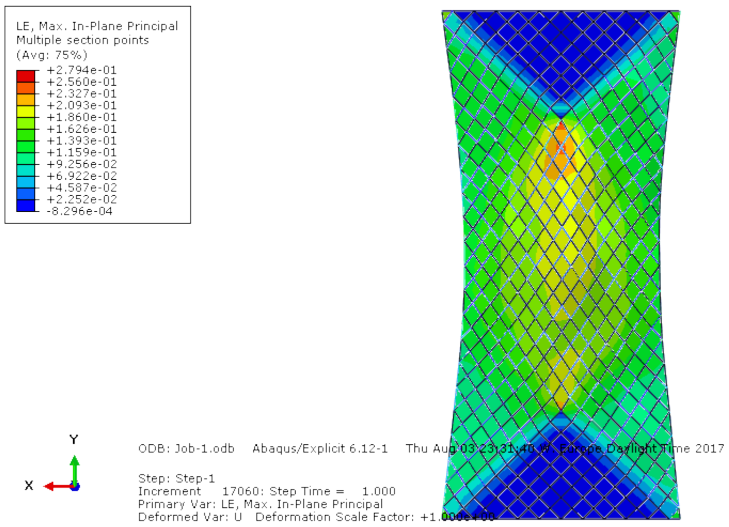

LE, Max. In-Plane Principal
Multiple section points
(Avg: 75%)
+2.794e-01
+2.560e-01
+2.327e-01
+2.093e-01
+1.860e-01
+1.626e-01
+1.393e-01
+1.159e-01
+9.256e-02
+6.922e-02
+4.587e-02
+2.252e-02
-8.296e-04
Y
X
ODB: Job-1.odb   Abaqus/Explicit 6.12-1   Thu Aug 03 23:31:40 W. Europe Daylight Time 2017
Step: Step-1
Increment   17060: Step Time =   1.000
Primary Var: LE, Max. In-Plane Principal
Deformed Var: U   Deformation Scale Factor: +1.000e+00

Abbildung 5.2: Bias extension test d = 25mm

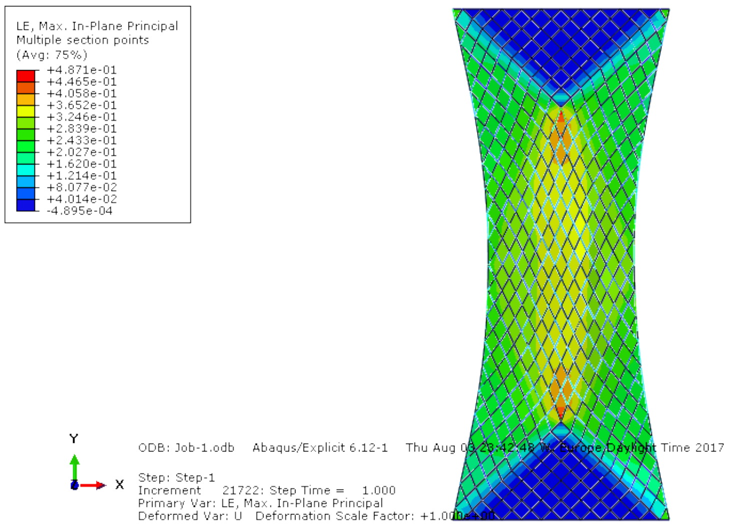

LE, Max. In-Plane Principal
Multiple section points
(Avg: 75%)
+4.871e-01
+4.465e-01
+4.058e-01
+3.652e-01
+3.246e-01
+2.839e-01
+2.433e-01
+2.027e-01
+1.620e-01
+1.214e-01
+8.077e-02
+4.014e-02
-4.895e-04
Y
X
ODB: Job-1.odb   Abaqus/Explicit 6.12-1   Thu Aug 03 23:42:48 W. Europe Daylight Time 2017
Step: Step-1
Increment   21722: Step Time =   1.000
Primary Var: LE, Max. In-Plane Principal
Deformed Var: U   Deformation Scale Factor: +1.000e+00

Abbildung 5.3: Bias extension test d = 50mm

6 CONCLUSION AND FUTURE WORK

This section briefly describes the goals achieved from this thesis. Comments are made regarding modeling of shell elements, verification method of user subroutines and future scope of this thesis.

6.1 Conclusion

In this mini thesis a discrete finite element based model has been developed to better study the in plane shear characteristics of non-crimp fabrics. A simple hyperelastic constitutive model was implemented using ABAQUS sub routine for the same. A study was carried out to study the reorientation of fibers with changing loading conditions. The results were found to describe those obtained in the bias extension test.

The accuracy of user subroutine was validated by comparing it with a symbolic Matlab code. The results of both the codes matched each other to a very high level of accuracy.

An approach for accurately modeling shell elements using user subroutines was discussed. The effect of assumptions of incompressibility and consequently the nature of hydrostatic pressure term was explained in detail. These concepts were then successfully implemented in the user subroutine.

6.2 Future work

The next part of this work will include carrying out actual forming simulations on the fabric using the user subroutine. Composite fabrics are extensively tested by draping experiments in order to study their drapability and flexibility. The user subroutine can then be used to test the forming characteristics of the composite material.

Literaturverzeichnis

[1]Vu Ngoc Khiem, Helga Krieger, Mikhail Itskov, Thomas Gries, Scott Stapleton *An averaging based hyperelastic modeling and experimental analysis of non-crimp fabrics.* International journal of solids and structures 2016

[2]Philip Harrison *Modelling the forming mechanics of engineering fabrics using a mutually constrained pantographic beam and membrane mesh.* Composites: Part A 2015

[3] P. Boisse, N. Hamila, E. Guzman-Maldonado, A Madeo, G. Hivet, F. Dell'Isola *The bias-extension test for the analysis of in-plane shear properties of textile composite reinforcements and prepregs: a review.* International Journal of Materials Forming 2016

[4] Jorgen S Bergstrom *Mechanics of solid polymers* Elsivier 2015

[5] ABAQUS CAE user manual

[6] Yang,Sueng-Yong *Conversion of ABAQUS user material sunroutines*

[7] Writing user subroutines with abaqus

[8] Seid Koric, Lance C. Hibbeler, and Brian G. Thomas *Explicit Coupled Thermo-Mechanical Finite Element Model of Steel Solidification.* International journal of numerical methods in engineering 2008

[9] Mikhail Itskov *Tensor algebra and tensor analysis for engineers* Springer 2014

[10] P Kelly. *Solid mechanics part III.* Springer 2014

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Aachen, den 01. April 2017

. . . .

(Unterschrift des Verfassers)

Appendix

1. MATLAB code for generation of non-crimp fabric in abaqus.

```
clc
clear

%INPUT DATA
%**************************************************%
%**************************************************%
lengthX=100;
lengthY=200;
thickness = 10;

N_col = 10; %X direction
N_row = 20; %Y direction

%**************************************************%
%**************************************************%

D = lengthX/N_col;

Inp=fopen('Output.inp','wt+');
fprintf(Inp,'%s\n','*Node');
StrNodeFormat='%d,  %d,  %d,  %d';
StrShellFormat='%d,  %d,  %d,  %d,  %d';
StrTriFormat='%d, %d, %d, %d';
StrBeamFormat='%d,  %d,  %d';
StrEsetFormat = '%d,  %d,    1';
```

```
%***************NODE GENERATION*****************%

TotalNode = (N_col+1)*(N_row +2) + (N_col+2)*(N_row + 1);

for i=1:(2*N_row+3)
if mod(i,2)~=0
for j=1:(N_col+1)
NodeNumber = j+  (2*N_col+3)*(i-1)/2;
index{NodeNumber} = NodeNumber;
x{NodeNumber} =  (j-1)*D;
y{NodeNumber} = (i-2)*(D/2);
z{NodeNumber} = 0;
end
else
for j=1:(N_col+2)
NodeNumber = j+ (N_col+1) + (2*N_col+3)*(i-2)/2;
index{NodeNumber} = NodeNumber;
x{NodeNumber} =  (2*j-3)*(D/2);
y{NodeNumber} = (i-2)*(D/2);
z{NodeNumber} = 0;
end
end
end

for i=1: TotalNode
Node_M{i}=sprintf(StrNodeFormat,i,x{i},y{i},z{i});
end
for i=1 : (TotalNode)
fprintf(Inp,'%s\n',Node_M{i});
end

%***************SHELL ELEMENT GENERATION*****************%


fprintf(Inp,'%s\n','*Element, type=S4R');
```

```
E_index = 0;
for i=1:(2*N_row+1)
if mod(i,2)~=0
for j=(1+(2*N_col+3)*(i-1)/2) : (N_col+1  +(2*N_col+3)*(i-1)/2)
E_index = E_index +1;
Shell{E_index} =
sprintf(StrShellFormat,E_index,j,(j+N_col+2),
         (j+2*N_col+3),(j+N_col+1));
fprintf(Inp,'%s\n',Shell{E_index});
end
else
for j=(1+ (N_col+2) + (2*N_col+3)*(i-2)/2)
        : (N_col + (N_col+2) + (2*N_col+3)*(i-2)/2)
E_index = E_index +1;
Shell{E_index} =
sprintf(StrShellFormat,E_index,j,(j+N_col+2),
         (j+2*N_col+3),(j+N_col+1));
fprintf(Inp,'%s\n',Shell{E_index});
end
end
end
TotalElement = E_index;


fprintf(Inp,'%s\n','*Elset, elset=QuadshellElem, generate');
QuadSet = sprintf(StrEsetFormat,1,TotalElement);
fprintf(Inp,'%s\n',QuadSet);



%----------Generation of Triangular shell elements--------%
%---------------------------------------------------------%
```

```
fprintf(Inp,'%s\n','*Element, type=S3R');

Tri_index=0;
for i=[1 (2*N_row+3)]
if (i==1)
for j=1:N_col
Tri_index=Tri_index+1;
Triele{Tri_index}=
sprintf(StrTriFormat,Tri_index+TotalElement,j,j+N_col+2,j+1);
end
end

if (i==(2*N_row+3))
for j=(1+(2*N_col+3)*(i-1)/2) : (N_col+(2*N_col+3)*(i-1)/2)
Tri_index=Tri_index+1;
Triele{Tri_index}=
sprintf(StrTriFormat,Tri_index+TotalElement,j,j-N_col-1,j+1);
end
end
end



for j= [1 2]
if (j==1)
for i=1:N_row
Tri_index=Tri_index+1;
k=((2*N_col+3)*(i-1));
Triele{Tri_index}=
sprintf(StrTriFormat,Tri_index+TotalElement,N_col+2+k,
         (2*N_col)+4+k,(3*N_col)+5+k);
end
end

if (j==2)
```

```
for i=1:N_row
Tri_index=Tri_index+1;
k=((2*N_col+3)*(i-1));
Triele{Tri_index}=
sprintf(StrTriFormat,Tri_index+TotalElement,
        (2*N_col)+3+k,(3*N_col)+4+k,(4*N_col)+6+k);
end
end
end


for i=1:Tri_index
fprintf(Inp,'%s\n', Triele{i});
end


fprintf(Inp,'%s\n','*Elset, elset=TriShellElement, generate');
TriSet =
sprintf(StrEsetFormat,1+TotalElement,Tri_index+TotalElement);
fprintf(Inp,'%s\n',TriSet);



%--------------------Beam Element Generation---------------------%


fprintf(Inp,'%s\n','*Element, type=B31');

B_index=-1;
for i= 1:(2*N_row+2)
if mod(i,2)~=0
for j=(1+(2*N_col+3)*(i-1)/2) : (N_col+1  +(2*N_col+3)*(i-1)/2)
B_index=B_index+2;
Beam{B_index}=
sprintf(StrBeamFormat,B_index+Tri_index+TotalElement,
          j,j+N_col+1);
Beam{B_index+1}=
sprintf(StrBeamFormat, B_index+1+Tri_index+TotalElement,
          j,j+N_col+2);
end
else

for j=(N_col+2 + (2*N_col+3)*(i-2)/2) :
        ((2*N_col+2) + (2*N_col+3)*(i-2)/2)
B_index=B_index+2;
Beam{B_index}=
sprintf(StrBeamFormat,B_index+Tri_index+TotalElement,
         j,j+N_col+2);
Beam{B_index+1}=
sprintf(StrBeamFormat,B_index+1+Tri_index+TotalElement,
         j+N_col+2,j+1);
end
end
end
B_index=B_index+1;
for i=[1 (2*N_row+3)]
if (i==1)
for j=1:N_col
B_index=B_index+1;
Beam{B_index}=
sprintf(StrBeamFormat,B_index+Tri_index+TotalElement,
         j,j+1);
end
end

if (i==(2*N_row+3))
for j=(1+(2*N_col+3)*(i-1)/2) : (N_col+(2*N_col+3)*(i-1)/2)
B_index=B_index+1;
Beam{B_index}=
sprintf(StrBeamFormat,B_index+Tri_index+TotalElement,
         j,j+1);
end
end
end

for j= [1 2]
if (j==1)
for i=1:N_row
B_index=B_index+1;
k=((2*N_col+3)*(i-1));
Beam{B_index}=
sprintf(StrBeamFormat,B_index+Tri_index+TotalElement,
          N_col+2+k,(3*N_col)+5+k);
end
end

if (j==2)
for i=1:N_row
B_index=B_index+1;
k=((2*N_col+3)*(i-1));
Beam{B_index}=
sprintf(StrBeamFormat,B_index+Tri_index+TotalElement,
         (2*N_col)+3+k,(4*N_col)+6+k);
end
end
end


for i=1:B_index
fprintf(Inp,'%s\n', Beam{i});
end

fprintf(Inp,'%s\n','*Elset, elset=BeamLayer, generate');
Beam1 =
 sprintf(StrEsetFormat,1+Tri_index+TotalElement,
            B_index+Tri_index+TotalElement);
fprintf(Inp,'%s\n',Beam1);


fclose(Inp);
```

2. MATLAB symbolic code for comparison of results with fortran code.

```
%**********Specify the constants************%
mu= 10;
kappa= 1.3;
wa= 0.3;
a= 45;

%********Calculate the Right Cauchy Green Tensor******%

syms CC11 CC12 CC21 CC22 real

C(1,1) = CC11;
C(1,2) = CC12;
C(2,1) = CC21;
C(2,2) = CC22;
C(3,3) = 1/(CC11*CC22 - CC12*CC21);
C(1,3) = 0;
C(3,1) = 0;
C(2,3) = 0;
C(3,2) = 0;

C_sym = 0.5*(C + transpose(C));
```

```
%********Defining the structural tensor and pressure term*****%

L = [(wa*cosd(a)^2)+(1-wa)/3 wa*cosd(a)*sind(a) 0;
      wa*cosd(a)*sind(a) (wa*sind(a)^2)+(1-wa)/3 0
     ; 0 0 (1-wa)/3 ];

pres = ((kappa*2*(1-wa)/3)*(1/(C_sym(1,1)*C_sym(2,2)
        -C_sym(1,2)*C_sym(2,1))) +
        (mu*2*(wa-1)/3)*((C_sym(1,1)*C_sym(2,2)
        -C_sym(1,2)*C_sym(2,1))));

%**********Equation for strain energy function*******%

psi = kappa*(trace(C_sym*L)-1) + mu*(trace(inv(C_sym)*L)-1)
     - pres*(det(C_sym)-1);

%**********Equation for Piola Kirchoff Stress**********%

for i=1:2
for j=1:2
S(i,j)= 2*diff(psi,C(i,j));
end
end

%**********Assumption of values to cross check with fortran**********%

CC11 = 17.277320000000003;
CC12 = 27.643700000000003;
CC21 = 27.643700000000003;
CC22 = 44.373620000000003;

format long

eval(S)
```

3. Fortran code for comparison with matlab

```
program matlabtally

implicit none


integer :: i,j

real*8 Iden(3,3), C(3,3), La(3,3), C_inv(3,3), PK_2(3,3)

real :: muein,kappaein,wa,pres
real :: a,det,det1,det2,det3,detinv
real*8 zero,one,two,three,half,third,four,Pi,two_third
parameter(zero=0.d0,one=1.d0,two=2.d0,three=3.d0,half=0.5d0,
third=1.d0/3.d0,two_third=2.d0/3.d0,four=4.d0,Pi=3.1415926d0)




! Read material properties needed here
a  = 0.25;
muein =     10;
kappaein = 1.3;
wa =     0.3;


! START LOOP OVER MATERIAL POINTS:


C(1,1) = 17.277320000000003d0
C(1,2) = 27.643700000000003d0
```

```
C(1,3) = 0.0d0
C(2,1) = C(1,2)
C(2,2) = 44.373620000000003d0
C(2,3) = 0.0d0
C(3,1) = 0.0d0
C(3,2) = 0.0d0
C(3,3) = 1.d0/(C(1,1)*C(2,2)-C(1,2)**2)


La(1,1) = cos(Pi*a)*cos(Pi*a);
La(1,2) = cos(Pi*a)*sin(Pi*a);
La(1,3) = 0;
La(2,1) = cos(Pi*a)*sin(Pi*a);
La(2,2) = sin(Pi*a)*sin(Pi*a);
La(2,3) = 0;
La(3,1) = 0;
La(3,2) = 0;
La(3,3) = 0;


Iden(1,1) = 1.d0
Iden(1,2) = 0.d0
Iden(1,3) = 0.d0
Iden(2,1) = 0.d0
Iden(2,2) = 1.d0
Iden(2,3) = 0.d0
Iden(3,1) = 0.d0
Iden(3,2) = 0.d0
Iden(3,3) = 1.d0


!**********************Compute the Inverse of the Stretch tensor matrix

! Calculate the determinanat inverse
```

```
det1 =   C(1,1)* C(2,2)* C(3,3) - C(1,1)* C(2,3)* C(3,2)
det2 = - C(1,2)* C(2,1)* C(3,3) + C(1,2)* C(2,3)* C(3,1)
det3 =   C(1,3)* C(2,1)* C(3,2) - C(1,3)* C(2,2)* C(3,1)
det  =    det1 + det2 + det3
detinv = 1.d0/det


! Calculate the inverse of the matrix
C_inv(1,1) = +detinv * (C(2,2)*C(3,3) - C(2,3)*C(3,2))
C_inv(2,1) = -detinv * (C(2,1)*C(3,3) - C(2,3)*C(3,1))
C_inv(3,1) = +detinv * (C(2,1)*C(3,2) - C(2,2)*C(3,1))
C_inv(1,2) = -detinv * (C(1,2)*C(3,3) - C(1,3)*C(3,2))
C_inv(2,2) = +detinv * (C(1,1)*C(3,3) - C(1,3)*C(3,1))
C_inv(3,2) = -detinv * (C(1,1)*C(3,2) - C(1,2)*C(3,1))
C_inv(1,3) = +detinv * (C(1,2)*C(2,3) - C(1,3)*C(2,2))
C_inv(2,3) = -detinv * (C(1,1)*C(2,3) - C(1,3)*C(2,1))
C_inv(3,3) = +detinv * (C(1,1)*C(2,2) - C(1,2)*C(2,1))

pres  =    (kappaein*2.d0*(1.d0-wa)*third*C(3,3))
            +(muein*2.d0*(wa-1.d0)*third*C_inv(3,3))

PK_2  =    (kappaein*2.d0*wa*La)
         + kappaein*2.d0*(1.d0-wa)*third*Iden
          -muein*2.d0*wa*matmul(C_inv,matmul(La,C_inv))+
           muein*2.d0*(wa-1.d0)*third*matmul(C_inv,C_inv) -
            pres*(C_inv)


do i=1,3
do j=1,3
print *, PK_2(i,j)
end do
end do
end program matlabtally
```

4. VUMAT subroutine in fortran for shell elements

```
!*********************************************************************
!
! A simple VUMAT for a single fibre bundle material
!
! Sumedh Joshi- Mini thesis
subroutine vumat(
+      nblock, ndir, nshr, nstatev, nfieldv, nprops, lanneal,
+      stepTime, totalTime, dt, cmname, coordMp, charLength,
+      props, density, strainInc, relSpinInc,
+      tempOld, stretchOld, defgradOld, fieldOld,
+      stressOld, stateOld, enerInternOld, enerInelasOld,
+      tempNew, stretchNew, defgradNew, fieldNew,
+      stressNew, stateNew, enerInternNew, enerInelasNew)

include 'vaba_param.inc'

dimension props(nprops), density(nblock), coordMp(nblock,*),
+      charLength(nblock), strainInc(nblock,ndir+nshr),
+      relSpinInc(nblock,nshr), tempOld(nblock),
+      stretchOld(nblock,ndir+nshr),
+      defgradOld(nblock,ndir+nshr+nshr),
+      fieldOld(nblock,nfieldv), stressOld(nblock,ndir+nshr),
+      stateOld(nblock,nstatev), enerInternOld(nblock),
+      enerInelasOld(nblock), tempNew(nblock),
+      stretchNew(nblock,ndir+nshr),
+      defgradNew(nblock,ndir+nshr+nshr),
+      fieldNew(nblock,nfieldv),
+      stressNew(nblock,ndir+nshr), stateNew(nblock,nstatev),
+      enerInternNew(nblock), enerInelasNew(nblock)
```

```
character*80 cmname

integer km

real*8 Iden(3,3), La(3,3)
real*8 U(3,3),U_inv(3,3),T_tau(3,3)

real*8 muein,kappaein,wa,a

real*8 zero,one,two,three,half,third,four,Pi,two_third
real*8 det,det1,det2,det3,detinv,pres
parameter(zero=0.0d0,one=1.0d0,two=2.0d0,three=3.0d0,half=0.5d0,
+          third=1.0d0/3.0d0,two_third=2.d0/3.0d0,Pi=3.1415926d0)



! Read material properties needed here

muein =     PROPS(1)
kappaein=   PROPS(2)
wa =        PROPS(3)
a     =     PROPS(4)



!*************Call the first structural tensor********************!


La(1,1) = cos(Pi*a)**2.d0;
La(1,2) = sin(Pi*a)*cos(Pi*a);
La(1,3) = 0;
La(2,1) = sin(Pi*a)*cos(Pi*a);
La(2,2) = sin(Pi*a)**2.d0;
La(2,3) = 0;
La(3,1) = 0;
La(3,2) = 0;
La(3,3) = 0;

!****************specify the identity tensor******************!

Iden(1,1) = 1.d0
Iden(1,2) = 0.d0
Iden(1,3) = 0.d0
Iden(2,1) = 0.d0
Iden(2,2) = 1.d0
Iden(2,3) = 0.d0
Iden(3,1) = 0.d0
Iden(3,2) = 0.d0
Iden(3,3) = 1.d0


! START LOOP OVER MATERIAL POINTS:

do km=1,nblock
U(1,1) = StretchNew(km,1)
U(2,2) = StretchNew(km,2)
U(1,2) = Stretchnew(km,4)
U(2,1) = U(1,2)
U(3,3) = 1.d0/((U(2,2)*U(1,1))-(U(1,2)*U(2,1)))
U(1,3) = zero
U(2,3) = zero
U(3,1) = zero
U(3,2) = zero



!*********************Compute the Inverse of the Stretch tensor matrix

! Calculate the determinanat inverse
```

```
det1 =   U(1,1)* U(2,2)* U(3,3) - U(1,1)* U(2,3)* U(3,2)
det2 = - U(1,2)* U(2,1)* U(3,3) + U(1,2)* U(2,3)* U(3,1)
det3 =   U(1,3)* U(2,1)* U(3,2) - U(1,3)* U(2,2)* U(3,1)
det  =   det1 + det2 + det3
detinv = 1.d0/det


! Calculate the inverse of the matrix
U_inv(1,1) = +detinv * (U(2,2)*U(3,3) - U(2,3)*U(3,2))
U_inv(2,1) = -detinv * (U(2,1)*U(3,3) - U(2,3)*U(3,1))
U_inv(3,1) = +detinv * (U(2,1)*U(3,2) - U(2,2)*U(3,1))
U_inv(1,2) = -detinv * (U(1,2)*U(3,3) - U(1,3)*U(3,2))
U_inv(2,2) = +detinv * (U(1,1)*U(3,3) - U(1,3)*U(3,1))
U_inv(3,2) = -detinv * (U(1,1)*U(3,2) - U(1,2)*U(3,1))
U_inv(1,3) = +detinv * (U(1,2)*U(2,3) - U(1,3)*U(2,2))
U_inv(2,3) = -detinv * (U(1,1)*U(2,3) - U(1,3)*U(2,1))
U_inv(3,3) = +detinv * (U(1,1)*U(2,2) - U(1,2)*U(2,1))



!**********************Compute the Cauchy Stress**************************

pres  =     (kappaein*2.d0*(1.d0-wa)*third*U(3,3)**2.d0)
+              +(muein*2.d0*(wa-1.d0)*third*U_inv(3,3)**2.d0)

T_tau =     kappaein*2.d0*(wa*matmul(U,matmul(La,U)))
+             + kappaein*2.d0*(1.d0-wa)*third*matmul(U,U)
+             - muein*2.d0*wa*matmul(U_inv,matmul(La,U_inv))
+             + muein*2.d0*(wa-1.d0)*third*matmul(U_inv,U_inv)
+             - pres*Iden

!*********************Update the stress vector*****************************
```

```
stressNew(km,1) = T_tau(1,1)
stressNew(km,2) = T_tau(2,2)
stressNew(km,3) = T_tau(3,3)
stressNew(km,4) = T_tau(1,2)
stressNew(km,5) = T_tau(2,3)
stressNew(km,6) = T_tau(1,3)



print *, "h" , pres
!  print *, "T" , T_tau
print *, "U" , U



enddo ! end loop over material points

end subroutine vumat
```