Implementation of a Viscoelastic Model for Hydrogels in ABAQUS

Alan Jason Correa

October 10, 2022

Contents

List of Figures

Chapter 1

Introduction

Hydrogel materials are very soft materials consisting of polymer networks and solvent molecules. These materials may exhibit large volume changes depending on their external chemical and mechanical environment. They also have viscoelastic properties which is common for many polymeric materials. Due to their favourable properties, they are increasingly used in biomedical applications. For example, hydrogels are used for contact eye lenses and also as filler materials for intervertebral spinal discs. In order to use hydrogels in an effective manner for such applications, it has become important to characterize these materials with mathematical models and to thoroughly understand their properties. These mathematical (material) models are then used in simulations to design solutions to various problems encountered in the field.

Previously, material models for hydrogels, that took into account the micro-scale chemical and mechanical phenomena, have been developed[7]. However, at the macro-scale, the mechanical response of an hydrogel is similar to that of a viscoelastic material. Hence, the goal of this work is to characterize the macro-scale mechanical properties of a hydrogel by using only the finite viscoelasticity theory as described in Chapter 2. For this a user material subroutine `UMAT` for the material model will be implemented, details of which are given in Chapter 3, so that it can be used for finite element simulations in ABAQUS. Furthermore, the parameter identification procedure for the implemented viscoelastic material model will also be explored in Chapter 4 followed by numerical simulations in Chapter 5.

Chapter 2

Theory

2.1 Preliminary Continuum Mechanics

In this section, the necessary concepts and results of Continuum Mechanics theory will be discussed, without getting into the details. The contents are largely based on the lecture notes provided by Prof. Itskov for his course on Continuum Mechanics[3] at RWTH Aachen University. For an in-depth coverage of these fundamentals, attending lectures or reading any reference book on this subject, if not required, is highly recommended. The following concepts and results are instrumental to understanding various material modelling theories, and hence, are quickly reviewed.

2.1.1 Kinematics

Material Bodies and Configurations A *material body*, say B, is understood to be a set of material particles occupying some bounded region, say $\mathcal{B}$, of a three-dimensional Euclidian space at various times. These particles are characterized by some positive measure called mass. One of the basic assumptions of continuum mechanics is that the material is continuously distributed in bodies at all scales. Thus, only the mathematical treatment of this bounded region is carried out without paying much attention to the actual physical body. Mathematically, a continuum is defined as a continuous compact metric space, while for practical purposes we are describing smooth solids or confined liquids.

At any time t, a mapping of the body B into a bounded region $\mathcal{B}$ of the three-dimensional Euclidian space is assumed to be one to one and is referred to as a *configuration.* Thus any point P of the body B can be associated with

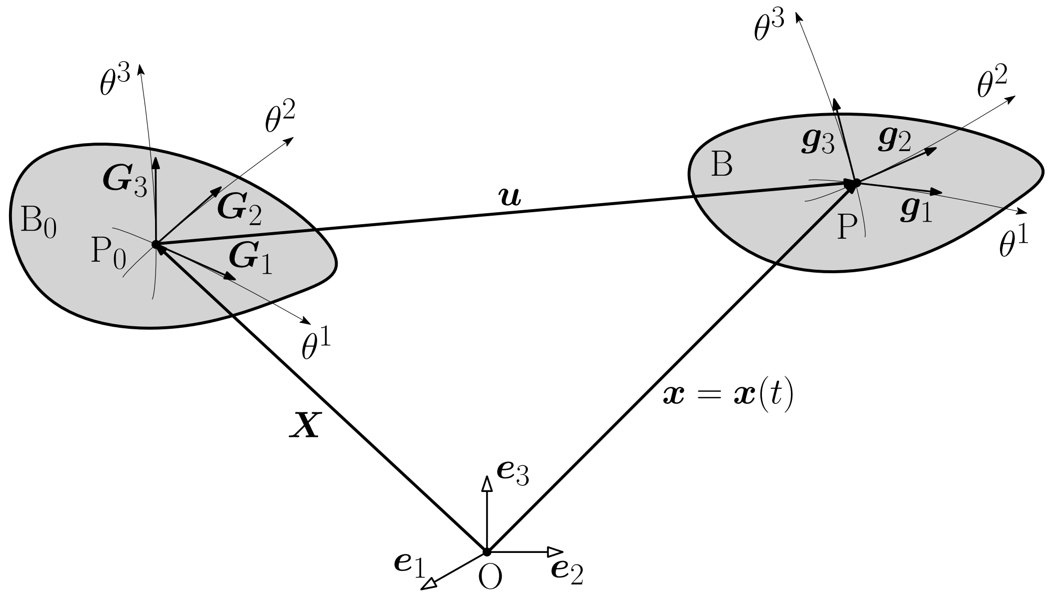


Figure 2.1: Motion of a body

a position vector $\boldsymbol{x} \in \mathbb{E}^3$. To describe the motion and deformation of the body B it is convenient to fix some configuration $\mathcal{B}_0$ at time t_0. This fixed configuration $\mathcal{B}_0$ is called the *reference configuration*.

Coordinates, Position Vectors and Displacement With a coordinate system $\theta^i (i = 1, 2, 3)$ the point P can be defined by its coordinates at time t_0. Thus we can write

$$\boldsymbol{x} = \boldsymbol{x}(\theta^1, \theta^2, \theta^3, t), \quad \theta^i = \theta^i(\boldsymbol{x}, t), \quad i = 1, 2, 3 \tag{2.1}$$

and

$$\boldsymbol{X} = \boldsymbol{x}(\theta^1, \theta^2, \theta^3, t_0) = \boldsymbol{X}(\theta^1, \theta^2, \theta^3), \quad \theta^i = \theta^i(\boldsymbol{X}, t_0), \quad i = 1, 2, 3 \tag{2.2}$$

where $\boldsymbol{X}$ denotes the *position vector* of the point P_0 in the reference configuration (Fig. 2.1). The coordinates $\theta^i (i = 1, 2, 3)$ identify the point P at any time. They are often referred to as convective coordinates because the coordinate lines are embedded into the material and deform with the body.

The difference between the position vectors of the point P in the current and reference configuration

$$\boldsymbol{u} = \boldsymbol{u}(\theta^1, \theta^2, \theta^3, t) = \boldsymbol{x} - \boldsymbol{X} \tag{2.3}$$

is called displacement.

The Euclidian space is characterized by the existence of an orthonormal basis given by a set of mutually orthogonal unit vectors, say $\boldsymbol{e}_i$, with the property $\boldsymbol{e}_i \cdot \boldsymbol{e}_i = \delta^{ij}$, where δ^{ij} denotes the Kronecker delta and is defined by

$$\delta_{ij} = \delta^{ij} = \delta^i_j = \begin{cases} 1, & \text{for } i = j \\ 0, & \text{for } i \neq j \end{cases} \tag{2.4}$$

With respect to this basis, the position vectors and displacement can be expressed as

$$\boldsymbol{X} = X^i \boldsymbol{e}_i, \qquad X^j = \boldsymbol{X} \cdot \boldsymbol{e}_j, \qquad j = 1,2,3 \tag{2.5}$$

$$\boldsymbol{u} = u^i \boldsymbol{e}_i, \qquad u^j = \boldsymbol{u} \cdot \boldsymbol{e}_j, \qquad j = 1,2,3 \tag{2.6}$$

$$\boldsymbol{x} = x^i \boldsymbol{e}_i, \qquad x^j = \boldsymbol{x} \cdot \boldsymbol{e}_j = X^j + u^j, \qquad j = 1,2,3 \tag{2.7}$$

where henceforth Einstein's summation convention over repeated indexes is applied. X^j and $x^j (j = 1,2,3)$ are called referential and current coordinate, respectively.

Other bases suitable for the description of deformation are formed by the vectors tangent to the coordinate lines in the reference and current cofigurations as shown in Fig. 2.1. These tangent vectors are defined by the derivatives of the position vectors with respect to the convective coordinates. For short, these derivatives will be denoted by $\dfrac{\partial\left(\bullet\right)}{\partial\theta^i} = \left(\bullet\right)_{,i}$ .Thus

$$\boldsymbol{G}_i = \frac{\partial \boldsymbol{X}}{\partial \theta^i} = \boldsymbol{X}_{,i} = X^j{}_{,i}\, e_j, \quad \boldsymbol{g}_i = \frac{\partial \boldsymbol{x}}{\partial \theta^i} = x_{,i} = \boldsymbol{x}^j{}_{,i}\, e_j, \quad i = 1,2,3 \tag{2.8}$$

where and henceforth we assume that the functions $\boldsymbol{X} = \boldsymbol{X}(\theta^1, \theta^2, \theta^3)$ and $\boldsymbol{x} = \boldsymbol{x}(\theta^1, \theta^2, \theta^3, t)$ are sufficiently differentiable.

It is also useful to define the bases dual to $\boldsymbol{G}_i$ and $\boldsymbol{g}_i (i = 1,2,3)$:

$$\boldsymbol{G}_i \cdot \boldsymbol{G}^j = \delta^j_i, \quad \boldsymbol{g}_i \cdot \boldsymbol{g}^j = \delta^j_i, \quad i,j = 1,2,3. \tag{2.9}$$

Thus we can write

$$\boldsymbol{G}^i = \frac{\partial \theta^i}{\partial X^j} \boldsymbol{e}^j, \quad \boldsymbol{g}^i = \frac{\partial \theta^i}{\partial x^j} \boldsymbol{e}^j, \quad i,j = 1,2,3. \tag{2.10}$$

Deformation Gradient The current postion $\boldsymbol{x}$ of some point P and its displacement $\boldsymbol{u}$ characterize only the *motion* of the body in this point. In order to describe its deformation it is necessary to know how fast these vectors change in a neighbourhood of this point. For example, there is no deformation if all points get the same displacement. In this case we deal with the rigid body motion.

The deformation gradient is defined as the gradient of the position vector in the current configuration and is denoted by $\mathbf{F}$:

$$\mathbf{F} = \operatorname{Grad} \boldsymbol{x} = \mathrm{F}^{i}_{\cdot j}\, \boldsymbol{e}_i \otimes \boldsymbol{e}^j \tag{2.11}$$

where the matrix $\left[\mathrm{F}^{i}_{\cdot j}\right]$ is given by

$$\left[\mathrm{F}^{i}_{\cdot j}\right] = \begin{bmatrix} \frac{\partial x^1}{\partial X^1} & \frac{\partial x^1}{\partial X^2} & \frac{\partial x^1}{\partial X^2} \\ \frac{\partial x^2}{\partial X^1} & \frac{\partial x^2}{\partial X^2} & \frac{\partial x^2}{\partial X^2} \\ \frac{\partial x^3}{\partial X^1} & \frac{\partial x^3}{\partial X^2} & \frac{\partial x^3}{\partial X^2} \end{bmatrix} = \begin{bmatrix} \frac{\partial u^1}{\partial X^1} + 1 & \frac{\partial u^1}{\partial X^2} & \frac{\partial u^1}{\partial X^2} \\ \frac{\partial u^2}{\partial X^1} & \frac{\partial u^2}{\partial X^2} + 1 & \frac{\partial u^2}{\partial X^2} \\ \frac{\partial u^3}{\partial X^1} & \frac{\partial u^3}{\partial X^2} & \frac{\partial u^3}{\partial X^2} + 1 \end{bmatrix} \tag{2.12}$$

The deformation gradient can also be written in a more compact form using the tangent vectors.

$$\mathbf{F} = \boldsymbol{g}_i \otimes \boldsymbol{G}^j \tag{2.13}$$

Deformation of line, surface and volume elements If an infinitesimal line element $d\boldsymbol{X}$ in the reference configuration is considered along with its counterpart $d\boldsymbol{x}$ in the current configuration, it can be shown that

$$d\boldsymbol{x} = \mathbf{F} d\boldsymbol{X}, \quad d\boldsymbol{X} = \mathbf{F}^{-1} d\boldsymbol{x} \tag{2.14}$$

and hence suggesting the following symbolic representation for the deformation gradient

$$\mathbf{F} = \operatorname{Grad} \boldsymbol{x} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}}, \quad \mathbf{F}^{-1} = \operatorname{grad} \boldsymbol{X} = \frac{\partial \boldsymbol{X}}{\partial \boldsymbol{x}}. \tag{2.15}$$

The deformation gradient is also suitable to describe the deformation of volume and surface elements. If dV_0 is the volume of an infinitesimal volume element in the reference configuration and dV is the corresponding volume of the same volume element after deformation in the current configuration, then it can be shown that

$$\frac{dV}{dV_0} = \det \mathbf{F} = |\mathrm{F}^{i}_{\cdot j}| = J \tag{2.16}$$

In the same way if we consider vectorial surface elements $d\boldsymbol{S}$ and $d\boldsymbol{s}$ along with the positive unit normals to these surfaces as $\boldsymbol{N}$ and $\boldsymbol{s}$ in the reference and current configuration respectively, we have the relation

$$d\boldsymbol{s} = J\mathbf{F}^{-T} d\boldsymbol{S}, \quad ds\boldsymbol{n} = J\mathbf{F}^{-T}\boldsymbol{N}dS \tag{2.17}$$

which is also referred to as Nanson's formula.

Strain A material is said to be strained in some point $\mathrm{P}(\boldsymbol{X})$ if at least one of the infinitesimal line elements $d\boldsymbol{X}$ based on $\boldsymbol{X}$ changes its length after deformation. For the square lengths we can write by virtue of Eq. (2.14) we have

$$\begin{aligned} ||d\boldsymbol{x}||^2 &= d\boldsymbol{x} \cdot d\boldsymbol{x} = (\mathbf{F}d\boldsymbol{X}) \cdot (\mathbf{F}d\boldsymbol{X}) \\ &= (d\boldsymbol{X}\mathbf{F}^T) \cdot (\mathbf{F}d\boldsymbol{X}) = d\boldsymbol{X}(\mathbf{F}^T\mathbf{F})d\boldsymbol{X} \\ &= d\boldsymbol{X}\mathbf{C}d\boldsymbol{X}, \end{aligned} \tag{2.18}$$

$$\begin{aligned} ||d\boldsymbol{X}||^2 &= d\boldsymbol{X} \cdot d\boldsymbol{X} = (\mathbf{F}^{-1}d\boldsymbol{x}) \cdot (\mathbf{F}^{-1}d\boldsymbol{x}) \\ &= (d\boldsymbol{x}\mathbf{F}^{-T}) \cdot (\mathbf{F}^{-1}d\boldsymbol{x}) = d\boldsymbol{x}(\mathbf{F}^{-T}\mathbf{F}^{-1})d\boldsymbol{x} \\ &= d\boldsymbol{x}\mathbf{b}^{-1}d\boldsymbol{x}, \end{aligned} \tag{2.19}$$

where the symmetric tensors

$$\mathbf{C} = \mathbf{F}^T\mathbf{F}, \quad \mathbf{b} = \mathbf{F}\mathbf{F}^T \tag{2.20}$$

are referred to as the right and left Cauchy-Green deformation tensor respectively. They can also be expressed in terms of the tangent vectors as

$$\mathbf{C} = g_{ij}\, \boldsymbol{G}^i \otimes \boldsymbol{G}^j \tag{2.21}$$

$$\mathbf{b} = G^{ij}\, \boldsymbol{g}_i \otimes \boldsymbol{g}_j \tag{2.22}$$

where the abbreviations

$$g_{ij} = \boldsymbol{g}_i \cdot \boldsymbol{g}_j, \quad G^{ij} = \boldsymbol{G}^i \cdot \boldsymbol{G}^j, \quad i,j = 1,2,3 \tag{2.23}$$

In order to express strains, the difference of square lengths are in Eq. (2.18) and Eq. (2.19) are considered and after mathematical manipulation the we get the symmetric tensors

$$\mathbf{E} = \frac{1}{2}(\mathbf{C} - \mathbf{I}) \quad = \frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{I}) \tag{2.24}$$

$$\mathbf{e} = \frac{1}{2}(\mathbf{I} - \mathbf{b}^{-1}) = \frac{1}{2}(\mathbf{I} - \mathbf{F}^{-T}\mathbf{F}^{-1}) \tag{2.25}$$

which are called the Green-Lagrange strain tensor and Almansi strain tensor respectively. The material body is thus said to be unstrained if $\mathbf{E} = \mathbf{e} = \mathbf{0}$ and the deformation gradient represents an orthogonal tensor, and hence, $\mathbf{F}^T\mathbf{F} = \mathbf{I}$. If this is the case in every point of the body, then we deal with the rigid body motion.

Stretch and Shear With the aid of the Cauchy-Green tensor, it is possible to express the stretch of a line element as well as change in angle between two orthogonal line elements defined in the reference configuration. The ratio of the deformed to the reference length of a line element is called stretch. If $\boldsymbol{N}$ and $\boldsymbol{n}$ are unit vectors along the infinitesimal line element $d\boldsymbol{X}$ and its counterpart in the current configuration $d\boldsymbol{x}$, respectively. Furthermore with $d\boldsymbol{X} = ||d\boldsymbol{X}||\boldsymbol{N}$ and $d\boldsymbol{x} = ||d\boldsymbol{x}||\boldsymbol{n}$, the stretch in the direction of $\boldsymbol{N}$ takes the form

$$\begin{aligned}\lambda(\boldsymbol{N}) &= \sqrt{\frac{||d\boldsymbol{x}||^2}{||d\boldsymbol{X}||^2}} \\ &= (\boldsymbol{NCN})^{\frac{1}{2}} \end{aligned} \tag{2.26}$$

and similarly the strecth in the direction of $\boldsymbol{n}$ it can be shown

$$\lambda(\boldsymbol{n}) = (\boldsymbol{n}\boldsymbol{b}^{-1}\boldsymbol{n})^{-\frac{1}{2}} \tag{2.27}$$

By representing the Cauchy-Green tensor with respect to an orthonormal basis $\boldsymbol{N}_i \cdot \boldsymbol{N}_j = \delta_{ij}, \ (i, j = 1, 2, 3)$, we get

$$\lambda(\boldsymbol{N}_i) = (\boldsymbol{N}_i\boldsymbol{C}\boldsymbol{N}_i)^{\frac{1}{2}} = \sqrt{\mathrm{C}_{ii}}, \quad \text{no sum over } i = 1, 2, 3 \tag{2.28}$$

Thus the square roots of the diagonal components of the Cauchy-Green tensor expressed in an orthonormal basis represent the stretches in the corresponding directions $\boldsymbol{N}_i$ $(i = 1, 2, 3)$ The decrease in the angle after deformation between two orthogonal line elements, say $d\boldsymbol{X}_i = ||d\boldsymbol{X}_i||\boldsymbol{N}_i, \ (i = 1, 2)$ defined in the reference configuration is given by

$$\sin\varphi_{12} = \frac{\mathrm{C}_{12}}{\sqrt{\mathrm{C}_{11}\mathrm{C}_{22}}} \tag{2.29}$$

Spectral Decomposition of Strain Tensors By setting up an eigen value problem for the right Cauchy-Green tensor we have

$$\boldsymbol{C}\boldsymbol{N} = \Lambda\boldsymbol{N}, \quad \boldsymbol{N} \neq 0 \tag{2.30}$$

where a non-zero vector $\boldsymbol{N}$ is called an eigen vector and Λ denotes the corresponding eigen value. To solve this eigen value problem, $\mathbf{C}$ and $\boldsymbol{N}$ are represented with respect to a basis. This finally results in

$$(\mathrm{C}^i_j - \Lambda\delta^i_j)N^j = 0, \quad i = 1,2,3 \tag{2.31}$$

which is a homogenous linear equation system with respect to the components of the eigen vector $\boldsymbol{N}$. This equation system has a non-trivial solution if an only if

$$|\mathrm{C}^i_j - \Lambda\delta^i_j| = 0 \tag{2.32}$$

or equivalently

$$\begin{vmatrix} \mathrm{C}^1_1 - \Lambda & \mathrm{C}^1_2 & \mathrm{C}^1_3 \\ \mathrm{C}^2_1 & \mathrm{C}^2_2 - \Lambda & \mathrm{C}^2_3 \\ \mathrm{C}^3_1 & \mathrm{C}^3_2 & \mathrm{C}^3_3 - \Lambda \end{vmatrix} = 0, \tag{2.33}$$

where $|\bullet|$ denotes the determinant of a matrix. Writing out this determinant and collecting the terms according to the powers of Λ one obtains the so-called characteristic equation which is given by

$$\Lambda^3 - \mathrm{I}_{\mathbf{C}}\Lambda^2 + \mathrm{II}_{\mathbf{C}}\Lambda - \mathrm{III}_{\mathbf{C}} = 0, \tag{2.34}$$

where the coefficients take the form

$$\mathrm{I}_{\mathbf{C}} = \mathrm{C}^i_i = \mathrm{tr}\mathbf{C} \tag{2.35}$$

$$\mathrm{II}_{\mathbf{C}} = \frac{1}{2}(\mathrm{C}^i_i\mathrm{C}^j_j - \mathrm{C}^i_j\mathrm{C}^j_i) = \frac{1}{2}[(\mathrm{tr}\mathbf{C})^2 - \mathrm{tr}\mathbf{C}^2] \tag{2.36}$$

$$\mathrm{III}_{\mathbf{C}} = |\mathrm{C}^i_j| = \det\mathbf{C} \tag{2.37}$$

The solution of the eigen value problem resulting from the characteristic equation does not depend on the choice of the coordinate system. Therefore, also the coefficients of the characteristic equation are independent of the coordinate system and, hence, are called the principal invariants of $\mathbf{C}$. Since the tensor $\mathbf{C}$ is symmetric its characteristic equation has three real roots $\Lambda_i (i = 1,2,3)$. The characteristic equation can also be represented in terms of the roots by

$$(\Lambda_1 - \Lambda)(\Lambda_2 - \Lambda)(\Lambda_3 - \Lambda) = 0 \tag{2.38}$$

By virtue of the tensor identity, $\mathrm{tr}(\mathbf{AB}) = \mathrm{tr}(\mathbf{BA})$ and taking Eq. (2.16) into account it can be shown that

$$\mathrm{tr}\mathbf{C} = \mathrm{tr}\mathbf{b} \tag{2.39}$$

$$\mathrm{tr}\mathbf{C}^2 = \mathrm{tr}\mathbf{b}^2 \tag{2.40}$$

$$\det\mathbf{C} = \det\mathbf{b} = J^2 \tag{2.41}$$

Thus, we conclude the tensors $\mathbf{C}$ and $\mathbf{b}$ have the same principal invariants

$$\mathrm{I}_{\mathbf{C}} = \mathrm{I}_{\mathbf{b}}, \quad \mathrm{II}_{\mathbf{C}} = \mathrm{II}_{\mathbf{b}}, \quad \mathrm{III}_{\mathbf{C}} = \mathrm{III}_{\mathbf{b}} \tag{2.42}$$

which are also referred to as strain invariants. Eq. (2.42) immediately implies the equivalence of the eigen values of the right and left Cauchy-Green tensors. This is an important property and is useful for material modelling.

The eigen vectors of symmetric tensors such as $\mathbf{C}$ and $\mathbf{b}$ are linearly independent and can be represented by an orthonormal basis. Thus, we can write the spectral decomposition as

$$\mathbf{C} = \sum_{n=1}^{3} \Lambda_i \boldsymbol{N}_i \otimes \boldsymbol{N}_i, \quad \mathbf{b} = \sum_{n=1}^{3} \Lambda_i \boldsymbol{n}_i \otimes \boldsymbol{n}_i, \tag{2.43}$$

where

$$\boldsymbol{N}_i \cdot \boldsymbol{N}_j = \delta_{ij}, \quad \boldsymbol{n}_i \cdot \boldsymbol{n}_j = \delta_{ij}, \quad i,j = 1,2,3. \tag{2.44}$$

In view of Eq. (2.18) and Eq. (2.19) the tensors $\mathbf{C}$ and $\mathbf{b}$ are positive definite which implies that their eigen values $\Lambda_i (i = 1,2,3)$ are positive. Using this property we can write the powers of these tensors on the basis of the spectral decomposition by

$$\mathbf{C}^\alpha = \sum_{n=1}^{3} \Lambda_i^\alpha \boldsymbol{N}_i \otimes \boldsymbol{N}_i, \quad \mathbf{b}^\alpha = \sum_{n=1}^{3} \Lambda_i^\alpha \boldsymbol{n}_i \otimes \boldsymbol{n}_i, \tag{2.45}$$

Polar Decomposition of Deformation Gradient The deformation gradient can be further decomposed into a rotation and stretch tensor. By setting in Eq. (2.45) $\alpha = \frac{1}{2}$ one defines the so-called right and left stretch tensors

$$\mathbf{U} = \mathbf{C}^{\frac{1}{2}} = \sum_{i=1}^{3} \lambda_i \boldsymbol{N}_i \otimes \boldsymbol{N}_i, \quad \mathbf{v} = \mathbf{b}^{\frac{1}{2}} = \sum_{i=1}^{3} \lambda_i \boldsymbol{n}_i \otimes \boldsymbol{n}_i, \tag{2.46}$$

where

$$\lambda_i = \sqrt{\Lambda_i}, \quad i = 1,2,3. \tag{2.47}$$

Further one can show that

$$\mathbf{R} = \mathbf{F}\mathbf{U}^{-1} \tag{2.48}$$

represents a proper orthogonal tensor as it can be shown that $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ and $\det(\mathbf{R}) \geq 0$. From Eq. (2.48) we further get

$$\mathbf{F} = \mathbf{R}\mathbf{U} = (\mathbf{R}\mathbf{U}\mathbf{R}^T)\mathbf{R}. \tag{2.49}$$

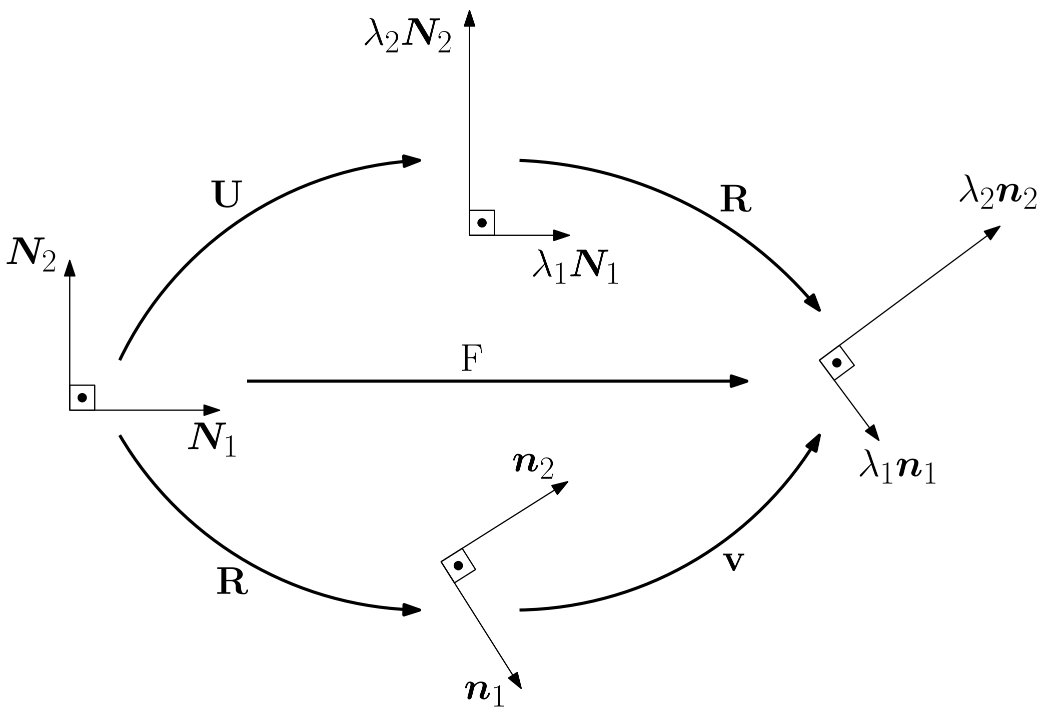


Figure 2.2: Polar decomposition

Due to the symmetry of $\mathbf{U}$ it can be shown that the tensor $\mathbf{RUR}^T$ is also symmetric and one can derive that $(\mathbf{RUR}^T)^2 = \mathbf{b}$. In view of Eq. $(2.46)_2$, there exists only one real symmetric tensor whose square is $\mathbf{b}$. Hence $\mathbf{RUR}^T = \mathbf{v}$. This results in the following identity

$$\mathbf{F} = \mathbf{RU} = \mathbf{vR} \tag{2.50}$$

which is referred to as the *polar decomposition* of the deformation gradient. By virtue of Eq. (2.50) we also obtain

$$\mathbf{U} = \mathbf{R}^T\mathbf{vR}, \quad \mathbf{C} = \mathbf{R}^T\mathbf{bR}, \quad \mathbf{v} = \mathbf{RUR}^T, \quad \mathbf{b} = \mathbf{RCR}^T \tag{2.51}$$

Considering a special case of pairwise distinct eigen values of the stretch tensors $\lambda 1 \neq \lambda 2 \neq \lambda 3 \neq \lambda 1$ and using Eq. (2.51) it can be shown that

$$\mathbf{v} = \mathbf{RUR}^T = \sum_{i=1}^{3} \lambda_i (\mathbf{R}\boldsymbol{N}_i) \otimes (\mathbf{R}\boldsymbol{N}_i) \tag{2.52}$$

Comparing this with Eq. $(2.46)_2$ we obtain

$$\boldsymbol{n}_i = \mathbf{R}\boldsymbol{N}_i, \quad \boldsymbol{V}_i = \mathbf{R}^T\boldsymbol{n}_i, \quad i = 1, 2, 3. \tag{2.53}$$

and consequently

$$\mathbf{R} = \sum_{i=1}^{3} \boldsymbol{n}_i \otimes \boldsymbol{N}_i, \quad \mathbf{F} = \sum_{i=1}^{3} \lambda_i \boldsymbol{n}_i \otimes \boldsymbol{N}_i, \quad \lambda_1 \neq \lambda_2 \neq \lambda_3 \neq \lambda_1 \tag{2.54}$$

Thus the deformation of an eigen vector $\boldsymbol{N}_i (i = 1, 2, 3)$ can be accomplished in two steps: rotation by $\mathbf{R}$ to $\boldsymbol{n}_i$ and stretching by $\mathbf{v}$ to $\lambda \boldsymbol{n}_i$ or vice versa: stretching by $\mathbf{U}$ to $\lambda \boldsymbol{N}_i$ and rotation by $\mathbf{R}$ to $\lambda \boldsymbol{n}_i$ as shown illustratively in Fig. 2.2 and mathematically as

$$\begin{aligned} \mathbf{F}\boldsymbol{N}_i &= \mathbf{v}(\mathbf{R}\boldsymbol{N}_i) &&= \mathbf{v}\boldsymbol{\lambda}_i &&= \lambda_i \boldsymbol{n}_i \quad \text{or} \\ \mathbf{F}\boldsymbol{N}_i &= \mathbf{R}(\mathbf{U}\boldsymbol{N}_i) &&= \lambda_i \mathbf{R}\boldsymbol{N}_i &&= \lambda_i \boldsymbol{n}_i, \quad i = 1, 2, 3 \end{aligned} \tag{2.55}$$

For this reason, $\mathbf{R}$ is called the rotation tensor and the eigen values $\lambda_i (i = 1, 2, 3)$ of the stretch tensors $\mathbf{U}$ and $\mathbf{v}$ are referred to as the principal stretches.

Velocity Gradient The velocity of a material particle P is defined by

$$\boldsymbol{v} = \frac{\partial \boldsymbol{x}(\boldsymbol{X}, t)}{\partial t} = \dot{\boldsymbol{x}} \tag{2.56}$$

The time derivative denoted by the superposed dot is called the material time derivative since the position vector $\mathbf{X}$ in the reference configuration is held constant. The material velocity gradient is defined similar to the deformation gradient by

$$\mathbf{L} = \mathrm{Grad}\boldsymbol{v} = \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{X}}. \tag{2.57}$$

Taking into account Eq. (2.56) and Eq. (2.15) we can write

$$\mathbf{L} = \mathrm{Grad}\dot{\boldsymbol{x}} = \frac{\partial}{\partial \boldsymbol{X}} \left[\frac{\partial \boldsymbol{x}(\boldsymbol{X}, t)}{\partial \boldsymbol{t}} \right] = \frac{\partial}{\partial t} \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}} \right) = \dot{\mathbf{F}} \tag{2.58}$$

Of special interest is also the spatial velocity gradient defined by

$$\mathbf{l} = \mathrm{grad}\boldsymbol{v} = \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}} \tag{2.59}$$

Using Eq. $(2.15)_2$ and Eq. (2.57) we thus obtain

$$\mathbf{l} = \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{X}} \frac{\partial \boldsymbol{X}}{\partial \boldsymbol{x}} = \dot{\mathbf{F}}\mathbf{F}^{-1} \tag{2.60}$$

According to Eq. (2.57) and Eq. (2.59) it follows that

$$d\boldsymbol{v} = d\dot{\boldsymbol{x}} = \mathbf{l}d\boldsymbol{x} = \mathbf{L}d\boldsymbol{X}. \tag{2.61}$$

Furthermore, the spatial gradient of velocity is usually decomposed into a symmetric $\mathbf{d} = \mathbf{d}^T$ and a skew-symmetric part $\mathbf{w} = -\mathbf{w}^T$ by

$$\mathbf{l} = \mathbf{d} + \mathbf{w}, \tag{2.62}$$

where

$$\begin{aligned}\mathbf{d} &= \frac{1}{2}\left(\mathbf{l} + \mathbf{l}^T\right) = \frac{1}{2}\left(\dot{\mathbf{F}}\mathbf{F}^{-1} + \mathbf{F}^{-T}\dot{\mathbf{F}}^T\right) = \frac{1}{2}\mathbf{F}^{-T}\dot{\mathbf{C}}\mathbf{F}^{-1} \\ &= \mathbf{F}^{-T}\dot{\mathbf{E}}\mathbf{F}^{-1}\end{aligned} \tag{2.63}$$

is called the rate of deformation tensor and

$$\mathbf{w} = -\mathbf{w}^T = \frac{1}{2}\left(\mathbf{l} - \mathbf{l}^T\right) = \frac{1}{2}\left(\dot{\mathbf{F}}\mathbf{F}^{-1} - \mathbf{F}^{-T}\dot{\mathbf{F}}^T\right)$$

is called spin (vorticity) tensor.

2.1.2 Stress

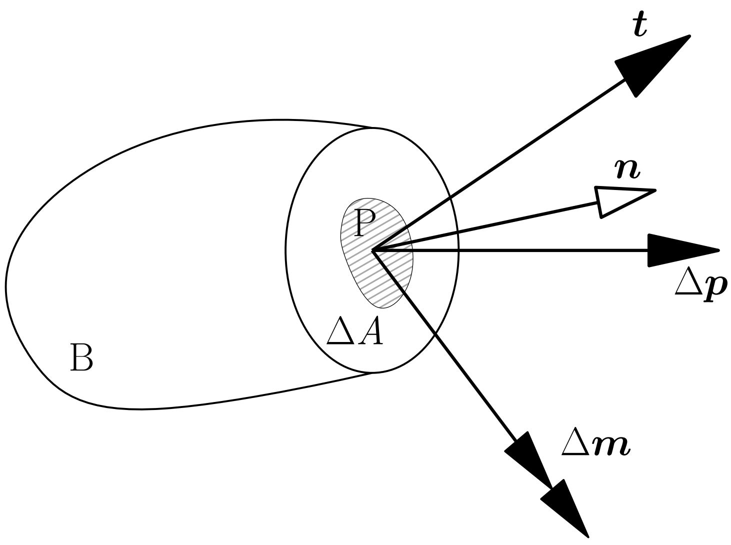


Figure 2.3: Cauchy stress vector

Let us consider a body B in the current configuration at a time t. In order to define stress in some point P let us further imagine a smooth surface going through P and separating B into two parts, of which one part is as shown in Fig. 2.3. One can then define a force $\Delta\boldsymbol{p}$ and a couple $\Delta\boldsymbol{m}$ resulting from the forces exerted by material on one side of the surface ΔA on the material on the other side. Let the area ΔA tend to zero permanently keeping P as an inner point. A basic postulate of continuum mechanics is that the limit

$$\boldsymbol{t} = \lim_{\Delta A \to 0} \frac{\Delta \boldsymbol{p}}{\Delta A} \tag{2.64}$$

exists and is finite. The so-defined vector $\boldsymbol{t}$ is called the Cauchy stress vector. The Cauchy's fundamental postulate states that the vector t depends on the surface only through the unit outward normal $\boldsymbol{n}$ as shown in Fig. 2.3. In other words, the Cauchy stress vector is the same for all surface through P which have a normal $\boldsymbol{n}$ in P. As a result one can write

$$\boldsymbol{t} = \boldsymbol{t}(\boldsymbol{x}, \boldsymbol{n}) = \boldsymbol{t}(\boldsymbol{X}, \boldsymbol{n}, t). \tag{2.65}$$

According to Cauchy's theorem the mapping $\boldsymbol{n} \to \boldsymbol{t}$ is linear provided the function Eq. (2.65) is continuous at $\boldsymbol{X}$. Hence this mapping can be described by a second-order tensor as

$$\boldsymbol{t} = \boldsymbol{\sigma} \boldsymbol{n} \tag{2.66}$$

The tensor $\boldsymbol{\sigma}$ is called the Cauchy stress tensor and is related to the surface in the current configuration.

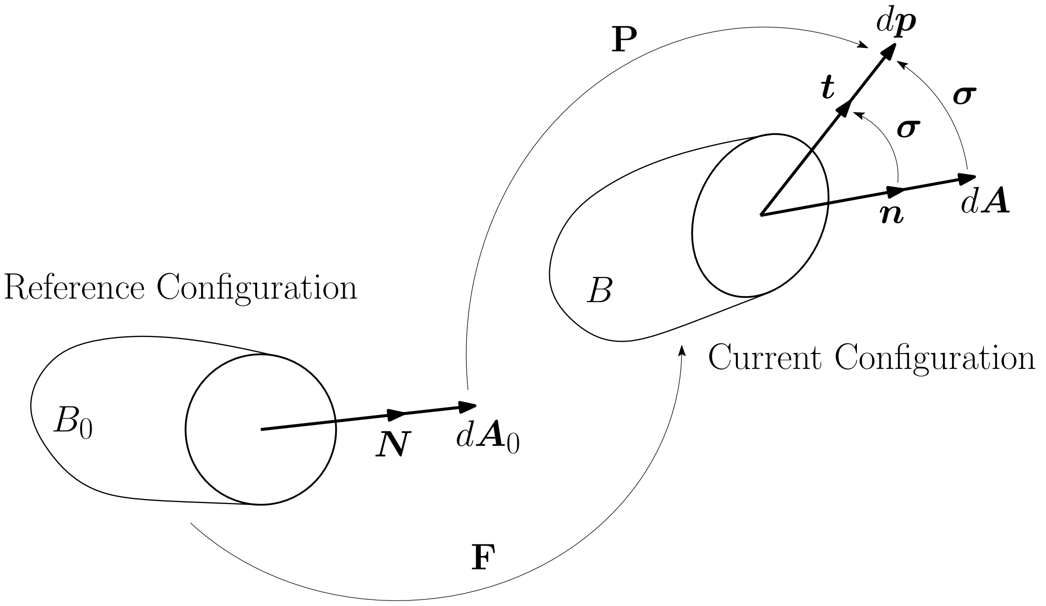


Figure 2.4: First Piola-Kirchoff stress tensor

Sometimes it is useful to define a stress tensor with respect to the reference configuration. For this consider a function $d\boldsymbol{p} = d\boldsymbol{p}(d\boldsymbol{A})$ where

$$d\boldsymbol{p} = \boldsymbol{t} dA \tag{2.67}$$

and $d\boldsymbol{A} = \boldsymbol{n}dA$ represents a vectorial surface element corresponding to the differential area dA. The function $\boldsymbol{t} = \boldsymbol{t}(\boldsymbol{n})$ is linear if and only if the function $d\boldsymbol{p} = d\boldsymbol{p}(d\boldsymbol{A})$ is linear. Furthermore, using Nanson's formula we have

$$d\boldsymbol{p} = \boldsymbol{t}dA = \boldsymbol{\sigma}d\boldsymbol{A} = J\boldsymbol{\sigma}\mathbf{F}^{-T}d\boldsymbol{A}_0 \tag{2.68}$$

With the aid of the definition

$$\mathbf{P} = J\boldsymbol{\sigma}\mathbf{F}^{-T} = \boldsymbol{\tau}\mathbf{F}^{-T} \tag{2.69}$$

where

$$\boldsymbol{\tau} = J\boldsymbol{\sigma} \tag{2.70}$$

is referred to as the Kirchoff stress tensor, we can write

$$d\boldsymbol{p} = \mathbf{P}d\boldsymbol{A}_0. \tag{2.71}$$

$\mathbf{P}$ is called the first Piola-Kirchoff stress tensor and it relates the stress vector to the surface in the reference configuration as shown in Eq. (2.71). Furthermore the second Piola-Kirchoff stress tensor is defined by

$$\mathbf{S} = J\mathbf{F}^{-1}\boldsymbol{\sigma}\mathbf{F}^{-T} = \mathbf{F}^{-1}\boldsymbol{\tau}\mathbf{F}^{-T} = \mathbf{F}^{-1}\mathbf{P} \tag{2.72}$$

which is an important stress tensor quantity but has no physical significance.

2.1.3 Balance Laws

Linear Momentum Balance If we consider the current configuration of the body B with the volume V, mass M and boundary surface A, the linear momentum of the body is defined by

$$\int_M \boldsymbol{v}\, dm \tag{2.73}$$

where $\boldsymbol{v} = \frac{\partial \boldsymbol{x}}{\partial t} = \dot{\boldsymbol{x}}$ denotes the velocity of a particle with the position vector $\boldsymbol{x}$ and mass dm. Using the relation $dm = \rho dV$, where ρ denotes the density, we can write

$$\int_M \boldsymbol{v}\, dm = \int_M \dot{\boldsymbol{x}}\, dm = \int_V \rho\dot{\boldsymbol{x}}\, dV \tag{2.74}$$

According to a fundamental principle of continuum mechanics the rate of change of the linear momentum is equal to the resultant force applied on the body. This resultant force comprises a body force (without inertia) and a surface force and is thus written by

$$\int_V \boldsymbol{f}\, dV + \int_A \boldsymbol{t}\, dA \tag{2.75}$$

Thus, the linear momentum balance (the first Euler law of motion) takes the form

$$\frac{\partial}{\partial t}\int_V \rho\dot{\boldsymbol{x}}\,dV = \int_V \boldsymbol{f}\,dV + \int_A \boldsymbol{t}\,dA \tag{2.76}$$

Using the mass conservation law we can rewrite the left hand side of Eq. (2.76) by

$$\frac{\partial}{\partial t}\int_V \rho\dot{\boldsymbol{x}}\,dV = \frac{\partial}{\partial t}\int_M \boldsymbol{v}\,dm = \int_M \dot{\boldsymbol{v}}\,dm = \int_V \rho\ddot{\boldsymbol{x}}\,dV = \int_V \rho\boldsymbol{a}\,dV \tag{2.77}$$

where $\boldsymbol{a} = \dot{\boldsymbol{v}} = \ddot{\boldsymbol{a}}$ represents the acceleration vector of particle, which finally leads to

$$\int_V \rho\ddot{\boldsymbol{x}}\,dV = \int_V \boldsymbol{f}\,dV + \int_A \boldsymbol{t}\,dA \tag{2.78}$$

By means of the Cauchy theorem Eq. (2.66) and Gauss divergence theorem from calculus we can write

$$\int_V \rho\ddot{\boldsymbol{x}}\,dV = \int_V \boldsymbol{f}\,dV + \int_V \mathrm{div}\boldsymbol{\sigma}\,dV \tag{2.79}$$

and consequently

$$\int_V (\mathrm{div}\boldsymbol{\sigma} + \boldsymbol{f} - \rho\ddot{\boldsymbol{x}})\,dV \tag{2.80}$$

This equation holds for the whole body and any arbitrary part of it with the volume dV. Assuming that the integrand in Eq. (2.80) is a continuous function in space we have

$$\mathrm{div}\boldsymbol{\sigma} + \boldsymbol{f} = \rho\ddot{\boldsymbol{x}} \tag{2.81}$$

The equation of motion Eq. (2.81) can also be written in terms of the first Piola-Kirchoff stress tensor Eq. (2.69) and is given by

$$\mathrm{Div}\mathbf{P} + \boldsymbol{f}_0 = \rho_0\ddot{\boldsymbol{x}} \tag{2.82}$$

Rotational Momentum Balance The rotational momentum (or moment of momentum) is defined by

$$\int_M \boldsymbol{x}\times\boldsymbol{v}\,dm = \int_M \boldsymbol{x}\times\dot{\boldsymbol{x}}\,dm = \int_V \boldsymbol{x}\times\rho\dot{\boldsymbol{x}}\,dV \tag{2.83}$$

where $\boldsymbol{x}$ denotes a position vector with respect to arbitrary origin O. The balance law of rotational momentum is another fundamental principle of continuum mechanics stating that the rate of the rotational momentum

$$\frac{\partial}{\partial t}\int_M \boldsymbol{x}\times\dot{\boldsymbol{x}}\,dm = \int_M \dot{\boldsymbol{x}}\times\dot{\boldsymbol{x}}\,dm + \int_M \boldsymbol{x}\times\ddot{\boldsymbol{x}}\,dm = \int_M \rho\boldsymbol{x}\times\ddot{\boldsymbol{x}}\,dV \tag{2.84}$$

is equal to the resultant of the external forces with respect to the same origin

$$\int_A \boldsymbol{x} \times \boldsymbol{t}\, dA + \int_V \boldsymbol{x} \times \boldsymbol{f}\, dV \tag{2.85}$$

Thus the rotational momentum balance can be written by

$$\int_M \rho \boldsymbol{x} \times \ddot{\boldsymbol{x}}\, dV = \int_A \boldsymbol{x} \times \boldsymbol{t}\, dA + \int_V \boldsymbol{x} \times \boldsymbol{f}\, dV \tag{2.86}$$

A consequence of this relation is (without proof) the symmetry of the Cauchy stress tensor

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}^T \tag{2.87}$$

Balance of Mechanical Energy The balance of mechanical energy can be formulated from the Cauchy equation of motion given by Eq. (2.81). Multiplying this equation scalarly with the velocity vector $\boldsymbol{v} = \dot{\boldsymbol{x}}$ we get

$$\boldsymbol{v} \cdot \mathrm{div}\boldsymbol{\sigma} + \boldsymbol{v} \cdot \boldsymbol{f} = \rho \boldsymbol{v} \cdot \ddot{\boldsymbol{x}} \tag{2.88}$$

Transforming this equation using tensor calculus identities and the symmetry property of the Cauchy stress tensor, we get

$$\mathrm{div}(\boldsymbol{v}\boldsymbol{\sigma}) - \boldsymbol{\sigma} : \mathbf{d} + \boldsymbol{v} \cdot \boldsymbol{f} = \rho \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{1}{2}\boldsymbol{v} \cdot \boldsymbol{v}\right) \tag{2.89}$$

Further, integration over the volume V of the body and applying the Gauss divergence theorem of calculus leads to the equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_M \left(\frac{1}{2}\boldsymbol{v} \cdot \boldsymbol{v}\right) dm + \int_V (\boldsymbol{\sigma} : \mathbf{d})\ dV = \int_A (\boldsymbol{v} \cdot \boldsymbol{t})\ dA + \int_V (\boldsymbol{v} \cdot \boldsymbol{f})\ dV \tag{2.90}$$

expressing the balance of mechanical energy. Every term in Eq. (2.90) has a special physical meaning connected with the mechanical energy:

$$K = \int_M \left(\frac{1}{2}\boldsymbol{v} \cdot \boldsymbol{v}\right) dm \qquad \text{– kinetic energy,} \tag{2.91}$$

$$W = \int_V (\boldsymbol{\sigma} : \mathbf{d})\ dV \qquad \text{– stress power,} \tag{2.92}$$

$$P = \int_A (\boldsymbol{v} \cdot \boldsymbol{t})\ dA + \int_V (\boldsymbol{v} \cdot \boldsymbol{f})\ dV \qquad \text{– power of external forces.} \tag{2.93}$$

With the above abbreviations the balance of mechanical energy Eq. (2.90) can be given in a short form as

$$\dot{K} + W = P, \tag{2.94}$$

indicating that the power of external forces goes into the stress power (also referred to as power of internal forces) and the change of the kinetic energy. The stress power can further be decomposed by

$$W = \mathcal{D} + \int_{V_0} \dot{\Psi}\, dV_0 \tag{2.95}$$

where $\mathcal{D}$ denotes the heat production (dissipation) and Ψ is the stored energy density related to the unit volume in the reference configuration.

Work-conjugate Stress-Strain pairs From Eq. (2.90) it is seen that the stress power is expressed by the term $\boldsymbol{\sigma} : \mathbf{d}$, where the rate of deformation tensor $\mathbf{d}$ Eq. (2.63) is not a material time derivative of a strain. Instead of Eq. (2.92), the stress power can be given in terms of a stress $\mathbf{Y}$ and the material time derivative of a strain $\mathbf{Z}$ by

$$W = \int_{V_0} \left(\mathbf{Y} : \dot{\mathbf{Z}}\right) dV_0 = \int_V J^{-1} \left(\mathbf{Y} : \dot{\mathbf{Z}}\right) dV \tag{2.96}$$

The pair of such stress and strain variables is called work-conjugate. It can be shown that

$$\boldsymbol{\sigma} : \mathbf{d} = J^{-1}\mathbf{S} : \frac{1}{2}\dot{\mathbf{C}} = J^{-1}\mathbf{S} : \dot{\mathbf{E}} = J^{-1}\mathbf{P} : \dot{\mathbf{F}} \tag{2.97}$$

where $(\mathbf{S}, \mathbf{E})$, $(\mathbf{S}, \frac{1}{2}\mathbf{C})$ and $(\mathbf{P}, \mathbf{F})$ are work-conjugate.

2.1.4 Hyperelastic Materials

Considering again the stress power defined by Eq. (2.92) and keeping Eq. (2.97) in mind we have

$$W = \int_V (\boldsymbol{\sigma} : \mathbf{d})\, dV = \int_{V_0} \left(\mathbf{S} : \dot{\mathbf{E}}\right) dV_0 = \int_{V_0} \left(\mathbf{P} : \dot{\mathbf{F}}\right) dV_0 \tag{2.98}$$

Generally it is not integrable over time, i.e. there is no scalar function Ψ such that

$$W = \int_{V_0} \dot{\Psi}\, dV_0 \tag{2.99}$$

A material for which such a function $\Psi = \hat{\Psi}(\mathbf{F})$ exists is called *hyperelastic* or Green elastic. The function Ψ is then referred to as elastic potential or strain energy function. It is further seen from Eq. (2.95)

$$\mathcal{D} = 0 \tag{2.100}$$

The elastic potential has to satisfy the material objectivity condition and, hence, it can be shown that

$$\Psi = \hat{\Psi}(\mathbf{U}) = \hat{\Psi}(\mathbf{C}) \tag{2.101}$$

Comparing Eq. (2.99) with Eq. (2.98) and using the chain rule of differentiation one obtains the relations

$$\mathbf{P} = \frac{\partial \Psi}{\partial \mathbf{F}}, \quad \mathbf{S} = \frac{\partial \Psi}{\partial \mathbf{E}} = 2\frac{\partial \Psi}{\partial \mathbf{C}} \tag{2.102}$$

It can be shown that an isotropic tensor function Ψ has the same value for all symmetric tensors with the same eigen values. Thus, this function can uniquely be defined by the eigen values of the tensor argument. The same valid for the principal invariants and principal traces because they can be expressed in terms of the eigen values in the unique form. This implies that for isotropic hyperelastic materials, the tensor function Ψ can be represented in terms of the strain invariants $\mathrm{I}_{\mathbf{C}}, \mathrm{II}_{\mathbf{C}}, \mathrm{III}_{\mathbf{C}}$, principal stretches λ_i or principal traces $\mathrm{tr}\mathbf{C}^i (i = 1, 2, 3)$ by

$$\Psi = \Psi(\mathrm{I}_{\mathbf{C}}, \mathrm{II}_{\mathbf{C}}, \mathrm{III}_{\mathbf{C}}) = \Psi(\lambda_1, \lambda_2, \lambda_3) = \Psi(\mathrm{tr}\mathbf{C}^1, \mathrm{tr}\mathbf{C}^2, \mathrm{tr}\mathbf{C}^3) \tag{2.103}$$

By using different functions Ψ, various isotropic hyperelastic strain energy functions have been proposed. One such class of functions is given by the *Ogden* model

$$\Psi = \sum_{r=1}^{s} \frac{\mu_r}{\alpha_r}(\lambda_1^{\alpha_r} + \lambda_2^{\alpha_r} + \lambda_3^{\alpha_r} - 3), \tag{2.104}$$

where μ_r and $\alpha_r (r = 1, \ldots, s)$ represent material constants.

2.2 Finite Viscoelasticity Theory

In this section a summary of the most important aspects of the Finite Viscoelasticity theory by Reese and Govindjee [8] is presented. This theory is valid for deviations of any size from the thermodynamic equilibrium, i.e. the strain rates do not have to be close to zero. Furthermore it involves a *multiplicative split* of the deformation gradient $\mathbf{F}$ and an *additive split* of the strain energy function Ψ. The evolution equation being similar to Finite Elastoplasticity is integrated using an exponential mapping algorithm, which is linear in the logarithmic principle values of a kinematic measure $\mathbf{b}$. It also leads to a symmetric tangent stiffness modulus due to derivation of the viscous stress (or over-stress) from a potential and usage of an evolution equation that has a special form.

2.2.1 Multiplicative Split of Deformation Gradient

In order to account for both elastic and viscous material behaviour at large deformations, the deformation gradient is split multiplicatively into an elastic and inelastic (or viscous in the case of viscoelasticity) part. For the case of several relaxation mechanisms ($k = 1, 2, \dots, N$) it is given by

$$\mathbf{F} = \mathbf{F}_e{}^k \mathbf{F}_i{}^k \tag{2.105}$$

Using Eq. (2.105) it follows that

$$\begin{aligned}\mathbf{C} = \mathbf{F}^T\mathbf{F} &= (\mathbf{F}_i{}^k\mathbf{F}_e{}^k)^T \mathbf{F}_e{}^k\mathbf{F}_i{}^k \\ &= \mathbf{F}_i{}^{k^T}(\mathbf{F}_e{}^{k^T}\mathbf{F}_e{}^k)\mathbf{F}_i{}^k = \mathbf{F}_i{}^{k^T}\mathbf{C}_e{}^k\mathbf{F}_i{}^k, \quad k = 1, 2, \dots, N\end{aligned} \tag{2.106}$$

and thus

$$\mathbf{C}_e{}^k = \mathbf{F}_e{}^{k^T}\mathbf{F}_e{}^k = \mathbf{F}_i{}^{k^{-T}}\mathbf{C}\mathbf{F}_i{}^{k^{-1}}, \quad k = 1, 2, \dots, N. \tag{2.107}$$

Furthermore the strain energy function is split additively into an equilibrium part (elastic) and a non-equilibrium part (viscous) given by

$$\begin{aligned}\Psi &= \Psi_{\mathrm{EQ}}(\mathbf{C}) + \sum_{k=1}^{N} \Psi_{\mathrm{NEQ}}{}^k(\mathbf{C}_e{}^k) \\ &= \Psi_{\mathrm{EQ}}(\mathbf{C}) + \sum_{k=1}^{N} \Psi_{\mathrm{NEQ}}{}^k(\mathbf{F}_i{}^{k^{-T}}\mathbf{C}^k\mathbf{F}_i{}^{k^{-1}}) \\ &= \Psi(\mathbf{C}, \mathbf{F}_i{}^1, \mathbf{F}_i{}^2, \dots \mathbf{F}_i{}^N)\end{aligned} \tag{2.108}$$

which is a function of the right Cauchy-Green deformation tensor Eq. (2.20) and a set of N internal variables $\mathbf{F}_i{}^k (k = 1, 2, \dots, N)$. In order to determine the internal variables, a set of N evolution equations of the form

$$\dot{\mathbf{F}}_{\mathbf{i}}{}^k = f^k(\mathbf{C}, \mathbf{F}_i{}^1, \mathbf{F}_i{}^2, \dots \mathbf{F}_i{}^N), \quad k = 1, 2, \dots, N. \tag{2.109}$$

2.2.2 Derivation of Evolution Equation

For the sake of ease the simple case of only a single relaxation mechanism ($k = 1$) will be considered in deriving the further equations, wherin

$$\mathbf{F} = \mathbf{F}_e\mathbf{F}_i, \quad \mathbf{C}_e = \mathbf{F}_i{}^{-T}\mathbf{C}\mathbf{F}_i{}^{-1}, \quad \Psi = \Psi(\mathbf{C}, \mathbf{F}_i), \quad \dot{\mathbf{F}}_{\mathbf{i}} = f(\mathbf{C}, \mathbf{F}_i) \tag{2.110}$$

However, these results can be extended to incorporate several relaxation mechanisms ($k \geq 1$).

According to the second law of thermodynamics in every real process

$$\mathcal{D} \geq 0 \tag{2.111}$$

where $\mathcal{D}$ denotes the dissipation rate. By means of Eq. (2.95) and Eq. (2.97) this inequality can be expressed by

$$\mathbf{S} : \frac{1}{2}\dot{\mathbf{C}} - \dot{\Psi} \geq 0 \tag{2.112}$$

where

$$\mathbf{S} = \mathbf{S}(\mathbf{C}, \mathbf{F}_i) \tag{2.113}$$

is the second Piola-Kirchoff stress tensor. By using Eq. (2.110), Eq. (2.112) and identities for derivatives of tensor functions [refer Eq. 1.149 4] we have

$$\mathbf{S} : \frac{1}{2}\dot{\mathbf{C}} - \frac{\partial \Psi_{\mathrm{EQ}}}{\partial \mathbf{C}} : \dot{\mathbf{C}} - \frac{\partial \Psi_{\mathrm{NEQ}}}{\partial \mathbf{C}_e} : \frac{\partial \mathbf{C}_e}{\partial \mathbf{F}_i} : \dot{\mathbf{F}}_{\mathbf{i}} - \frac{\partial \Psi_{\mathrm{NEQ}}}{\partial \mathbf{C}_e} : \frac{\partial \mathbf{C}_e}{\partial \mathbf{C}} : \dot{\mathbf{C}} \geq 0 \tag{2.114}$$

where

$$\frac{\partial \mathbf{C}_e}{\partial \mathbf{C}} = \mathbf{F}_i^{-T} \overset{4}{\mathbf{I}} \mathbf{F}_i^{-1} \tag{2.115}$$

After rearranging we have

$$\left(\mathbf{S} - 2\frac{\partial \Psi_{\mathrm{EQ}}}{\partial \mathbf{C}} - 2\mathbf{F}_i^{-1} \frac{\partial \Psi_{\mathrm{NEQ}}}{\partial \mathbf{C}_e} \mathbf{F}_i^{-T} \right) : \frac{1}{2}\dot{\mathbf{C}} - \frac{\partial \Psi_{\mathrm{NEQ}}}{\partial \mathbf{C}_e} : \frac{\partial \mathbf{C}_e}{\partial \mathbf{F}_i} : \dot{\mathbf{F}}_{\mathbf{i}} \geq 0 \tag{2.116}$$

This inequality holds for all values of $\dot{\mathbf{C}}$ and $\dot{\mathbf{F}}_{\mathbf{i}}$, which are independent of each other. This leads to the constitutive equation

$$\mathbf{S} = \mathbf{S}_{\mathrm{EQ}} + \mathbf{S}_{\mathrm{NEQ}} = \underbrace{2\frac{\partial \Psi_{\mathrm{EQ}}}{\partial \mathbf{C}}}_{\mathbf{S}_{\mathrm{EQ}}} + \underbrace{2\mathbf{F}_i^{-1} \frac{\partial \Psi_{\mathrm{NEQ}}}{\partial \mathbf{C}_e} \mathbf{F}_i^{-T}}_{\mathbf{S}_{\mathrm{NEQ}}} = 2\frac{\partial \Psi}{\partial \mathbf{C}} \tag{2.117}$$

and the residual inequality

$$\frac{\partial \Psi_{\mathrm{NEQ}}}{\partial \mathbf{C}_e} : \frac{\partial \mathbf{C}_e}{\partial \mathbf{F}_i} : \dot{\mathbf{F}}_{\mathbf{i}} \geq 0 \tag{2.118}$$

which after simplification leads to

$$\frac{\partial \Psi_{\mathrm{NEQ}}}{\partial \mathbf{C}_e} : (\mathbf{l}_i{}^T \mathbf{C}_e + \mathbf{C}_e \mathbf{l}_i) \geq 0 \tag{2.119}$$

and by exploiting symmetry properties we have

$$2\frac{\partial \Psi_{\mathrm{NEQ}}}{\partial \mathbf{C}_e} : (\mathbf{C}_e \mathbf{l}_i) \geq 0 \tag{2.120}$$

where $\mathbf{l}_i = \dot{\mathbf{F}}_\mathbf{i} \mathbf{F}_i^{-1}$ is termed the inelastic part of the spatial velocity gradient $\mathbf{l} = \dot{\mathbf{F}} \mathbf{F}^{-1}$. Furthermore, rewriting Eq. (2.120) in the form

$$2\mathbf{F}_e \, \frac{\partial \Psi_{\text{NEQ}}}{\partial \mathbf{C}_e} \, \mathbf{F}_e^T : (\mathbf{F}_e^{-T} \mathbf{C}_e \, \mathbf{l}_i \, \mathbf{F}_e^{-1}) \geq 0 \tag{2.121}$$

leads to

$$\boldsymbol{\tau}_{\text{NEQ}} \, \mathbf{b}_e^{-1} : (\mathbf{F}_e \, \mathbf{l}_i \, \mathbf{F}_e^T) \geq 0 \tag{2.122}$$

where the relations

$$\boldsymbol{\tau}_{\text{NEQ}} = \mathbf{F}_e \, \mathbf{S}_{\text{NEQ}} \, \mathbf{F}_e^T = 2\mathbf{F}_e \, \frac{\partial \Psi_{\text{NEQ}}}{\partial \mathbf{C}_e} \, \mathbf{F}_e^T \quad \text{and} \quad \mathbf{b}_e = \mathbf{F}_e \mathbf{F}_e^T \tag{2.123}$$

have been used keeping Eq. (2.70), Eq. (2.72) Eq. (2.117) and Eq. (2.20) in mind.

The term $\mathbf{F}_e \, \mathbf{l}_i \, \mathbf{F}_e^T$ can be split into a symmetric and skew-symmetric tensor as

$$\mathbf{F}_e \, \mathbf{l}_i \, \mathbf{F}_e^T = \text{sym}(\mathbf{F}_e \, \mathbf{l}_i \, \mathbf{F}_e^T) + \text{skew}(\mathbf{F}_e \, \mathbf{l}_i \, \mathbf{F}_e^T) \tag{2.124}$$

Considering the symmetric part, by virtue of Eq. (2.105) and Eq. (2.63)

$$\begin{aligned}
\text{sym}(\mathbf{F}_e \, \mathbf{l}_i \, \mathbf{F}_e^T) &= \frac{1}{2} \left(\mathbf{F}_e \, \mathbf{l}_i \, \mathbf{F}_e^T + (\mathbf{F}_e \, \mathbf{l}_i \, \mathbf{F}_e^T)^T \right) = \frac{1}{2} \left(\mathbf{F}_e \, (\mathbf{l}_i + \mathbf{l}_i) \, \mathbf{F}_e^T \right) \\
&= \frac{1}{2} \left(\mathbf{F} \mathbf{F}_i^{-1} \, (2\mathbf{d}_i) \, \mathbf{F}_i^{-T} \mathbf{F}^T \right) = \frac{1}{2} \left(\mathbf{F} \mathbf{F}_i^{-1} \left(\mathbf{F}_i^{-T} \dot{\mathbf{C}}_\mathbf{i} \mathbf{F}_i^{-1} \right) \mathbf{F}_i^{-T} \mathbf{F}^T \right) \\
&= \frac{1}{2} \left(\mathbf{F} (\mathbf{F}_i^{-1} \mathbf{F}_i^{-T}) \dot{\mathbf{C}}_\mathbf{i} (\mathbf{F}_i^{-1} \mathbf{F}_i^{-T}) \mathbf{F}^T \right) = \frac{1}{2} \left(\mathbf{F} \mathbf{C}_i^{-1} \dot{\mathbf{C}}_\mathbf{i} \mathbf{C}_i^{-1} \mathbf{F}^T \right) \\
\text{sym}(\mathbf{F}_e \, \mathbf{l}_i \, \mathbf{F}_e^T) &= -\frac{1}{2} \left(\mathbf{F} \overline{(\mathbf{C}_\mathbf{i}^{-\mathbf{1}})}^{\boldsymbol{\cdot}} \mathbf{F}^T \right)
\end{aligned} \tag{2.125}$$

where

$$\mathbf{C}_i^{-1} \dot{\mathbf{C}}_\mathbf{i} \mathbf{C}_i^{-1} = -\overline{(\mathbf{C}_\mathbf{i}^{-\mathbf{1}})}^{\boldsymbol{\cdot}} \tag{2.126}$$

can be proved by taking the time derivative of the identitiy $\mathbf{C}_i \mathbf{C}_i^{-1} = \mathbf{I}$. Furthermore it can be shown that the inelastic right Cauchy-Green tensor

$$\mathbf{C}_i = \mathbf{F}_i^T \mathbf{F}_i = (\mathbf{F}_e^{-1} \mathbf{F})^T (\mathbf{F}_e^{-1} \mathbf{F}) = \mathbf{F}^T (\mathbf{F}_e^{-T} \mathbf{F}_e^{-1}) \mathbf{F} = \mathbf{F}^T \mathbf{b}_e^{-1} \mathbf{F} \tag{2.127}$$

and hence

$$\mathbf{b}_e = \mathbf{F} \mathbf{C}_i^{-1} \mathbf{F}^T \tag{2.128}$$

The Eq. (2.124) thus reduces to

$$\mathbf{F}_e \, \mathbf{l}_i \, \mathbf{F}_e^T = -\frac{1}{2} \underbrace{\left(\mathbf{F} \overline{(\mathbf{C}_\mathbf{i}^{-\mathbf{1}})}^{\boldsymbol{\cdot}} \mathbf{F}^T \right)}_{\mathcal{L}_v \mathbf{b}_e} + \text{skew}(\mathbf{F}_e \, \mathbf{l}_i \, \mathbf{F}_e^T) \tag{2.129}$$

Here the notation $\mathcal{L}_v \mathbf{b}_e$ stands for the Lie derivative of the contravariant tensor $\mathbf{b}_e$. The Lie derivative has the meaning of pushing back a tensor quantity into the reference configuration, taking the time derivative and then pushing it forward into the current configuration. For a contravariant tensor $\mathbf{F}^{-1}(\bullet)\mathbf{F}^{-T}$ represents the pull back transformation and $\mathbf{F}(\bullet)\mathbf{F}^{T}$ represents the push forward transformation. Indeed we have by virtue of Eq. (2.128)

$$\mathcal{L}_v \mathbf{b}_e = \mathbf{F}\,\dot{\overline{(\mathbf{F^{-1}(b_e)F^{-T}})}}\,\mathbf{F}^T = \mathbf{F}\,\dot{\overline{(\mathbf{C_i^{-1}})}}\,\mathbf{F}^T \tag{2.130}$$

Now assuming isotropy of the material and consequently Ψ_{NEQ} to be an isotropic tensor function. Therefore we may write $\Psi_{\text{NEQ}}(\mathbf{b}_e)$ because both $\mathbf{C}_e$ and $\mathbf{b}_e$ are symmetric have the same eigen values. In the case $\boldsymbol{\tau}_{\text{NEQ}}$ and $\mathbf{b}_e$ commute, which gives symmetry of the tensor $\boldsymbol{\tau}_{\text{NEQ}}\,\mathbf{b}_e^{-1}$. Thus in Eq. (2.122) only the symmetric part of $\mathbf{F}_e\,\mathbf{l}_i\,\mathbf{F}_e^T$ plays a role and gives rise to the (isotropic) inequality

$$-\boldsymbol{\tau}_{\text{NEQ}}\,\mathbf{b}_e^{-1} : \frac{1}{2}(\mathcal{L}_v \mathbf{b}_e)\mathbf{b}_e^{-1} \geq 0 \tag{2.131}$$

In order to fulfil Eq. (2.131) the evolution equation is chosen to be of the form

$$-\frac{1}{2}(\mathcal{L}_v \mathbf{b}_e)\mathbf{b}_e^{-1} = \mathcal{V}^{-1} : \boldsymbol{\tau}_{\text{NEQ}} \tag{2.132}$$

where $\mathcal{V}^{-1} = \mathcal{V}^{-1}(\mathbf{b}_e)$ is an isotropic rank four tensor which has to be positive definite and is given by

$$\mathcal{V}^{-1} = \frac{1}{2\eta_{\text{dev}}}\underbrace{\left(\overset{4}{\mathbf{I}} - \frac{1}{3}\mathbf{I}\otimes\mathbf{I}\right)}_{\substack{\text{Deviatoric}\\ \text{Map}}} + \frac{1}{3\eta_{\text{vol}}}\underbrace{\left(\frac{1}{3}\mathbf{I}\otimes\mathbf{I}\right)}_{\substack{\text{Volumetric}\\ \text{Map}}} \tag{2.133}$$

Here, $\eta_{\text{dev}} > 0$ and $\eta_{\text{vol}} > 0$ represent the deviatoric and volumetric viscosities respectively and could possibly, but not necessarily, be deformation dependent. If Eq. (2.133) is inserted into Eq. (2.132), we finally obtain the evolution equation as

$$-(\mathcal{L}_v \mathbf{b}_e)\mathbf{b}_e^{-1} = \frac{1}{\eta_{\text{dev}}}\text{dev}(\boldsymbol{\tau}_{\text{NEQ}}) + \frac{2}{3\eta_{\text{vol}}}\text{vol}(\boldsymbol{\tau}_{\text{NEQ}}) \tag{2.134}$$

where

$$\text{vol}(\boldsymbol{\tau}_{\text{NEQ}}) = \frac{1}{3}\text{tr}(\boldsymbol{\tau}_{\text{NEQ}})\,\mathbf{I} \quad \text{and} \quad \text{dev}(\boldsymbol{\tau}_{\text{NEQ}}) = \boldsymbol{\tau}_{\text{NEQ}} - \text{vol}(\boldsymbol{\tau}_{\text{NEQ}}) \tag{2.135}$$

2.2.3 Integration of the Evolution Equation

The main objective in this section is to determine in an efficient manner the value of the governing inelastic internal variable

$$\begin{aligned}
\mathbf{b}_e(\mathbf{C},\mathbf{F}_i) &= \mathbf{F}_e\mathbf{F}_e^T = (\mathbf{F}\mathbf{F}_i^{-1})(\mathbf{F}\mathbf{F}_i^{-1})^T = \mathbf{F}(\mathbf{F}_i^{-1}\mathbf{F}_i^{-T})\mathbf{F}^T \\
&= \mathbf{F}(\mathbf{C}_i^{-1})\mathbf{F}^T \\
&= (\mathbf{F}^{-T}\mathbf{C})\mathbf{C}_i^{-1}(\mathbf{C}\mathbf{F}^{-1}) \\
\mathbf{b}_e(\mathbf{C},\mathbf{F}_i) &= \mathbf{F}^{-T}(\mathbf{C}\mathbf{F}_i^{-1}\mathbf{F}_i^{-T}\mathbf{C})\mathbf{F}^{-1}
\end{aligned} \tag{2.136}$$

for an advancement of time in order that the stresses may be evaluated when the material motion $\mathbf{F}$ is known, which is a typical setting in a finite element program. For this, an operator split of the material time derivative of the $\mathbf{b}_e$ is carried into an elastic predictor E and an inelastic correcter I given by

$$\dot{\mathbf{b}_\mathbf{e}} = \overline{\mathbf{F}\,\dot{\mathbf{C}_\mathbf{i}^{-1}}\,\mathbf{F^T}} = \underbrace{\mathbf{l}\,\mathbf{b}_e + \mathbf{b}_e\,\mathbf{l}^T}_{E} + \underbrace{\mathbf{F}\overline{(\dot{\mathbf{C}_\mathbf{i}^{-1}})}\mathbf{F}^T}_{I} \tag{2.137}$$

First, in the elastic predictor step the material time derivative of $\mathbf{C}_i^{-1}$ is set to zero and we have

$$(\mathbf{C}_i^{-1})_{\text{trial}} = (\mathbf{C}_i^{-1})_{t=t_{n-1}} \rightarrow (\mathbf{b}_e)_{\text{trial}} = (\mathbf{F})_{t=t_n}\,(\mathbf{C}_i^{-1})_{t=t_{n-1}}\,(\mathbf{F}^T)_{t=t_n} \tag{2.138}$$

Then in the elastic corrector step, the spatial velocity gradient is set to zero, which leads to $\mathcal{L}_v\mathbf{b}_e = \dot{\mathbf{b}_\mathbf{e}}$ and finally to

$$\dot{\mathbf{b}_\mathbf{e}} = -(2\mathcal{V}^{-1} : \boldsymbol{\tau}_{\text{NEQ}})\,\mathbf{b}_e \tag{2.139}$$

Solving the above differential equation using the exponential mapping algorithm gives

$$\mathbf{b}_e = \exp\left(-2\int_{t_{n-1}}^{t}\mathcal{V}^{-1} : \boldsymbol{\tau}_{\text{NEQ}}\,\mathrm{d}t\right)(\mathbf{b}_e)_{\text{trial}} \tag{2.140}$$

or by a first order accurate approximation gives

$$\mathbf{b}_e \approxeq \exp\left(-2\underbrace{(t_n - t_{n-1})}_{\Delta t}(\mathcal{V}^{-1} : \boldsymbol{\tau}_{\text{NEQ}})_{t=t_n}\right)(\mathbf{b}_e)_{\text{trial}} \tag{2.141}$$

Due to isotropy, $\boldsymbol{\tau}_{\text{NEQ}}$ commutes with $\mathbf{b}_e$ and also with $(\mathbf{b}_e)_{\text{trial}}$. Since $\mathcal{V}^{-1}$ is assumed to be isotropic, by virtue of Eq. (2.132) and Eq. (2.134) we can furthur write Eq. (2.141) as

$$\lambda_{i_e}^2 = \exp\left(-\Delta t\left[\frac{1}{\eta_{\text{dev}}}\text{dev}(\tau_{\text{NEQ}})_i + \frac{2}{9\eta_{\text{vol}}}\text{tr}(\boldsymbol{\tau}_{\text{NEQ}})\right]\right)(\lambda_{i_e}^2)_{\text{trial}} \tag{2.142}$$

where $\operatorname{dev}(\boldsymbol{\tau}_{\text{NEQ}})_i$ are the principal values of $\operatorname{dev}(\boldsymbol{\tau}_{\text{NEQ}})$, $\lambda_{i_e}^2$ the principal values of $(\mathbf{b}_e)_{t=t_n}$ and $(\lambda_{i_e}^2)_{\text{trial}}$ the principal values of $(\mathbf{b}_e)_{\text{trial}}$. Furthermore taking the logarithm of both sides we have

$$\varepsilon_{i_e} = -\Delta t \left(\frac{1}{2\eta_{\text{dev}}} \operatorname{dev}(\tau_{\text{NEQ}})_i + \frac{1}{9\eta_{\text{vol}}} \operatorname{tr}(\boldsymbol{\tau}_{\text{NEQ}}) \right) + (\varepsilon_{i_e})_{\text{trial}} \tag{2.143}$$

where $\varepsilon_{i_e} = \ln \lambda_{i_e}$ are the elastic logarithmic principal stretches.

Next, due to isotropy, we can consider the non-equilibrium part of the strain energy Ψ_{NEQ} as a function of the principal values $b_{i_e} = \lambda_{i_e}^2$ of $\mathbf{b}_e$ and we thus have

$$\Psi_{\text{NEQ}} = \Psi_{\text{NEQ}}(b_{1_e}, b_{2_e}, b_{3_e}) \tag{2.144}$$

which is futher split into volumetric and deviatoric parts given by

$$\Psi_{\text{NEQ}} = (\Psi_{\text{NEQ}})_D(\bar{b}_{1_e}, \bar{b}_{2_e}, \bar{b}_{3_e}) + (\Psi_{\text{NEQ}})_V(J_e) \tag{2.145}$$

where

$$J_e = \lambda_{1_e}\lambda_{2_e}\lambda_{3_e} \tag{2.146}$$

denotes the determinant of $\mathbf{F}_e$ and

$$\bar{b}_{i_e} = J_e^{-2/3} b_{i_e} = J_e^{-2/3} \lambda_{i_e}^2 \tag{2.147}$$

the principal values of

$$\overline{\mathbf{b}}_e = J_e^{-2/3}\, \mathbf{b}_e \tag{2.148}$$

For the strain energy function Ψ_{NEQ} the Ogden class of models is used which after a split into deviatoric and volumetric parts is given by

$$\Psi_{\text{NEQ}} = \sum_{r=1}^{N} \frac{(\mu_v)_r}{(\alpha_v)_r} \left(\bar{b}_{1_e}^{(\alpha_v)_r/2} + \bar{b}_{2_e}^{(\alpha_v)_r/2} + \bar{b}_{3_e}^{(\alpha_v)_r/2} \right) + \frac{\mathrm{K}_v}{4} \left(J_e^2 - 2\ln J_e - 1 \right) \tag{2.149}$$

where $(r = 1, \ldots, N)$ is the order of the Ogden model, μ_v and α_v are elasticity constants, such that for the shear modulus the relationship

$$\mu_{vis} = \sum_{r=1}^{N} \frac{1}{2} (\mu_v)_r (\alpha_v)_r \tag{2.150}$$

holds. Furthermore, the non-equilibrium part of the Kirchoff stress tensor $\boldsymbol{\tau}_{\text{NEQ}}$ is then given by

$$\boldsymbol{\tau}_{\text{NEQ}} = 2 \frac{\partial \Psi_{\text{NEQ}}}{\partial \mathbf{b}_e}\, \mathbf{b}_e = \sum_{i=1}^{3} \tau_{\text{NEQ}_i}\, \boldsymbol{n}_i \otimes \boldsymbol{n}_i \tag{2.151}$$

The principal Kirchoff stresses are calculated by

$$\tau_{\mathrm{NEQ}_i} = \mathrm{dev}(\tau_{\mathrm{NEQ}})_i + \frac{1}{3}\mathrm{tr}(\boldsymbol{\tau}_{\mathrm{NEQ}}) \tag{2.152}$$

where

$$\mathrm{dev}(\boldsymbol{\tau}_{\mathrm{NEQ}})_i = \sum_{r=1}^{N}(\mu_v)_r \left(\frac{2}{3}\,\bar{b}_{i_e}^{(\alpha_v)_r/2} - \frac{1}{3}\,\bar{b}_{j_e}^{(\alpha_v)_r/2} - \frac{1}{3}\,\bar{b}_{k_e}^{(\alpha_v)_r/2}\right) \quad i \neq j \neq k \tag{2.153}$$

and

$$\frac{1}{3}\mathrm{tr}(\boldsymbol{\tau}_{\mathrm{NEQ}}) = \mathrm{vol}(\tau_{\mathrm{NEQ}})_i = \frac{\mathrm{K}_v}{4}(J_e^2 - 1) \tag{2.154}$$

We thus have that $\boldsymbol{\tau}_{\mathrm{NEQ}_i} = \boldsymbol{\tau}_{\mathrm{NEQ}_i}(\varepsilon_{i_e})$. Thus with $\lambda_{i_e} = \exp(\varepsilon_{i_e})$, it is evident that Eq. (2.143) is a non-linear equation in the elastic logarithmic strains ε_{i_e} which can be solved using the Newton-Raphson's method. The steps for the Newton-Raphson iterations are given below with $i, j, k = 1, 2, 3$ and m is the current iteration number

1. Develop the residual from the non-linear equation in Eq. (2.143)

$$r_i = \varepsilon_{i_e} + \Delta t\left(\frac{1}{2\eta_{\mathrm{dev}}}\mathrm{dev}(\boldsymbol{\tau}_{\mathrm{NEQ}})_i + \frac{1}{9\eta_{\mathrm{vol}}}\mathrm{tr}(\boldsymbol{\tau}_{\mathrm{NEQ}})\right) - (\varepsilon_{i_e})_{\mathrm{trial}} = 0 \tag{2.155}$$

2. Linearize the residual Eq. (2.155) around the logarithmic principal strain $\varepsilon_{i_e} = (\varepsilon_{i_e})_m$ for the current iteration

$$r_i \approxeq \underbrace{r_i\big|_{(\varepsilon_{i_e})_m}}_{(r_i)_m} + \underbrace{\left.\frac{\partial r_i}{\partial \varepsilon_{j_e}}\right|_{(\varepsilon_{i_e})_m}}_{(K_{ij})_m}(\Delta\varepsilon_{j_e})_m = 0 \tag{2.156}$$

3. Using Eq. (2.156) solve for $(\Delta\varepsilon_{i_e})_m$

$$(K_{ij})_m(\Delta\varepsilon_{i_e})_m = -(r_i)_m \tag{2.157}$$

where

$$(K_{ij})_m = \delta_{ij} + \Delta t\frac{1}{2\eta_{\mathrm{dev}}}\frac{\partial(\mathrm{dev}(\tau_{\mathrm{NEQ}})_i)}{\partial\varepsilon_{j_e}} - \Delta t\frac{1}{3\eta_{\mathrm{vol}}}\frac{\partial(\frac{1}{3}\mathrm{tr}(\boldsymbol{\tau}_{\mathrm{NEQ}}))}{\partial\varepsilon_{j_e}} \tag{2.158}$$

and with $(i \neq j \neq k)$

$$\frac{\partial(\mathrm{dev}(\boldsymbol{\tau}_{\mathrm{NEQ}})_i)}{\partial \varepsilon_{i_e}} = \sum_{r=1}^{N} (\mu_v)_r (\alpha_v)_r \left(\frac{4}{9} \bar{b}_{i_e}^{(\alpha_v)_r/2} + \frac{1}{9} \bar{b}_{j_e}^{(\alpha_v)_r/2} + \frac{1}{9} \bar{b}_{k_e}^{(\alpha_v)_r/2} \right) \tag{2.159}$$

$$\frac{\partial(\mathrm{dev}(\boldsymbol{\tau}_{\mathrm{NEQ}})_i)}{\partial \varepsilon_{j_e}} = \sum_{r=1}^{N} (\mu_v)_r (\alpha_v)_r \left(-\frac{2}{9} \bar{b}_{i_e}^{(\alpha_v)_r/2} - \frac{2}{9} \bar{b}_{j_e}^{(\alpha_v)_r/2} + \frac{1}{9} \bar{b}_{k_e}^{(\alpha_v)_r/2} \right) \tag{2.160}$$

$$\frac{\partial(\frac{1}{3}\mathrm{tr}(\boldsymbol{\tau}_{\mathrm{NEQ}}))}{\partial \varepsilon_{j_e}} = \mathrm{K}_v J_e^2 \tag{2.161}$$

4. Update the logarithmic principal strain for the next iteration and increment the iteration number

$$(\varepsilon_{i_e})_{m+1} = (\varepsilon_{i_e})_m + (\Delta\varepsilon_{i_e})_m \tag{2.162}$$

and

$$m = m + 1 \tag{2.163}$$

5. Repeat steps 1.-4. until the norm of the residual

$$||\boldsymbol{r}|| = \sqrt{\sum_{i=1}^{3} (r_i)^2} \tag{2.164}$$

is less than the specified tolerance, i.e $||\boldsymbol{r}|| <$ tolerance

Keeping isotropy in mind $\mathbf{b}_e$ and $\boldsymbol{\tau}_{\mathrm{NEQ}}$ have the same eigen vectors as $(\mathbf{b}_e)_{\mathrm{trial}}$, i.e $\boldsymbol{n}_i = (\boldsymbol{n}_i)_{\mathrm{trial}}$. Thus after convergence of the Newton-Raphsons iteration we can calculate

$$\mathbf{b}_e = \sum_{i=1}^{3} \lambda_{i_e}^2 \boldsymbol{n}_i \otimes \boldsymbol{n}_i = \sum_{i=1}^{3} \lambda_{i_e}^2 (\boldsymbol{n}_i)_{\mathrm{trial}} \otimes (\boldsymbol{n}_i)_{\mathrm{trial}} \tag{2.165}$$

and

$$\boldsymbol{\tau}_{\mathrm{NEQ}} = \sum_{i=1}^{3} \tau_{\mathrm{NEQ}_i} \boldsymbol{n}_i \otimes \boldsymbol{n}_i = \sum_{i=1}^{3} \tau_{\mathrm{NEQ}_i} (\boldsymbol{n}_i)_{\mathrm{trial}} \otimes (\boldsymbol{n}_i)_{\mathrm{trial}} \tag{2.166}$$

where $(\boldsymbol{\tau}_{\mathrm{NEQ}})_i$ is given by Eq. (2.152), Eq. (2.153), Eq. (2.154) and Eq. (2.147)

2.2.4 Solution of Weak Formulation: FE Procedure

In a finite element program, the balance of mechanical energy Eq. (2.90) is solved in the weak form. In such a formulation, it is required to caclulate the stresses $\boldsymbol{\tau}$ and the spatial tangent modulus $\overset{4}{\mathscr{C}}$ given by the sum of their equilibrium and non-equilibrium parts as

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{\mathrm{EQ}} + \boldsymbol{\tau}_{\mathrm{NEQ}} \tag{2.167}$$

$$\overset{4}{\mathscr{C}} = \overset{4}{\mathscr{C}}_{\mathrm{EQ}} + \overset{4}{\mathscr{C}}_{\mathrm{NEQ}} \tag{2.168}$$

where $\overset{4}{\mathscr{C}}$ is also called the 4-th order constitutive tensor. It can be calculated by pushing forward the material tangent modulus $\overset{4}{\mathscr{L}}$ with $\mathbf{F}$ where

$$\overset{4}{\mathscr{L}} = \overset{4}{\mathscr{L}}_{\mathrm{EQ}} + \overset{4}{\mathscr{L}}_{\mathrm{NEQ}} \tag{2.169}$$

and keeping Eq. (2.102)

$$\overset{4}{\mathscr{L}} = \frac{\partial \mathbf{S}}{\partial \mathbf{E}} = \frac{\partial^2 \Psi}{\partial \mathbf{E}\, \partial \mathbf{E}} \tag{2.170}$$

After integration of the evolution equation, $\boldsymbol{\tau}_{\mathrm{NEQ}}$ Eq. (2.166) is already calculated. Thus it remains to calculate $\boldsymbol{\tau}_{\mathrm{EQ}}$, $\overset{4}{\mathscr{C}}_{\mathrm{EQ}}$ and $\overset{4}{\mathscr{C}}_{\mathrm{NEQ}}$

Calculation of Equilibrium Kirchoff Stress Tensor Let us again consider a strain energy function according to the Ogden model similar to that in Eq. (2.149) for the equilibrium part Ψ_{EQ} of Ψ given by

$$\Psi_{\mathrm{EQ}} = \sum_{r=1}^{N} \frac{(\mu)_r}{(\alpha)_r} \left(\bar{b}_1^{(\alpha)_r/2} + \bar{b}_2^{(\alpha)_r/2} + \bar{b}_3^{(\alpha)_r/2} \right) + \frac{\mathrm{K}}{4} \left(J^2 - 2 \ln J - 1 \right) \tag{2.171}$$

where

$$J = \lambda_1 \lambda_2 \lambda_3 \tag{2.172}$$

denotes the determinant of $\mathbf{F}$ and

$$\bar{b}_i = J^{-2/3} b_i = J^{-2/3} \lambda_i^2 \tag{2.173}$$

the principal values of

$$\bar{\mathbf{b}} = J^{-2/3}\, \mathbf{b} = \sum_{i=1}^{3} J^{-2/3} b_i\, \boldsymbol{n}_i \otimes \boldsymbol{n}_i \tag{2.174}$$

The principal values of the equilibrium Kirchoff stress tensor

$$\boldsymbol{\tau}_{\mathrm{EQ}} = \sum_{i=1}^{3} \tau_{\mathrm{EQ}_i}\, \boldsymbol{n}_i \otimes \boldsymbol{n}_i \tag{2.175}$$

considering isotropic elasticity [11, see Sec 6.2.3] are given by

$$\begin{aligned}\boldsymbol{\tau}_{\mathrm{EQ}_i} &= \frac{\partial \Psi_{\mathrm{EQ}}}{\partial \varepsilon_i} \qquad (2.176)\\ &= \frac{\partial \Psi_{\mathrm{EQ}}}{\partial \bar{b}_k}\frac{\partial \bar{b}_k}{\partial \lambda_j}\frac{\partial \lambda_j}{\partial \varepsilon_i} + \frac{\partial \Psi_{\mathrm{EQ}}}{\partial J}\frac{\partial J}{\partial \lambda_j}\frac{\partial \lambda_j}{\partial \varepsilon_i}\\ \boldsymbol{\tau}_{\mathrm{EQ}_i} &= \underbrace{\frac{\partial \Psi_{\mathrm{EQ}}}{\partial \bar{b}_j}\frac{\partial \bar{b}_j}{\partial \lambda_i}\lambda_i}_{\mathrm{dev}(\boldsymbol{\tau}_{\mathrm{EQ}})_i} + \underbrace{\frac{\partial \Psi_{\mathrm{EQ}}}{\partial J}J}_{\mathrm{vol}(\boldsymbol{\tau}_{\mathrm{EQ}})_i} \qquad (2.177)\end{aligned}$$

where

$$\mathrm{dev}(\boldsymbol{\tau}_{\mathrm{EQ}})_i = \sum_{r=1}^{N} (\mu)_r \left(\frac{2}{3}\,\bar{b}_i^{(\alpha)_r/2} - \frac{1}{3}\,\bar{b}_j^{(\alpha)_r/2} - \frac{1}{3}\,\bar{b}_k^{(\alpha)_r/2}\right) \quad i \neq j \neq k \tag{2.178}$$

and

$$\mathrm{vol}(\boldsymbol{\tau}_{\mathrm{EQ}})_i = \frac{1}{3}\mathrm{tr}(\boldsymbol{\tau}_{\mathrm{EQ}}) = \frac{\mathrm{K}}{4}(J^2 - 1) \tag{2.179}$$

Calculation of Equilibrium Spatial Tangent Stiffness Modulus Once the principal values of the equilibrium Kirchoff stress tensor are known, it remains to calculate the equilibrium spatial tangent stiffness modulus [11, Eq 6.50] given by

$$\begin{aligned}J\overset{4}{\mathscr{C}}_{\mathrm{EQ}} &= \sum_{i=1}^{3}\sum_{j=1}^{3} \left(c_{ij} - 2\tau_{\mathrm{EQ}_i}\delta_{ij}\right) \boldsymbol{n}_i \otimes \boldsymbol{n}_i \otimes \boldsymbol{n}_j \otimes \boldsymbol{n}_j\\ &\quad + \frac{1}{2}\sum_{i=1}^{3}\sum_{j\neq i=1}^{3} (g_{ij})\; [\boldsymbol{n}_i \otimes \boldsymbol{n}_j \otimes \boldsymbol{n}_i \otimes \boldsymbol{n}_j + \boldsymbol{n}_i \otimes \boldsymbol{n}_j \otimes \boldsymbol{n}_j \otimes \boldsymbol{n}_i] \qquad (2.180)\end{aligned}$$

In the above equation, keeping Eq. (2.176) in mind we have

$$\begin{aligned}c_{ij} &= \frac{\partial^2 \Psi_{\mathrm{EQ}}}{\partial \varepsilon_i\, \partial \varepsilon_j} = \frac{\partial}{\partial \varepsilon_i}\frac{\partial \Psi_{\mathrm{EQ}}}{\partial \varepsilon_j} = \frac{\partial \tau_{\mathrm{EQ}_j}}{\partial \varepsilon_i}\\ c_{ij} &= \frac{\partial(\mathrm{dev}(\tau_{\mathrm{EQ}})_j)}{\partial \varepsilon_i} + \frac{\partial(\mathrm{vol}(\tau_{\mathrm{EQ}})_i)}{\partial \varepsilon_i} \qquad (2.181)\end{aligned}$$

where similar to Eq. (2.159), Eq. (2.160), Eq. (2.184), we have

$$\frac{\partial(\mathrm{dev}(\tau_{\mathrm{EQ}})_i)}{\partial\varepsilon_i} = \sum_{r=1}^{N}(\mu)_r(\alpha)_r\left(\frac{4}{9}\,\bar{b}_i^{(\alpha)_r/2} + \frac{1}{9}\,\bar{b}_j^{(\alpha)_r/2} + \frac{1}{9}\,\bar{b}_k^{(\alpha)_r/2}\right) \tag{2.182}$$

$$\frac{\partial(\mathrm{dev}(\tau_{\mathrm{EQ}})_i)}{\partial\varepsilon_j} = \sum_{r=1}^{N}(\mu)_r(\alpha)_r\left(-\frac{2}{9}\,\bar{b}_i^{(\alpha)_r/2} - \frac{2}{9}\,\bar{b}_j^{(\alpha)_r/2} + \frac{1}{9}\,\bar{b}_k^{(\alpha)_r/2}\right) \tag{2.183}$$

$$\frac{\partial(\mathrm{vol}(\tau_{\mathrm{EQ}})_i)}{\partial\varepsilon_j} = \frac{\partial(\frac{1}{3}\mathrm{tr}(\boldsymbol{\tau}_{\mathrm{EQ}}))}{\partial\varepsilon_j} = \mathrm{K}J^2 \quad i,j = 1,2,3 \tag{2.184}$$

with $k \neq j \neq i = 1,2,3$ in Eq. (2.159) Eq. (2.160) and

$$g_{ij} = \begin{cases} \dfrac{\tau_{\mathrm{EQ}_i}\lambda_j^2 - \tau_{\mathrm{EQ}_j}\lambda_i^2}{\lambda_i^2 - \lambda_j^2}, & \text{if } \lambda_i \neq \lambda_j \\ \dfrac{\partial(\tau_{\mathrm{EQ}_i} - \tau_{\mathrm{EQ}_j})}{\partial\varepsilon_i}, & \text{if } \lambda_i = \lambda_j \end{cases} \qquad i,j = 1,2,3 \tag{2.185}$$

For the second case where the equilibrium principal stretches are equal, considering Eq. (2.177), Eq. (2.178) and Eq. (2.179), we first evaluate

$$\tau_{\mathrm{EQ}_i} - \tau_{\mathrm{EQ}_j} = \sum_{r=1}^{N}(\mu)_r\left(\bar{b}_i^{(\alpha)_r/2} - \bar{b}_j^{(\alpha)_r/2}\right) \tag{2.186}$$

which after differentiation with ε_i gives

$$\frac{\partial(\tau_{\mathrm{EQ}_i} - \tau_{\mathrm{EQ}_j})}{\partial\varepsilon_i} = \sum_{r=1}^{N}(\mu)_r(\alpha)_r\left(\frac{2}{3}\bar{b}_i^{(\alpha)_r/2} + \frac{1}{3}\bar{b}_j^{(\alpha)_r/2}\right) \tag{2.187}$$

Calculation of Non-Equilibrium Spatial Tangent Stiffness Modulus
For the non-equilibrium part, instead of calculating the spatial tangent stiffness modulus $\overset{4}{\mathscr{C}}_{\mathrm{NEQ}}$ by push forward of the material tangent stiffness modulus $\overset{4}{\mathscr{L}}_{\mathrm{NEQ}}$ with $\mathbf{F}$, an alternative method is used [8]. Here, instead of the expression in Eq. (2.110), an equivalent alternative multiplicative split of $\mathbf{F}$ is considered

$$\mathbf{F} = (\mathbf{F}_e)_{\mathrm{trial}}(\mathbf{F}_i)_{t=t_{n-1}} \tag{2.188}$$

which follows from the fact that $(\mathbf{F}_e)_{\mathrm{trial}} = (\mathbf{F})_{t=t_n}(\mathbf{F}_i^{-1})_{t=t_{n-1}}$. Due to this, at time $t = t_n$, $(\mathbf{F}_i^{-1})_{t=t_{n-1}}$ has to be treated constant. Thus, the increment

$$\Delta\mathbf{F} = \Delta(\mathbf{F}_e)_{\mathrm{trial}}(\mathbf{F}_i)_{t=t_{n-1}} \tag{2.189}$$

depends only on the increment $\Delta(\mathbf{F}_e)_{\text{trial}}$. Hence, an intermediate second Piola-Kirchoff stress tensor $\tilde{\mathbf{S}}_{\text{NEQ}}$ is sought. Consider the non-equilibrium second Piola-Kirchoff stress tensor given by

$$\begin{aligned}\mathbf{S}_{\text{NEQ}} &= \mathbf{F}^{-1}\boldsymbol{\tau}_{\text{NEQ}}\mathbf{F}^{-T} \\ &= (\mathbf{F}_i^{-1})_{t=t_{n-1}}\underbrace{(\mathbf{F}_e^{-1})_{\text{trial}}\boldsymbol{\tau}_{\text{NEQ}}(\mathbf{F}_e^{-T})_{\text{trial}}}_{\tilde{\mathbf{S}}_{\text{NEQ}}}(\mathbf{F}_i^{-T})_{t=t_{n-1}}\end{aligned} \tag{2.190}$$

which leads to the definition

$$\tilde{\mathbf{S}}_{\text{NEQ}} = (\mathbf{F}_i)_{t=t_{n-1}}\mathbf{S}_{\text{NEQ}}(\mathbf{F}_i^T)_{t=t_{n-1}} \tag{2.191}$$

The non-equilibrium spatial tangent modulus $\overset{4}{\mathscr{C}}_{\text{NEQ}}$ is then found by first determining the material tensor

$$\overset{4}{\widetilde{\mathscr{L}}}_{\text{NEQ}} = 2\frac{\partial\tilde{\mathbf{S}}_{\text{NEQ}}}{\partial(\mathbf{C}_e)_{\text{trial}}} \tag{2.192}$$

and then pushing it forward with $(\mathbf{F}_e)_{\text{trial}}$. Using the spectral decomposition, we also have

$$\tilde{\mathbf{S}}_{\text{NEQ}} = \sum_{i=1}^{3}\frac{\tau_{\text{NEQ}_i}}{(\lambda_{i_e}^2)_{\text{trial}}}\tilde{\boldsymbol{N}}_i\otimes\tilde{\boldsymbol{N}}_i \tag{2.193}$$

where

$$\tilde{S}_{\text{NEQ}_i} = \frac{\tau_{\text{NEQ}_i}}{(\lambda_{i_e}^2)_{\text{trial}}}. \tag{2.194}$$

Finally, taking into account

$$\Delta\tilde{\mathbf{S}}_{\text{NEQ}} = \overset{4}{\widetilde{\mathscr{L}}}_{\text{NEQ}} : \frac{1}{2}\Delta(\mathbf{C}_e)_{\text{trial}} \tag{2.195}$$

leads to the non-equilibrium tangent stiffness modulus in the intermediate configuration given by

$$\begin{aligned}\overset{4}{\widetilde{\mathscr{L}}}_{\text{NEQ}} &= \sum_{i=1}^{3}\sum_{j=1}^{3}(L_{ij})\,\tilde{\boldsymbol{N}}_i\otimes\tilde{\boldsymbol{N}}_i\otimes\tilde{\boldsymbol{N}}_j\otimes\tilde{\boldsymbol{N}}_j \\ &\quad+\sum_{i=1}^{3}\sum_{j\neq i=1}^{3}(G_{ij})\left[\tilde{\boldsymbol{N}}_i\otimes\tilde{\boldsymbol{N}}_j\otimes\tilde{\boldsymbol{N}}_i\otimes\tilde{\boldsymbol{N}}_j+\tilde{\boldsymbol{N}}_i\otimes\tilde{\boldsymbol{N}}_j\otimes\tilde{\boldsymbol{N}}_j\otimes\tilde{\boldsymbol{N}}_i\right]\end{aligned} \tag{2.196}$$

$$\overset{4}{\widetilde{\mathscr{L}}}_{\text{NEQ}} = \sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3}\sum_{l=1}^{3}(\tilde{L}_{ijkl})\tilde{\boldsymbol{N}}_i\otimes\tilde{\boldsymbol{N}}_j\otimes\tilde{\boldsymbol{N}}_k\otimes\tilde{\boldsymbol{N}}_l \tag{2.197}$$

where

$$L_{ij} = \frac{C^{\text{alg}}_{ij} - 2\tau_{\text{NEQ}_i}\delta_{ij}}{(\lambda^2_{i_e})_{\text{trial}}(\lambda^2_{j_e})_{\text{trial}}} \quad i,j = 1,2,3 \tag{2.198}$$

$$G_{ij} = \begin{cases} \dfrac{S_{\text{NEQ}_j} - S_{\text{NEQ}_i}}{(\lambda^2_{j_e})_{\text{trial}} - (\lambda^2_{i_e})_{\text{trial}}}, & \text{if } (\lambda_{i_e})_{\text{trial}} \neq (\lambda_{j_e})_{\text{trial}} \\ \dfrac{\partial\left(S_{\text{NEQ}_j} - S_{\text{NEQ}_i}\right)}{\partial(\lambda^2_{j_e})_{\text{trial}}}, & \text{if } (\lambda_{i_e})_{\text{trial}} = (\lambda_{j_e})_{\text{trial}} \end{cases} \quad i,j = 1,2,3 \tag{2.199}$$

Here again, in Eq. (2.199) for the case of coalesence of the trial principal stretches, i.e. for $(\lambda_{i_e})_{\text{trial}} \to (\lambda_{j_e})_{\text{trial}}$, the limiting value is calculated using L'Hospital rule. This expression can then be calculated using Eq. (2.198) [cf. 10, Eq 3.263] and is given by

$$\frac{\partial\left(S_{\text{NEQ}_j} - S_{\text{NEQ}_i}\right)}{\partial(\lambda^2_{j_e})_{\text{trial}}} = \frac{1}{2}L_{ii} - L_{ij} \tag{2.200}$$

Furthermore, in Eq. (2.198) we have for the algorithmic tangent matrix

$$\begin{aligned} C^{\text{alg}}_{ij} &= \frac{\partial\tau_{\text{EQ}_i}}{\partial\varepsilon_k}\mathrm{K}^{-1}_{kj} \\ &= \left[\frac{\partial(\text{dev}(\tau_{\text{NEQ}})_j)}{\partial\varepsilon_{k_e}} + \frac{\partial(\text{vol}(\tau_{\text{NEQ}})_i)}{\partial\varepsilon_{k_e}}\right]\mathrm{K}^{-1}_{kj} \end{aligned} \tag{2.201}$$

which can be found by using the values from Eq. (2.159), Eq. (2.160), Eq. (2.184) and Eq. (2.158) after the convergence of the Newton-Raphson iteration procedure. Finally after push forward with $(\mathbf{F}_e)_{\text{trial}}$, Eq. (2.197) yields

$$\overset{4}{\mathscr{C}}_{\text{NEQ}} = \sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3}\sum_{l=1}^{3}(\tilde{L}_{ijkl})(\lambda_{i_e})_{\text{trial}}(\lambda_{j_e})_{\text{trial}}(\lambda_{k_e})_{\text{trial}}(\lambda_{l_e})_{\text{trial}}\boldsymbol{n}_i \otimes \boldsymbol{n}_j \otimes \boldsymbol{n}_k \otimes \boldsymbol{n}_l \tag{2.202}$$

2.2.5 Jaumann Rate Formulation of Stiffness Modulus

In the finite element formulation for large deformations, the construction of a rate form for constitutive equations deduced from a strain energy function is required. This is easily performed in the reference configuration and takes the form

$$\dot{\mathbf{S}} = \overset{4}{\mathscr{L}} : \dot{\mathbf{E}} \tag{2.203}$$

However such a definition is not appropriate for the rate of Cauchy stress tensor $\dot{\boldsymbol{\sigma}}$ or the Kirchoff stress tensor $\dot{\boldsymbol{\tau}}$, since they are related to different configurations at time t_n and t_{n-1} and thus would not satisfy the requirements of material objectivity.

In order to compute an objective time derivative we consider with the help of Eq. (2.72), Eq. (2.60)

$$\begin{aligned}\dot{\boldsymbol{\tau}} &= \overline{(\mathbf{F}\mathbf{S}\mathbf{F}^{\mathbf{T}})}^{\cdot} \\ &= \dot{\mathbf{F}}\mathbf{S}\mathbf{F}^T + \mathbf{F}\dot{\mathbf{S}}\mathbf{F}^T + \mathbf{F}\mathbf{S}\dot{\mathbf{F}}^{\mathbf{T}} \\ &= \mathbf{l}\mathbf{F}\mathbf{S}\mathbf{F}^T + \mathbf{F}\dot{\mathbf{S}}\mathbf{F}^T + \mathbf{F}\mathbf{S}\mathbf{F}^T\mathbf{l}^T \\ \dot{\boldsymbol{\tau}} &= \mathbf{l}\boldsymbol{\tau} + \mathbf{F}\dot{\mathbf{S}}\mathbf{F}^T + \boldsymbol{\tau}\mathbf{l}^T\end{aligned} \tag{2.204}$$

which leads to the definition of the Trusdell rate [9] or equivalently the Lie derivative of the Kirchoff stress tensor $\mathring{\boldsymbol{\tau}}$ given by

$$\mathring{\boldsymbol{\tau}} = \mathcal{L}_v\boldsymbol{\tau} = \mathbf{F}\dot{\mathbf{S}}\mathbf{F}^T = \dot{\boldsymbol{\tau}} - \mathbf{l}\boldsymbol{\tau} - \boldsymbol{\tau}\mathbf{l}^T \tag{2.205}$$

Another objective rate form of the Kirchoff stress tensor, which is widely used in finite element formulations and also ABAQUS, is the Jaumann rate which is given by

$$\overset{\nabla J}{\boldsymbol{\tau}} = \dot{\boldsymbol{\tau}} - \mathbf{w}\boldsymbol{\tau} - \boldsymbol{\tau}\mathbf{w}^T \tag{2.206}$$

where $\mathbf{w}$ is the spin tensor. Keeping in mind Eq. (2.62), it can be shown that

$$\overset{\nabla J}{\boldsymbol{\tau}} = \dot{\boldsymbol{\tau}} - \mathbf{l}\boldsymbol{\tau} - \boldsymbol{\tau}\mathbf{l}^T + \mathbf{d}\boldsymbol{\tau} + \boldsymbol{\tau}\mathbf{d}^T \tag{2.207}$$

Comparing Eq. (2.205) with the above equation and by virtue of symmetry of the deformation tensor $\mathbf{d}$, we have

$$\mathring{\boldsymbol{\tau}} = \overset{\nabla J}{\boldsymbol{\tau}} - \mathbf{d}\boldsymbol{\tau} - \boldsymbol{\tau}\mathbf{d} \tag{2.208}$$

which gives a relation between the Trussdell rate and Jaumann rate of the Kirchoff stress tensor.

With the help of Eq. (2.205), Eq. (2.203) and Eq. (2.63), we have

$$\begin{aligned}\mathring{\boldsymbol{\tau}} &= \mathbf{F}\dot{\mathbf{S}}\mathbf{F}^T \\ &= \mathbf{F}\left(\overset{4}{\mathscr{L}} : \mathbf{F}^T\mathbf{d}\mathbf{F}\right)\mathbf{F}^T \\ &= \mathbf{F}\mathbf{F}(\overset{4}{\mathscr{L}})\mathbf{F}^T\mathbf{F}^T : \mathbf{d} \\ \mathring{\boldsymbol{\tau}} &= \overset{4}{\mathscr{C}} : \mathbf{d}\end{aligned} \tag{2.209}$$

where Eq. (2.209) gives the objective (Trussdell) rate formulation for the constitutive relation in the current configuration. Furthermore, using Eq. (2.208) in Eq. (2.209), we have

$$\overset{\nabla\mathrm{J}}{\boldsymbol{\tau}} = \overset{4}{\mathscr{C}} : \mathbf{d} + \mathbf{d}\boldsymbol{\tau} + \boldsymbol{\tau}\mathbf{d} \tag{2.210}$$

which finally leads to the Jaumann rate formulation of the constitutive relation

$$\overset{\nabla\mathrm{J}}{\boldsymbol{\tau}} = \overset{4}{\mathscr{C}}\Big|_{\boldsymbol{\tau}}^{\mathrm{FE}} : \mathbf{d} \tag{2.211}$$

where

$$\overset{4}{\mathscr{C}}\Big|_{\boldsymbol{\tau}}^{\mathrm{FE}} = \overset{4}{\mathscr{C}} + \frac{1}{2}\left[\delta_{ik}\tau_{jl} + \delta_{il}\tau_{jk} + \delta_{jl}\tau_{ik} + \delta_{jk}\tau_{il}\right] \boldsymbol{e}_i \otimes \boldsymbol{e}_j \otimes \boldsymbol{e}_k \otimes \boldsymbol{e}_l \tag{2.212}$$

In the above equation, the first term when contracted with $\mathbf{d}$ contributes to the objective rate of the Kirchoff stress tensor while the second term when contracted with $\mathbf{d}$ contributes to the rate of the Kirchoff stress tensor associated with local spin or rigid body motion. Furthermore, the rate formulation of the constitutive relation involving the Cauchy stress tensor is given by

$$\dot{\boldsymbol{\sigma}} = \overset{4}{\mathscr{C}}\Big|_{\boldsymbol{\sigma}}^{\mathrm{FE}} : \mathbf{d} = \frac{1}{J}\overset{4}{\mathscr{C}}\Big|_{\boldsymbol{\tau}}^{\mathrm{FE}} : \mathbf{d} \tag{2.213}$$

where

$$\overset{4}{\mathscr{C}}\Big|_{\boldsymbol{\sigma}}^{\mathrm{FE}} = \frac{1}{J}\overset{4}{\mathscr{C}} + \frac{1}{2J}\left[\delta_{ik}\tau_{jl} + \delta_{il}\tau_{jk} + \delta_{jl}\tau_{ik} + \delta_{jk}\tau_{il}\right] \boldsymbol{e}_i \otimes \boldsymbol{e}_j \otimes \boldsymbol{e}_k \otimes \boldsymbol{e}_l \tag{2.214}$$

$$= \sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3}\sum_{l=1}^{3}(C_{ijkl})\boldsymbol{e}_i \otimes \boldsymbol{e}_j \otimes \boldsymbol{e}_k \otimes \boldsymbol{e}_l \tag{2.215}$$

is the spatial tangent stiffness modulus which is symmetric [8]. Here, the coefficient matrix (C_{ijkl}) is sparse, and thus, can be stored in the Voigt notation form given by

$$\left(\overset{4}{\mathscr{C}}\Big|_{\boldsymbol{\sigma}}^{\mathrm{FE}}\right)_{\mathrm{VOIGT}} = C_{ij}^{\mathrm{ABAQUS}} = \begin{bmatrix} C_{1111} & C_{1122} & C_{1133} & C_{1112} & C_{1113} & C_{1123} \\ & C_{2222} & C_{2233} & C_{2212} & C_{2213} & C_{2223} \\ & & C_{3333} & C_{3312} & C_{3313} & C_{3323} \\ & & & C_{1212} & C_{1213} & C_{1223} \\ & \text{sym.} & & & C_{1313} & C_{1323} \\ & & & & & C_{2323} \end{bmatrix} \tag{2.216}$$

and is also implemented in the ABAQUS finite element software [2]. Lastly, for the finite viscoelasticity theory introduced earlier, we have for the equilibrium and non-equilibrium parts

$$\overset{4}{\mathscr{C}}_{\mathrm{EQ}}\Big|_{\boldsymbol{\sigma}}^{\mathrm{FE}} = \frac{1}{J}\overset{4}{\mathscr{C}}_{\mathrm{EQ}} + \frac{1}{2J}\Big[\delta_{ik}\tau_{\mathrm{EQ}_{jl}} + \delta_{il}\tau_{\mathrm{EQ}_{jk}} + \delta_{jl}\tau_{\mathrm{EQ}_{ik}} + \delta_{jk}\tau_{\mathrm{EQ}_{il}}\Big]\, \boldsymbol{e}_i \otimes \boldsymbol{e}_j \otimes \boldsymbol{e}_k \otimes \boldsymbol{e}_l \qquad (2.217)$$

and

$$\overset{4}{\mathscr{C}}_{\mathrm{NEQ}}\Big|_{\boldsymbol{\sigma}}^{\mathrm{FE}} = \frac{1}{J}\overset{4}{\mathscr{C}}_{\mathrm{NEQ}} + \frac{1}{2J}\Big[\delta_{ik}\tau_{\mathrm{NEQ}_{jl}} + \delta_{il}\tau_{\mathrm{NEQ}_{jk}} + \delta_{jl}\tau_{\mathrm{NEQ}_{ik}} + \delta_{jk}\tau_{\mathrm{NEQ}_{il}}\Big]\, \boldsymbol{e}_i \otimes \boldsymbol{e}_j \otimes \boldsymbol{e}_k \otimes \boldsymbol{e}_l \qquad (2.218)$$

Chapter 3

Implementation of User Material (UMAT) Subroutine

The necessary theoretical basis for the finite viscoelasticity theory has been discussed in the previous chapter. The goal in this chapter is to describe the implementation details of the finite viscoelastic material model for the User Material subroutine `UMAT` that is used in the ABAQUS/Standard finite element software. The source code for the subroutine is written using the FORTRAN programming language.

3.1 Interface to the UMAT Subroutine

Through the `UMAT` subroutine, it is possible to program specialized material models that are not already available in ABAQUS/Standard. For this, a well-documented interface is made available. Access to a large number of variables that could be necessary to implement the constitutive relations are made available within the subroutine. Out of these

- `DFGRD0` - the deformation gradient $\mathbf{F}_{t=t_{n-1}}$ at the start of the current time increment step,
- `DFGRD1` - the deformation gradient $\mathbf{F}_{t=t_n}$ at the end of the current time increment step,
- `DTIME` - the time increment Δt,
- `PROPS` - the parameters of the material model, and
- `STATEV` - the internal state variable(s)

are relevant.

Finally as required by the finite element formulation, the subroutine requires the calculation of

- `STRESS` - the Cauchy stress in the Voigt-notation
- `DDSDDE` - the spatial tangent stiffness modulus $\overset{4}{\mathscr{C}}\Big|_{\boldsymbol{\sigma}}^{\mathrm{FE}}$, and
- `STATEV` - the internal state variable(s)

as outputs. Furthermore, during the finite element solution procedure, this subroutine is called at every gauss integration point within an element in the model, and hence, should be programmed as efficiently as possible.

3.2 Adaptations to Viscoelasticity Theory

In order to implement a `UMAT` that is as simple as possible while also being able to adequately describe a wide variety of viscoelastic materials (or Hydrogels), the following adaptations to the previously described theory are considered.

First order Ogden strain energy function The strain energy functions for the equilibrium part Ψ_{EQ} and the non-equilibrium part Ψ_{NEQ} of the viscoelastic material model are defined in Eq. (2.171) and Eq. (2.149). These are valid for any order ($r = 1, \ldots, N$) of the Ogden class of strain energy functions. Generally, a 3-rd order ($r = 3$) or 4-th order ($r = 4$) strain energy function is used to model hyperelastic materials. However, hereafter we shall consider simplified strain energy function for both the equilibrium and non-equilibrium parts with $r = 1$. This is done to keep the number of material parameters of the model to a minimum.

Multiple non-equilibrium relaxation mechanisms Recall that the total strain energy Ψ in Eq. (2.108) is made up of one equilibrium(EQ) part and k non-equilibrium(NEQ) parts corresponding to the several relaxation mechanisms. For the sake of describing the theory, however, the simple case of a single relaxation mechanism ($k = 1$) was considered. It will be seen later that such a simplified material model cannot adequately describe the material response observed in experimental data. Hence, models with two or more relaxation mechanims ($k > 1$) should to be considered. To this end, models with one, two and three non-equilibrium parts have been developed and tested as part of this work.

3.3 Utility Subroutines

For implemention of the `UMAT`, trivial operations that are encountered in tensor algebra were required. It would be imprudent to write algorithms for these operations by oneself, when they have already been developed over the years. Two such operations, for which freely available implementations were used, are further described.

Tensor Inverse The inverse of a tensor $\mathbf{A} = \mathrm{A}$ can be given by

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})}\mathrm{adj}(\mathbf{A}) \tag{3.1}$$

such that

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I} \tag{3.2}$$

If $\mathbf{A} = [\mathrm{A_{ij}}]\, \boldsymbol{e}_i \otimes \boldsymbol{e}_j$ is expressed in the Euclidian basis $\boldsymbol{e}_i$, its inverse can be calculated by

$$\mathbf{A}^{-1} = [\mathrm{A_{ij}}]^{-1}\boldsymbol{e}_i \otimes \boldsymbol{e}_j \tag{3.3}$$

where $[\mathrm{A_{ij}}]^{-1}$ is the inverse of the matrix $[\mathrm{A_{ij}}]$.

Matrix inversion is one of the most widely used operation in matrix algebra. Here, a subroutine `M33INV` made available by David G. Simpson from NASA Goddard Space Flight Center has been used.

Spectral Decomposition The spectral decomposition of a tensor $\mathbf{A}$ expressed in any basis involves solving an eigen value problem. This tensor can then be expressed in the eigen basis by

$$\mathbf{A} = \sum_{i=1}^{N} \Lambda_i\, \boldsymbol{N}_i \otimes \boldsymbol{N}_i \tag{3.4}$$

where Λ_i are the eigen values or principal values and $\boldsymbol{N}_i$ are the eigen vectors or principal directions of $\mathbf{A}$. If $\mathbf{A} = [\mathrm{A_{ij}}]\, \boldsymbol{e}_i \otimes \boldsymbol{e}_j$ is expressed in the Euclidian basis $\boldsymbol{e}_i$, then the spectral decomposition of $\mathbf{A}$ reduces to solving the eigen value problem for the matrix $[\mathrm{A_{ij}}]$. A wide variety of algorithms are already available to compute the eigen values and eigen vectors of a matrix. Here, a freely available subroutine `DSYEVJ3` [6] that implements the Jacobi method is used.

3.4 Implementation Algorithm

The implementation of the described finite viscoelastic material model with Ogden strain energy functions is discussed here. This implementation is valid for models with one or more relaxation mechanisms $k > 1$. As part of this work, models with one, two and three relaxation mechanisms have been developed. Briefly the entire algorithm can be divided into seven steps as follows:

Step 1 Initialize and compute necessary variables and constants

Step 2 Calculate trial left Cauchy-Green tensor(s)

Step 3 Solve NEQ evolution equation

Step 4 Update of the internal state variable(s)

Step 5 Calculate NEQ Cauchy stress and spatial tangent stiffness modulus

Step 6 Calculate EQ Cauchy stress and spatial tangent stiffness modulus

Step 7 Calculate total Cauchy stress and spatial tangent stiffness modulus

Furthermore, before proceeding with the aforementioned steps, local variables (scalars, vectors and tensors) along with their type and size have to be declared, so that the sufficient memory can be allocated for these variables during the finite element procedure.

Next, the implementation details for every step in the algorithm are elaborated. These details were used as pseudocode for the final implementation and also helped to effectively structure the source code.

Step 1

- **Goal:** Initialize and compute necessary varibles and constants
- **Requirements:** `STATEV`, `PROPS`, `DFGRD1`
- **Output:** $(\mathbf{b}_e)^k_{t_{n-1}}$, μ, α, K, J, $(\mu_v$, α_v, K_v, η_{dev}, $\eta_{\mathrm{vol}})^k$

$k = 1, \ldots, N$

Procedure:

- Define the second order Identity tensor $\mathbf{I}$
- Retrieve μ, α, K, $(\mu_v$, α_v, K_v, η_{dev}, $\eta_{\mathrm{vol}})^k$ from `PROPS`
- For the first increment, set `STATEV` keeping in mind $(\mathbf{b}_e)^k_{t_0} = \mathbf{I}$

- Retrieve $(\mathbf{b}_e)^k_{t_{n-1}}$ from `STATEV`
- Calculate J from `DFGRD1` | Eq. (2.16)

Notes:
Since internal variable(s) $\mathbf{b}^k_e$ is a symmetric tensor, it is stored in and retrieved from `STATEV` using the Voigt notation. Furthermore the values from two or more internal variables are stacked one after the other. Thus we have for a model with two relaxation mechanisms ($k = 2$),

$$\texttt{STATEV} = \left[\mathrm{b}_{\mathrm{e}11}^{1}\ \mathrm{b}_{\mathrm{e}22}^{1}\ \mathrm{b}_{\mathrm{e}33}^{1}\ \mathrm{b}_{\mathrm{e}12}^{1}\ \mathrm{b}_{\mathrm{e}13}^{1}\ \mathrm{b}_{\mathrm{e}23}^{1}\ \mathrm{b}_{\mathrm{e}11}^{2}\ \mathrm{b}_{\mathrm{e}22}^{2}\ \mathrm{b}_{\mathrm{e}33}^{2}\ \mathrm{b}_{\mathrm{e}12}^{2}\ \mathrm{b}_{\mathrm{e}13}^{2}\ \mathrm{b}_{\mathrm{e}23}^{2}\right]^T$$

Step 2

- **Goal:** Calculate trial left Cauchy-Green tensor(s)
- **Requirements:** $(\mathbf{b}_e)^k_{t_{n-1}}$ `DFGRD0`, `DFGRD1`
- **Output:** $(\mathbf{b}_e)^k_{\text{trial}}$

$k = 1, \ldots, N$

Procedure:

- Calculate $(\mathbf{b}^{-1}_e)^k_{t_{n-1}}$
- Calculate $(\mathbf{C}_i)^k_{t_{n-1}} = \mathbf{F}^T_{t=t_{n-1}}(\mathbf{b}^{-1}_e)^k_{t_{n-1}}\mathbf{F}_{t=t_{n-1}}$ | Eq. (2.127)
- Calculate $(\mathbf{C}^{-1}_i)^k_{t_{n-1}}$
- Calculate $(\mathbf{b}_e)^k_{\text{trial}} = \mathbf{F}_{t=t_n}(\mathbf{C}^{-1}_i)^k_{t_{n-1}}\mathbf{F}^T_{t=t_n}$ | Eq. (2.138)

Notes:
Alternatively, it is possible to arrive at $(\mathbf{b}_e)^k_{\text{trial}}$ from $(\mathbf{b}_e)^k_{t_{n-1}}$ using just two steps given by

$$(\mathbf{C}^{-1}_i)^k_{t_{n-1}} = \mathbf{F}^{-1}_{t=t_{n-1}}(\mathbf{b}^{-1}_e)^k_{t_{n-1}}\mathbf{F}^{-T}_{t=t_{n-1}}$$
$$(\mathbf{b}_e)^k_{\text{trial}} = \mathbf{F}_{t=t_n}(\mathbf{C}^{-1}_i)^k_{t_{n-1}}\mathbf{F}^T_{t=t_n}$$

However, since $\mathbf{b}_e$ is symmetric, positive definite and also inverting $\mathbf{F}$ could lead to numerical errors, this is not used.

Step 3

- **Goal:** Solve the NEQ evolution equation
- **Requirements:** $(\mathbf{b}_e)^k_{\text{trial}}$, $(\mu_v,\ \alpha_v,\ \mathrm{K}_v,\ \eta_{\text{dev}},\ \eta_{\text{vol}})^k$, `DTIME`
- **Output:** $(\mathbf{b}_e)^k$, $(\boldsymbol{\tau}_{\text{NEQ}})^k$, $(\boldsymbol{n}_i)^k$, $(\lambda^2_{i_e})^k_{\text{trial}}$, $(\tau_{\text{NEQ}_i})^k$, $(C^{\text{alg}}_{ij})^k$

$k = 1, \dots, N$

Procedure:

```
- Calculate principal values $(\lambda^2_{i_e})^k_{\text{trial}}$ and directions $(\boldsymbol{n}_i)^k_{\text{trial}}$ of $(\mathbf{b}_e)^k_{\text{trial}}$
- Calculate trial principal strain $(\varepsilon_{i_e})^k_{\text{trial}} = \frac{1}{2}\ln(\lambda^2_{i_e})^k_{\text{trial}}$
- Initialize iteration variables $(\lambda^2_{i_e})^k \leftarrow (\lambda^2_{i_e})^k_{\text{trial}}$ and $(\varepsilon_{i_e})^k \leftarrow (\varepsilon_{i_e})^k_{\text{trial}}$
- Perform Newton-Raphson iteration, for $m = 1, \dots,$ MAXITER
    – Calculate NEQ Jacobian $(J_e)^k$ | Eq. (2.146)
    – Calculate principal values $(\bar{b}_{i_e})^k$ of $(\bar{\mathbf{b}}_e)^k$ | Eq. (2.147)
    – Calculate prinicpal values $(\text{dev}(\tau_{\text{NEQ}})_i)^k$ of $(\text{dev}(\boldsymbol{\tau}_{\text{NEQ}}))^k$
    – Calculate the residual vector $(r_i)^k$ | Eq. (2.155)
    – Calculate norm of residual vector $||\boldsymbol{r}||$ | Eq. (2.164)
      If $||\boldsymbol{r}|| <$ tol and $m > 1$, Newton-Raphson iteration completed.
    – Calculate $\left(\frac{\partial(\text{dev}(\boldsymbol{\tau}_{\text{NEQ}})_i)}{\partial\varepsilon_{i_e}}\right)^k$, $\left(\frac{\partial(\text{dev}(\boldsymbol{\tau}_{\text{NEQ}})_i)}{\partial\varepsilon_{j_e}}\right)^k$ | Eq. (2.159), Eq. (2.160)
    – Calculate $(K_{ij})^k$ | Eq. (2.158), Eq. (2.184)
    – Calculate $(K^{-1}_{ij})^k$
    – Calculate strain increment $(\Delta\varepsilon_{i_e})^k = -(K^{-1}_{ij})^k(r_i)^k$ | Eq. (2.157)
    – Update principal strain $(\varepsilon_{i_e})^k$ | Eq. (2.162)
    – Update principal values $(\lambda^2_{i_e})^k = \exp\left(2\,(\varepsilon_{i_e})^k\right)$ of $(\mathbf{b}_e)^k$
- Calculate required quantities
    – $(\mathbf{b}_e)^k$ | Eq. (2.165)
    – $(\boldsymbol{\tau}_{\text{NEQ}})^k$ | Eq. (2.166)
    – $(C^{\text{alg}}_{ij})^k$ | Eq. (2.201)
```

Notes:
In order to calculate the prinicipal values and directions of $(\mathbf{b}_e)^k_{\text{trial}}$, instead of the inbuilt function `SPRIND` in ABAQUS, the general purpose subroutine `DSYEVJ3` is used. This allowed for standalone compilation and testing of the `UMAT` without having access to ABAQUS.

Step 4

- **Goal:** Update internal state variable(s)
- **Requirements:** $(\mathbf{b}_e)^k$
- **Output:** `STATEV`

$k = 1, \ldots, N$

Procedure:

- Update `STATEV` with values from $(\mathbf{b}_e)^k$

Step 5

- **Goal:** Calculate NEQ Cauchy stress and tangent stiffness modulus
- **Requirements:** J, $(\boldsymbol{\tau}_{\text{NEQ}})^k$, $(\boldsymbol{n}_i)^k$, $(\lambda^2_{i_e})^k_{\text{trial}}$, $(\tau_{\text{NEQ}_i})^k$, $(C^{\text{alg}}_{ij})^k$
- **Output:** $(\boldsymbol{\sigma}_{\text{NEQ}})^k$, $\left(\overset{4}{\mathscr{C}}_{\text{NEQ}}\Big|^{\text{FE}}_{\boldsymbol{\sigma}}\right)^k_{\text{VOIGT}}$

$k = 1, \ldots, N$

Procedure:

- Calculate $(\boldsymbol{\sigma}_{\text{NEQ}})^k = \frac{1}{J}(\boldsymbol{\tau}_{\text{NEQ}})^k$ | Eq. (2.70)
- Calculate coefficients of $\left(\overset{4}{\widetilde{\mathscr{L}}}_{\text{NEQ}}\right)^k$ | Eq. (2.197), Eq. (2.198), Eq. (2.199)
- Calculate $\left(\overset{4}{\mathscr{C}}_{\text{NEQ}}\right)^k$ | Eq. (2.202)

- Calculate $\left(\overset{4}{\mathscr{C}}_{\text{NEQ}}\Big|_{\boldsymbol{\sigma}}^{\text{FE}}\right)^k$ | Eq. (2.218)

- Calculate $\left(\overset{4}{\mathscr{C}}_{\text{NEQ}}\Big|_{\boldsymbol{\sigma}}^{\text{FE}}\right)^k_{\text{VOIGT}}$ | Eq. (2.216)

Step 6

- **Goal:** Calculate EQ Cauchy stress and tangent stiffness modulus
- **Requirements:** `DFGRD1`, J, μ, α, K
- **Output:** $\boldsymbol{\sigma}_{\text{EQ}}$, $\left(\overset{4}{\mathscr{C}}_{\text{EQ}}\Big|_{\boldsymbol{\sigma}}^{\text{FE}}\right)_{\text{VOIGT}}$

Procedure:

- Calculate $\mathbf{b}$ | Eq. (2.20)
- Calculate principal values λ_i^2 and directions $\boldsymbol{n}_i$ of $\mathbf{b}$
- Calculate principal values $\overline{b}_i$ of $\overline{\mathbf{b}}$ | Eq. (2.173)
- Calculate τ_{EQ_i} | Eq. (2.177), Eq. (2.178), Eq. (2.179)
- Calculate $\boldsymbol{\sigma}_{\text{EQ}}$ | Eq. (2.175), Eq. (2.70)
- Calculate c_{ij} | Eq. (2.181), Eq. (2.182), Eq. (2.183), Eq. (2.184)
- Calculate g_{ij} | Eq. (2.185), Eq. (2.187)
- Calculate coefficients of $\overset{4}{\mathscr{C}}_{\text{EQ}}$ | Eq. (2.180)
- Calculate $\overset{4}{\mathscr{C}}_{\text{EQ}}$ | Eq. (2.180)
- Calculate $\overset{4}{\mathscr{C}}_{\text{EQ}}\Big|_{\boldsymbol{\sigma}}^{\text{FE}}$ | Eq. (2.217)
- Calculate $\left(\overset{4}{\mathscr{C}}_{\text{EQ}}\Big|_{\boldsymbol{\sigma}}^{\text{FE}}\right)_{\text{VOIGT}}$ | Eq. (2.216)

Notes:
Here, to calculate the principal values and directions of $\mathbf{b}$, again the subroutine `DSYEVJ3` is used.

Step 7

- **Goal:** Calculate total Cauchy stress and tangent stiffness modulus
- **Requirements:** $\boldsymbol{\sigma}_{\text{EQ}}$, $(\boldsymbol{\sigma}_{\text{NEQ}})^k$, $\left({}^4\mathscr{C}_{\text{EQ}}\Big|_{\boldsymbol{\sigma}}^{\text{FE}}\right)_{\text{VOIGT}}$, $\left({}^4\mathscr{C}_{\text{NEQ}}\Big|_{\boldsymbol{\sigma}}^{\text{FE}}\right)^k_{\text{VOIGT}}$
- **Output:** $(\boldsymbol{\sigma})_{\text{VOIGT}}$, $\left({}^4\mathscr{C}\Big|_{\boldsymbol{\sigma}}^{\text{FE}}\right)_{\text{VOIGT}}$

$k = 1, \ldots, N$

Procedure:

- Calculate $\boldsymbol{\sigma} = \boldsymbol{\sigma}_{\text{EQ}} + \sum_{k=1}^{N} \boldsymbol{\sigma}_{\text{NEQ}}$
- Calculate $(\boldsymbol{\sigma})_{\text{VOIGT}}$
- Calculate $\left({}^4\mathscr{C}\Big|_{\boldsymbol{\sigma}}^{\text{FE}}\right)_{\text{VOIGT}} = \left({}^4\mathscr{C}_{\text{EQ}}\Big|_{\boldsymbol{\sigma}}^{\text{FE}}\right)_{\text{VOIGT}} + \sum_{k=1}^{N} \left({}^4\mathscr{C}_{\text{NEQ}}\Big|_{\boldsymbol{\sigma}}^{\text{FE}}\right)_{\text{VOIGT}}$

Notes:
The Voigt notation for $\boldsymbol{\sigma}$ compatible with Eq. (2.216) is given by

$$(\boldsymbol{\sigma})_{\text{VOIGT}} = [\sigma_{11}\ \sigma_{22}\ \sigma_{33}\ \sigma_{12}\ \sigma_{13}\ \sigma_{23}]^T$$

3.5 Twin Implementation in MATLAB

As an example, the FORTRAN source code for the `UMAT` using the previously described implementation with two relaxation mechanisms can be found in Appendix A. However, in order to facilitate quick debugging and, later on, have access to optimization libraries for parameter identification, it was useful to have a one-to-one *twin* implementation of the `UMAT` as a MATLAB function.

Since the MATLAB programming language is also 1-indexed, it was easy to translate the FORTRAN source code into MATLAB. Furthermore, certain in-built functions available within MATLAB, for example `eig(A)` instead of `DSYEVJ3`, were also used. Finally, the equivalence of both the MATLAB function and FORTRAN subroutine were tested by using the same input parameter arguments and comparing the outputs.

Chapter 4

Material Parameter Identification

The finite viscoelastic material model decribed in Section 2.2 is characterized by a set of material parameters. For the response of the implemented material model to be in close agreement with real materials, these parameters need to be identified. The task of identifying suitable parameters of the finite viscoelastic model such that it is able to resemble a dual cross-link self-healing hydrogel is dealt with in this chapter.

4.1 Material Parameters

The parameters of the viscoelastic material model comprise of parameters from the elastic and viscous (inelastic) parts. As we have considered a Ogden class of strain energy of the first order, the set of parameters required to fully describe the model with k relaxation mechanisms is given by

$$\boldsymbol{x}_p = \Big[\mu,\, \alpha,\, \mathrm{K},\, (\mu_v,\, \alpha_v,\, \mathrm{K}_v,\, \eta_{\mathrm{dev}},\, \eta_{\mathrm{vol}})^k\Big]^T \tag{4.1}$$

where $\boldsymbol{x}_p$ is the parameter vector and the total number of parameters is given by $3 + 5k$. Thus, for a viscoelastic model with two relaxation mechanisms ($k = 2$), the material model is characterized by 13 parameters in total. Furthermore, the viscosity parameters η_{dev} and η_{vol} can be given by

$$\eta_{\mathrm{dev}} = \mu_{vis}\tau_r = \mu_v\alpha_v\tau_r \tag{4.2}$$
$$\eta_{\mathrm{vol}} = \mathrm{K}_v\tau_r \tag{4.3}$$

where τ_r is the relaxation time [5]. The set of parameters can thus be reduced to

$$\hat{\boldsymbol{x}}_p = \Big[\mu,\, \alpha,\, \mathrm{K},\, (\mu_v,\, \alpha_v,\, \mathrm{K}_v,\, \tau_r)^k\Big]^T \tag{4.4}$$

where the total number of parameters can now be given by $3 + 4k$. As it will be seen further, reducing the number of the material parameters in this way benefits to the parameter identification procedure.

4.2 Experimental Data

In order to find a suitable set of parameters as described in Eq. (4.4) of the viscoelastic material model, experimental data is required. However, performing experiments and collecting test data is a very time consuming process. Here, experimental data from uniaxial tension tests for a dual cross-link self-healing hydrogel is taken from Long et al. [7]. In total, data from five experiments that were carried out at constant stretch rates varying from $0.1s^{-1}$ to $0.003s^{-1}$ as shown in Fig. 4.1 were used. At every time step t, the data contained a value for nominal stress σ_{nom} or $\mathrm{P}_{11}|_{\text{exp}}$ and stretch λ in the axial direction of the test.

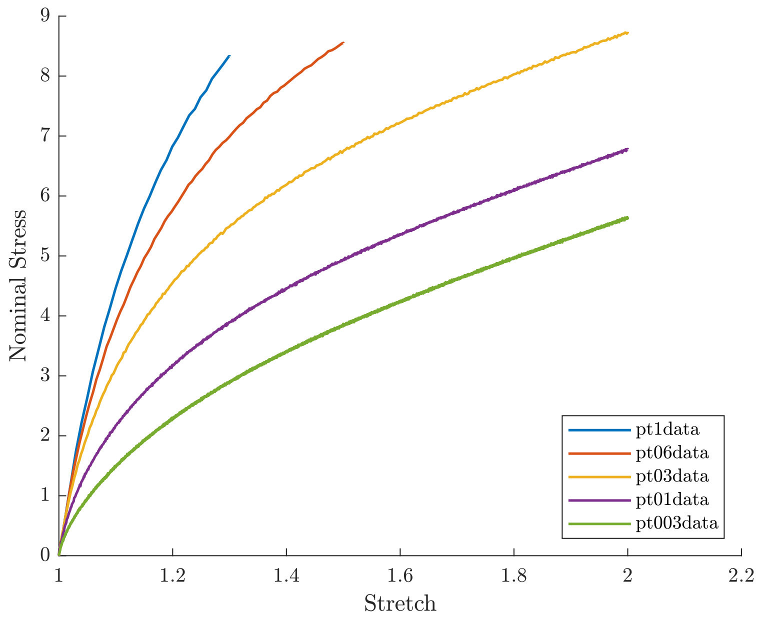


Figure 4.1: Uniaxial tension experimental data of a hydrogel.

4.3 Non-Linear Least-Squares Optimization

The goal of the parameter identification procedure is to find the set of paramaters for which the error between the experimental data and simulation is minimized. This then reduces to a non-linear least-squares optimization problem, in which a cost function is minimized over a set of variables (material parameters). Here, the cost function is given by

$$\mathrm{Cost}(\hat{\boldsymbol{x}}_p) = \sum_{j=1}^{E} \sum_{i=1}^{N} \left|\left| (\mathbf{P}|_{\mathrm{exp}})_i^j - (\mathbf{P}|_{\mathrm{sim}})_i^j \right|\right| \tag{4.5}$$

where $\mathbf{P}$ is the first Piola-Kirchoff stress tensor Eq. (2.69) representing the nominal stress, E is the total number of experiments, N is the number of data points within an experiment, and $|| \bullet ||$ is the Frobenius norm. In the case of uniaxial tension this reduces to

$$\mathrm{Cost}(\hat{\boldsymbol{x}}_p) = \sum_{j=1}^{E} \sum_{i=1}^{N} \left|\left| (\mathrm{P}_{11}|_{\mathrm{exp}})_i^j - (\mathrm{P}_{11}|_{\mathrm{sim}})_i^j \right|\right| \tag{4.6}$$

The optimization problem can be then written as

$$\underset{\hat{\boldsymbol{x}}_p}{\mathrm{argmin}} \ \mathrm{Cost}(\hat{\boldsymbol{x}}_p) \tag{4.7}$$

additionally, subject to constraints

$$\mathrm{K} > 0 \quad \text{and} \quad (\mathrm{K}_v)^k > 0 \tag{4.8}$$

$$\mathrm{sgn}(\mu) = \mathrm{sgn}(\alpha) \quad \text{and} \quad \mathrm{sgn}(\mu_v)^k = \mathrm{sgn}(\alpha_v)^k$$

or

$$\mu\alpha > 0 \quad \text{and} \quad (\mu_v)^k(\alpha_v)^k > 0 \tag{4.9}$$

since the bulk and shear modulus cannot have a negative value. In general, Eq. (4.8) and Eq. (4.9) represent inequality constraints. Furthermore keeping Eq. (4.2) and Eq. (4.3) in mind, if the parameter vector specified in Eq. (4.1) would have been used, an additional equality constraint given by

$$\frac{\eta_{\mathrm{dev}}}{\mu_v \alpha_v} = \frac{\eta_{\mathrm{vol}}}{\mathrm{K}_v} \tag{4.10}$$

would have to be satisfied. However, since the parameter vector specified in Eq. (4.4) is used, this constraint is already satisfied and need not be considered.

Solving a constrained optimization problem is much more resource intensive and time consuming as compared to solving an unconstrained optimization problem. Hence, the constrained optimization problem at hand Eq. (4.7), was converted into an unconstrained problem using the penalty method. To this end, a large penalty value is added to the cost when any of the constraints in Eq. (4.8), Eq. (4.9) are violated. With this conversion, the optimization problem was given by

$$\underset{\hat{\boldsymbol{x}}_p}{\operatorname{argmin}} \ \mathrm{Cost}(\hat{\boldsymbol{x}}_p)$$

$$\mathrm{Cost}(\hat{\boldsymbol{x}}_p) = \sum_{j=1}^{E}\sum_{i=1}^{N} \mid\mid (\mathrm{P}_{11}|_{\mathrm{exp}})_i^j - (\mathrm{P}_{11}|_{\mathrm{sim}})_i^j \mid\mid + \mathrm{Penalty}(\hat{\boldsymbol{x}}_p) \tag{4.11}$$

To solve this, the `lsqnonlin` function in MATLAB was used. For the cost function, the material response was simulated for a given set of parameters $\hat{\boldsymbol{x}}_p$ using the twin implementation of the material model described in Section 3.5. Here, the first Piola-Kirchoff stress tensor can be found using Eq. (2.72). As is needed by the material model, the deformation gradient $\mathbf{F}$ at the start (`DFGRD0`) and end (`DFGRD1`) of the current increment step were computed from stretch at the start $\lambda_{t_{n-1}}$ and end λ_{t_n} of the current increment step, respectively from the experimental data. Furthermore, the time increment Δt was computed by taking the difference in time at the start and end of the current increment step. For the case of uniaxial tension, we have the deformation gradient at the end of the current time increment

$$(\mathbf{F})_{t_n} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \boldsymbol{e}_i \otimes \boldsymbol{e}_j \tag{4.12}$$

where $\lambda_1 = \lambda_{t_n}$ is the stretch in the axial direction and $\lambda_2 = \lambda_3 = f(\lambda_1)$ are the stretches in the transverse directions. The deformation gradient $(\mathbf{F})_{t_{n-1}}$ at the start of the time increment can be found in a similar manner.

Initially during the optimization process, the material was considered to be perfectly incompressible, i.e. $J = \det(\mathbf{F}) = 1$. Thus, the stretches in the transverse directions were given by,

$$\lambda_2 = \lambda_3 = \frac{1}{\sqrt{\lambda_1}} \tag{4.13}$$

However, consequently the optimization problem was found to be insensitive to the bulk modulus K and $(\mathrm{K}_v)^k$ as will be seen in Section 4.4. Indeed if $J = 1$, keeping Eq. (2.179), Eq. (2.154) in mind, there is no contribution of volumetric stress to the total stress, and hence, K and $(\mathrm{K}_v)^k$ could take any value without affecting the simulated response. Furthemore, as a result, the nominal stress in the tranvese directions $\mathrm{P}_{22}|_{\text{sim}} = \mathrm{P}_{33}|_{\text{sim}} \neq 0$. This negates the fact that, for a uniaxial tension experiment stresses in the transverse direction are zero.

Later on, to avoid these shortcomings, given the stretch in the axial direction λ_1, the stretches in the tranverse directions λ_2 and λ_3 were computed by finding the roots of the function

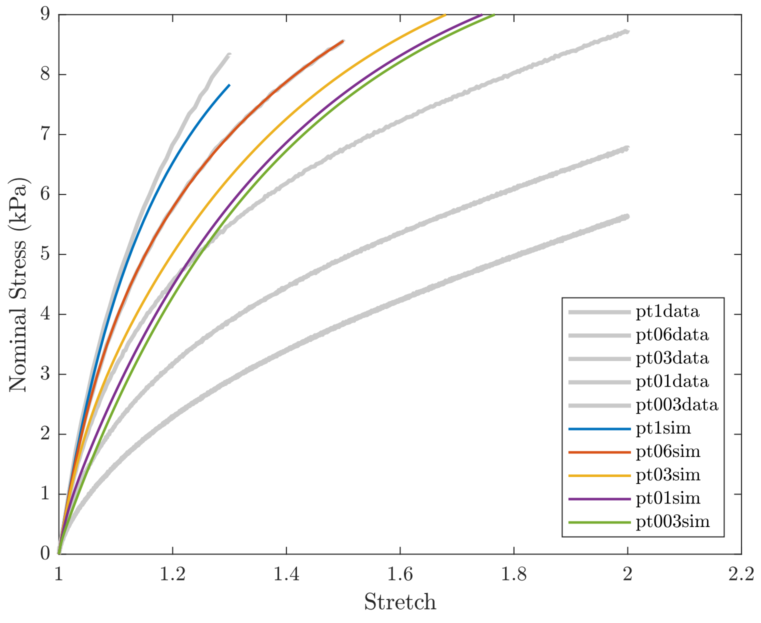
$$\mathrm{P}_{22}|_{\text{sim}}(\lambda_1, \lambda_2, \lambda_3) = 0 \text{ or } \mathrm{P}_{33}|_{\text{sim}}(\lambda_1, \lambda_2, \lambda_3) = 0 \tag{4.14}$$

For this, the root-finding function `fsolve` from MATLAB, which employs a Newton-Raphson's method, was used. Here, again the first Piola-Kirchoff stress tensor was calculated using the twin implementation of the material model described in section Section 3.5.

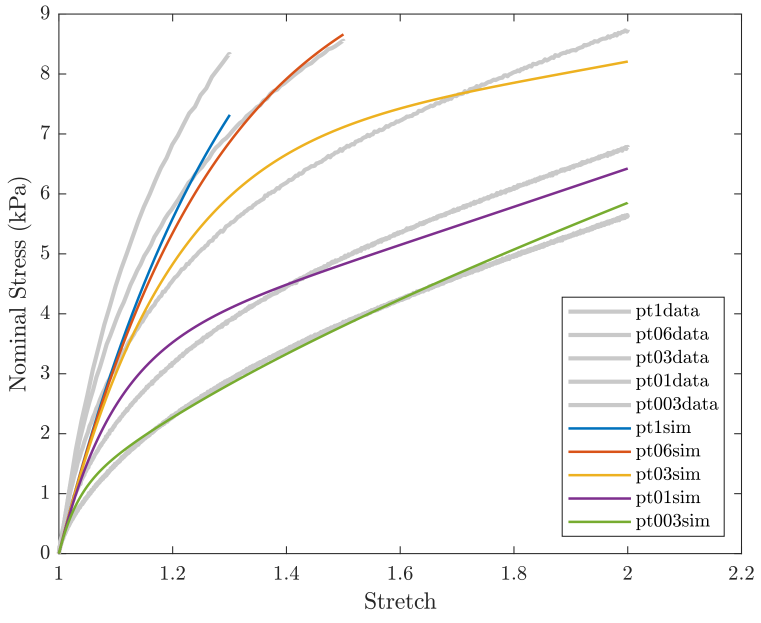
4.4 Results of the Optimization Problem

Firstly, considering the viscoelastic material model with a single relaxation mechanism ($k = 1$), when the cost was calculated including data from all the five experiments, it was not possible to find a suitable set of material parameters to fit the simulated response to the experimental data. However, for certain cases, when only a single experiment was considered to calculate the cost, it was possible to find a set of material parameters that would fit the material model response to the selected experiment. On the other hand, when these set of parameters were used to simulate the other experiments, the material response was not in agreement with the experimental data. This indicated that the viscoelastic material model with a single relaxation mechanism was too simple to sufficiently characterize the hydrogel considered.

Next, the viscoelastic material models with two ($k = 2$) or three ($k = 3$) relaxation mechanisms were considered. Even though, it was computationally more expensive to solve the optimization problem Eq. (4.7), it was possible to find a set of suitable parameters that would fit the material response to all the experiments together. As can be seen in Fig. 4.3 the response from the model with three relaxation mechanisms ($k = 3$) fits better to the experimental data. However, since certain material model parameters have no physical meaning when considered individually, the optimization problem provided different solutions when different starting points $(\boldsymbol{x}_p)_0$ in the parameter space were considered. Moreover, as can be seen from optimized parameter sets (a) and (b) in Table 4.1, it was found that the bulk $\mathrm{K}, \mathrm{K}_v^1, \mathrm{K}_v^2$ for the elastic and



(a)



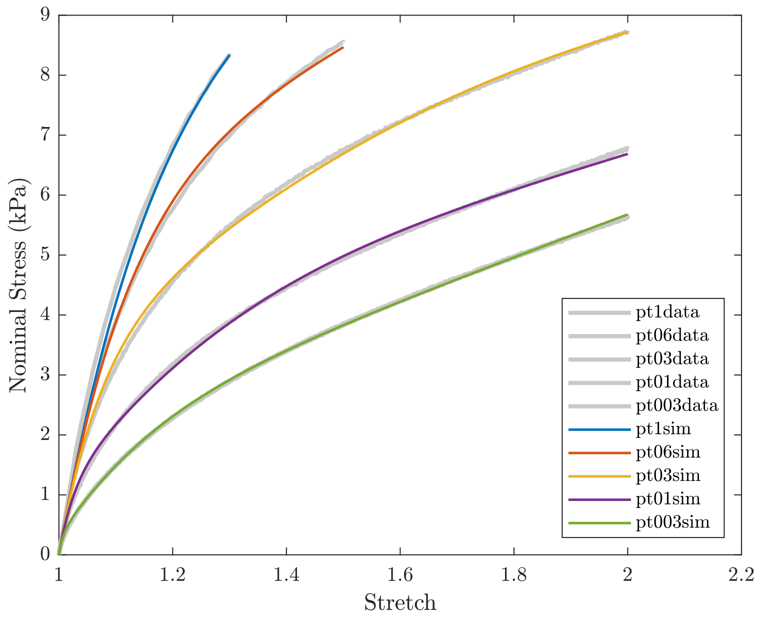
(b)

Figure 4.2: Parameter identification of viscoelastic material model with one relaxation mechanisms ($k = 1$) fit to (a) the experiment with stretch rate $0.06s^{-1}$ and (b) all the five experiments together

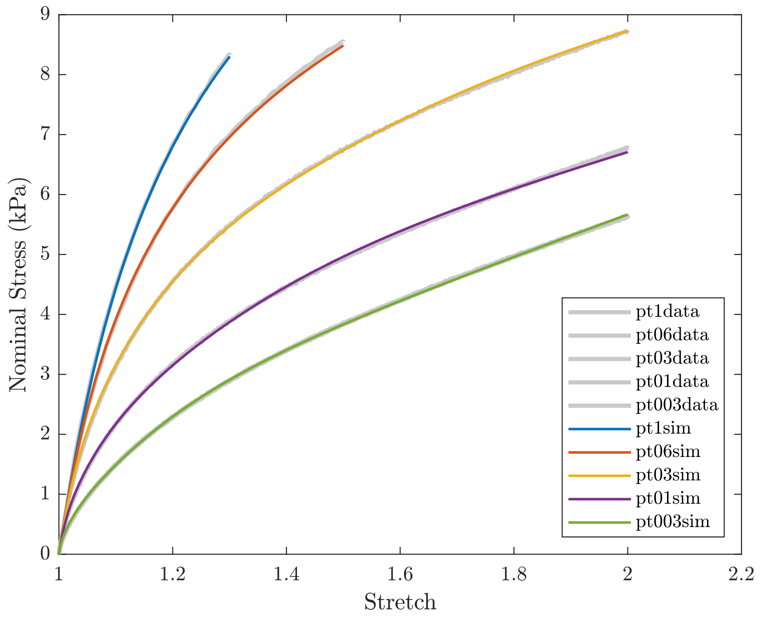
inealstic parts in the model with two relaxation mechanisms (2EL) had little to no effect on the parameter optimization procedure.

Furthermore, the cost function was updated such that the stresses in the transverse direction would be zero in order to exactly mimic the uniaxial tension experiment. It must be noted that, the root-finding procedure (Newton-Raphson's method), that was necessary to achieve this condition, led to a tremendous increase in the number of computations performed within the cost function. Nevertheless, using this revised cost function, it was possible find a set of suitable parameters for the models with two ($k = 2$) and three ($k = 3$) relaxation mechanisms that simulated a response which was in close agreement to experimental data. Here, the response of the model with two relaxation mechanisms ($k = 2$) fit the experimental data better as can be seen in Fig. 4.4. For the model with one relaxation mechanism ($k = 1$), however, even after the cost function update, it was still not possible to find a set of parameters to fit the simulation response to all the experiments together.

Finally, in an attempt to further reduce the number of model parameters, the viscous parameters $(\mu_v)^k, (\alpha_v)^k, (\mathrm{K}_v)^k$ were chosen to be equal to elastic parameters μ, α, K respectively, as is sometimes done in literature [8, see Sec. 5]. For the model with two relaxation mechanisms($k = 2$), using this methodology was found to be ineffective. Again as can be seen in Fig. 4.5, the model rendered itself to be very simple to characterize the uniaxial tension experiments of the hydrogel. The sets of optimized parameters for all the optimization procedures carried out are summarized in Table 4.1.

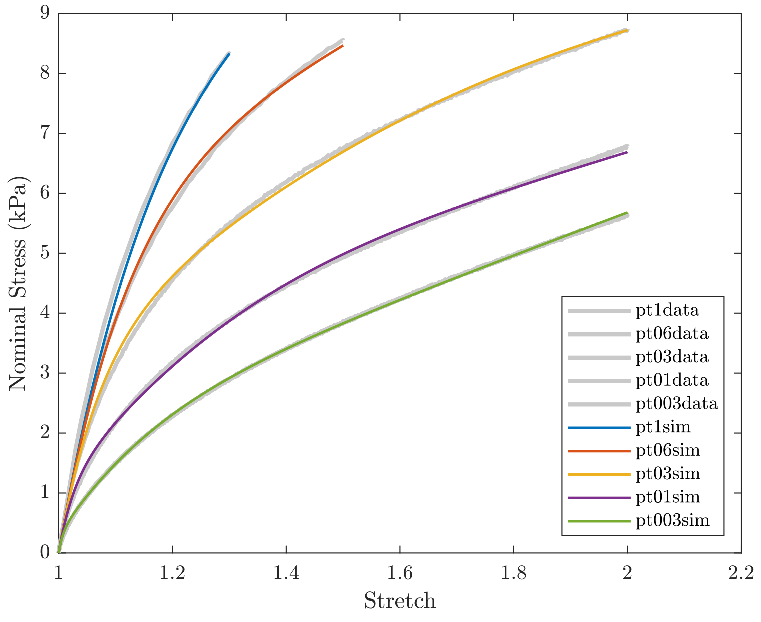


(a)

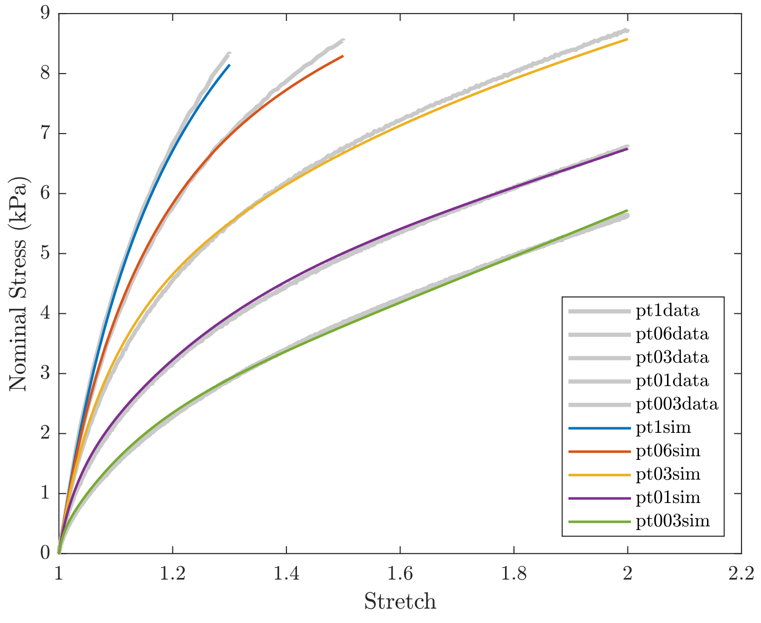


(b)

Figure 4.3: Parameter identification of viscoelastic material model (a) with two relaxation mechanisms ($k = 2$) and (b) with three relaxation mechanisms ($k = 3$) considering $J = 1$ and $\lambda_{2,3} = \frac{1}{\sqrt{\lambda_1}}$



(a)



(b)

Figure 4.4: Parameter identification of viscoelastic material model (a) with two relaxation mechanisms ($k = 2$) and (b) with three relaxation mechanisms ($k = 3$) satisfying the condition $\mathrm{P}_{22,33} = 0$

	$2EL_1$	$2EL_2$	$2EL_{NR}$	3EL	$3EL_{NR}$
$\boldsymbol{x}_p$	(a)	(b)	(c)	(d)	(e)
μ	0.0040	0.0040	0.0026	0.0039	0.0024
α	2.1471	2.1474	2.1478	2.1578	2.2817
K	4.0000	40.0000	29.4615	40.0000	5.2712
μ_v^1	0.0686	0.0664	0.0643	0.0414	0.1182
α_v^1	0.5827	0.6011	0.4168	0.8837	0.1168
K_v^1	6.0000	60.0000	61.3862	59.9990	25.7008
τ_r^1	3158.0	1.3159	1.2985	0.4912	0.3146
μ_v^2	0.0017	0.0017	0.0011	0.0042	0.1340
α_v^2	3.5330	3.5302	3.5251	3.5012	0.1376
K_v^2	5.0000	25.0000	29.0539	60.0006	14.4123
τ_r^2	36.9815	36.9446	36.5179	3.3429	1.5602
μ_v^3	-	-	-	0.0014	0.0032
α_v^3	-	-	-	3.4523	1.7457
K_v^3	-	-	-	24.9999	2.7381
τ_r^3	-	-	-	45.9230	26.4516

Table 4.1: Optimized parameter sets for the viscoealstic material model for the dual cross-link self-healing hydrogel.
(k)EL stands for a model with k relaxation mechanisms
$(\bullet)_{NR}$ stands for models using cost function with the modified root-finding (Newton-Raphson's) procedure Eq. (4.14)

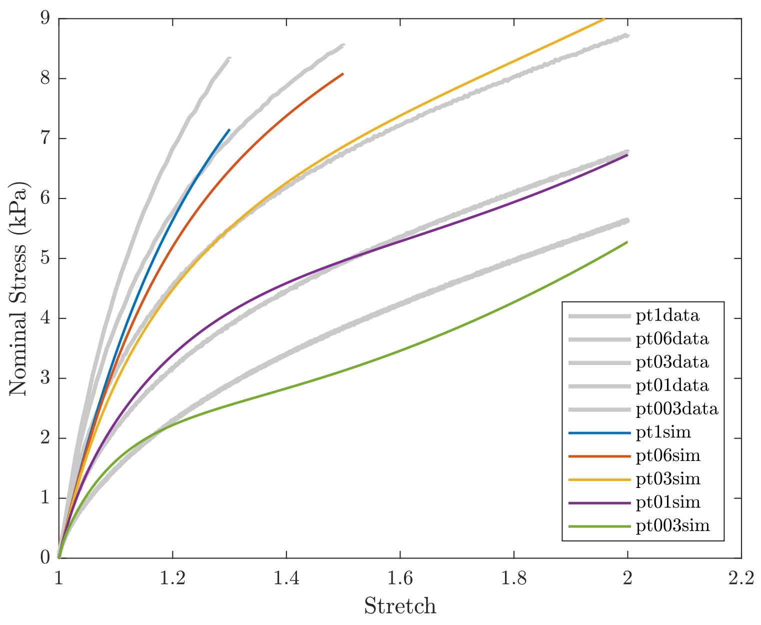


Figure 4.5: Parameter identification for a viscoelastic material model with two relaxation mechanisms ($k = 2$) and a reduced parameter set

Chapter 5

Numerical Simulations

In order to test the implemented `UMAT` subroutines for the viscoelastic material model, numerical simulations were carried out in ABAQUS. To this end, finite element models with both a single element and multiple elements were considered. Here, material parameters identified in Table 4.1 for the dual cross-link self-healing hydrogel material were used.

5.1 Single-Element Simulations

Using a model with a single finite element, it is possible to test the developed `UMAT` and check for any implementation errors. For this, a uniaxial tension test mimicing the experimental data in Fig. 4.1 was simulated using one C3D8H[1] element of size 10mm x 10mm x 10mm.

Firstly, the element was fixed ($u_x, u_y, u_z = 0$) on one end ($x = 0$) and a ramp displacement of 10mm was applied in the x-direction with a constant displacement rate on the other end ($x = 10$). The constant displacement rate

$$\dot{u}_x = \mathrm{L}_0 \dot{\lambda_x} \tag{5.1}$$

was derived from the stretch rate given the fact that

$$\lambda_x = 1 + \frac{u_x|_{x=10}}{\mathrm{L}_0} \tag{5.2}$$

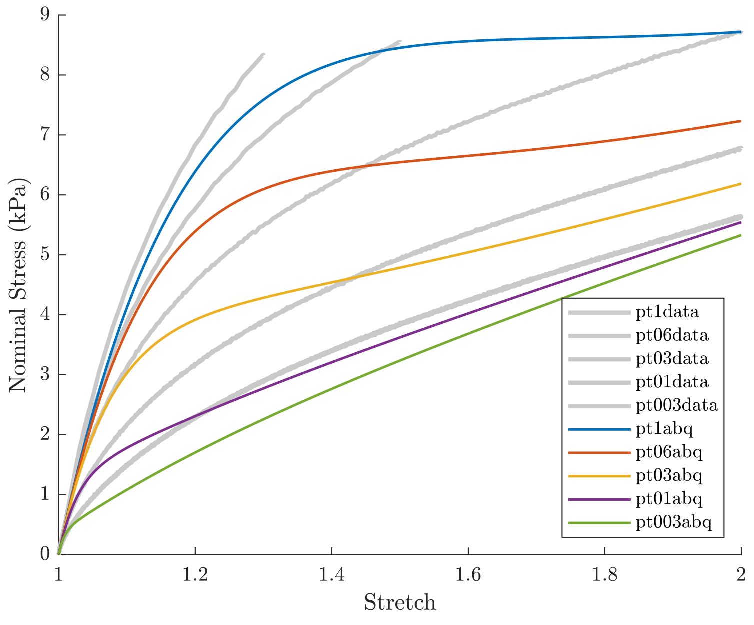
where $L_0 = 10$. In order to compare the results with the experimental data and the twin implementation in MATLAB, the simulations were carried

[1] 8-noded 3D continuum element with hybrid formulation in ABAQUS

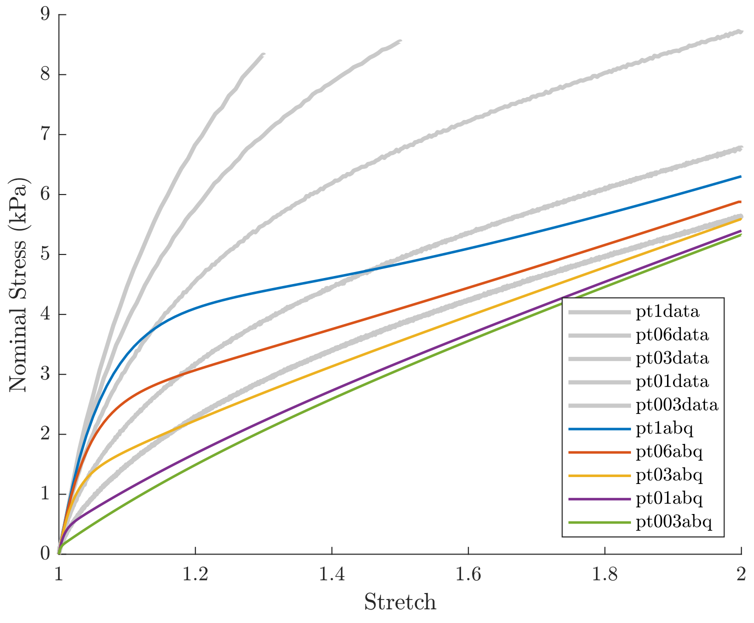
out for stretch rates ranging from $0.1s^{-1}$ to $0.003s^{-1}$. Furthermore, the implemented viscoelastic material models with, both, two ($k = 2$) and three ($k = 3$) relaxation mechanisms were used. It was found that, the response of the material model was affected by the stretch rates as seen in Fig. 5.1, which was as expected. However, for both the models (2EL and 3EL), the material response was not in agreement with the experimental data. The reason for this contradiction could either be the fact that, ABAQUS implements logarithmic strains for problems with large deformation, or, that the finite element formulation used in ABAQUS differs from the purely analytical approach that was used to calculate the material response in MATLAB during the parameter identification procedure. It is also possible that experimental data from multiple loading cases is required for identifying the optimal set of parameters[5]. Nevertheless, this needs further investigation. One possible way to avoid this deviation would be to use ABAQUS to calculate the material response instead of MATLAB for the cost function in Eq. (4.11). This, however, is also not dealt with in this work.

Next, the single element model was subjected to uniaxial loading followed by unloading. In this case, the element was fixed ($u_x, u_y, u_z = 0$) on one end ($x = 0$). On the other end ($x = 10$) a ramp displacement of 10mm followed by a negative ramp displacement of 10mm was applied in the x-direction with a constant displacement rate given by Eq. (5.1). Here, again the simulations were carried out for stretch rates varying from $0.1s^{-1}$ to $0.003s^{-1}$. For this simulation, only the viscoelastic material model with two relaxation mechanisms ($k = 2$) was used. The material response for the simulations is given in Fig. 5.2. Here, it can be seen that the material response is generally affected by the stretch rates. Furthermore, during unloading, the nominal stress $\mathrm{P}_{11} = 0$ for stretch $\lambda_1 > 1$. This effect is desired and is also in agreement with the material model developed by Long et al. [7] for the hydrogel material at hand.

Finally, the single element model was subjected to uniaxial loading and relaxation. Here, again the element was fixed ($u_x, u_y, u_z = 0$) on one end ($x = 0$). On the other end ($x = 10$), a ramp displacement of 10mm was applied in the x-direction with a constant displacement rate given by Eq. (5.1). At the end of this step, the displacement was held constant at 10mm and the material was allowed to relax. For this simulation too, only the implemented viscoelastic material model with two relaxation mechanisms ($k = 2$) was used. The material response for the simulations is given in Fig. 5.3. Here during the relaxation phase, it can be seen that after certain time, the stress $\boldsymbol{\sigma}$, and hence, $\mathbf{P}$ within the element reduces to the equillibrium stress $\boldsymbol{\sigma}_{\mathrm{EQ}}$, and hence $\mathbf{P}_{\mathrm{EQ}}$, which is expected behaviour for a viscoelastic material model.



(a)



(b)

Figure 5.1: Uniaxial tension test simulation in ABAQUS using viscoelastic material model (a) with two relaxation mechanisms ($k = 2$) and (b) with three relaxation mechanisms ($k = 3$) and respective material parameters $\boldsymbol{x}_p$ for 2EL_{NR}, 3EL_{NR} taken from Table 4.1

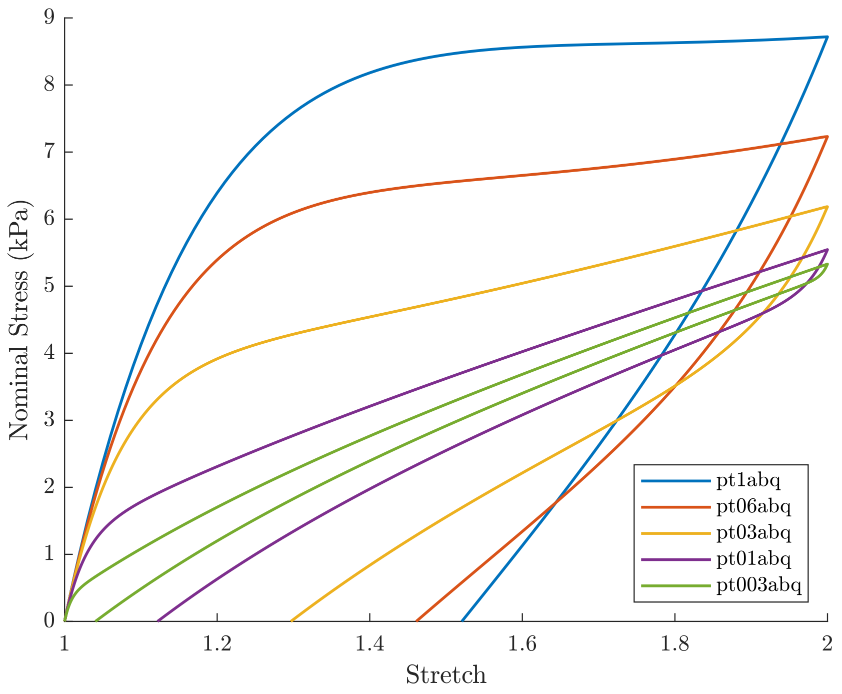


Figure 5.2: Uniaxial loading-unloading simulation in ABAQUS for viscoelastic material model with two relaxation mechanisms ($k = 2$) using material parameters $\boldsymbol{x}_p$ for 2EL$_{\mathrm{NR}}$ (Table 4.1)

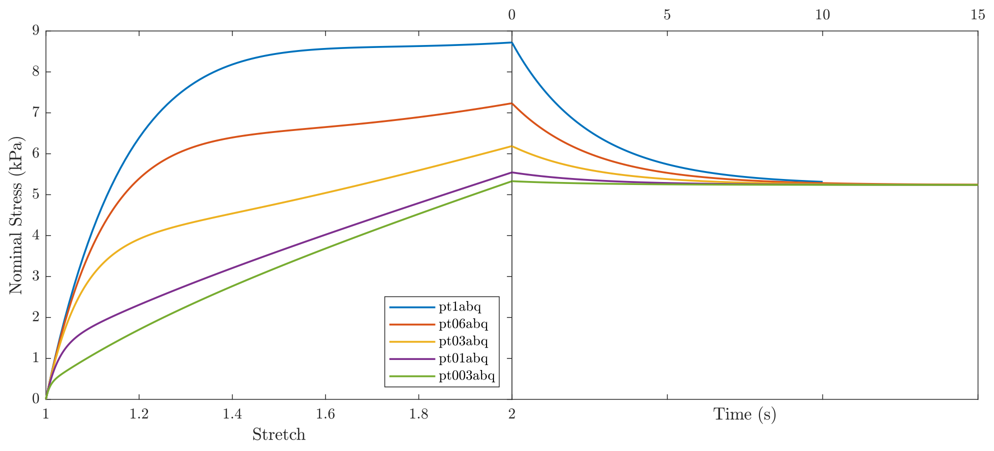


Figure 5.3: Uniaxial loading-relaxation simulation for viscoelastic material model in ABAQUS with two relaxation mechanisms ($k = 2$) using material parameters $\boldsymbol{x}_p$ for 2EL$_{\mathrm{NR}}$ (Table 4.1)

5.2 Multi-Element Simulations

Now that the implemented viscoelastic material models demonstrated the expected behaviour in finite element models involving a single element, they were further tested with finite element models involving multiple elements. To this end, a column twisting simulation with a pressure-follower load as described in Fig. 5.4 was employed[1]. The column was 6mm tall and had a cross section of 1mm x 1mm. An additional section of 3mm x 1mm x 1mm on the top was used to apply the pressure-follower load. The response of the model was then observed at the point $A \equiv (1.0, 0.0, 6.5)$. The geometry was discretized with a mesh size of 0.25mm or 0.1mm resulting in 576 and 9000 elements respectively. Furthermore, both C3D8H and C3D20H[1] element formulations were employed.

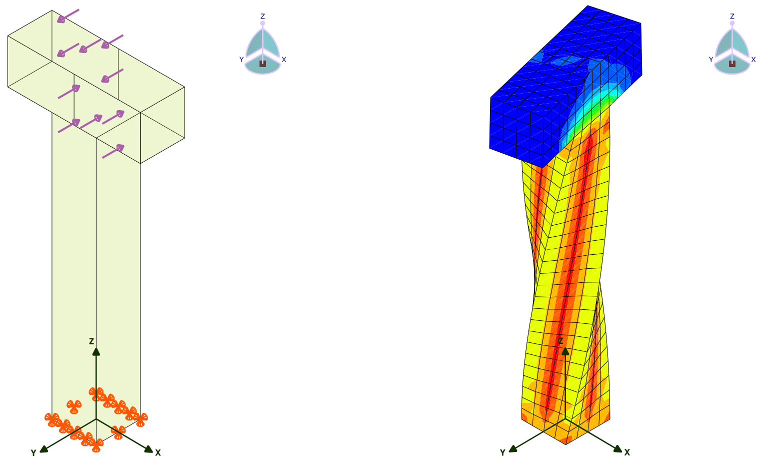
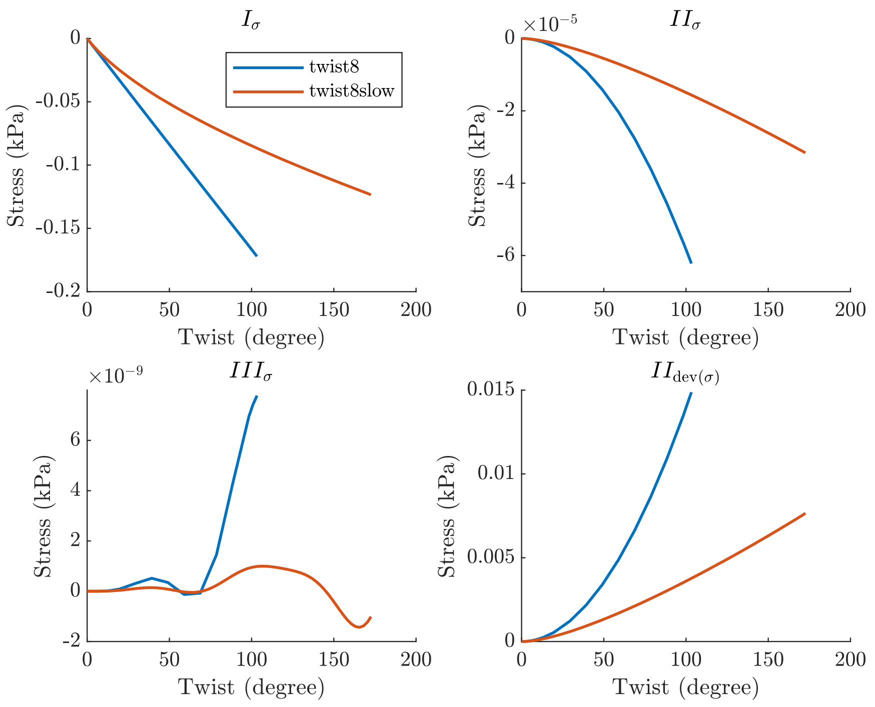


Figure 5.4: Column twisting simulation model definition using pressure-follower load (left) and exemplary deformed shape (right)

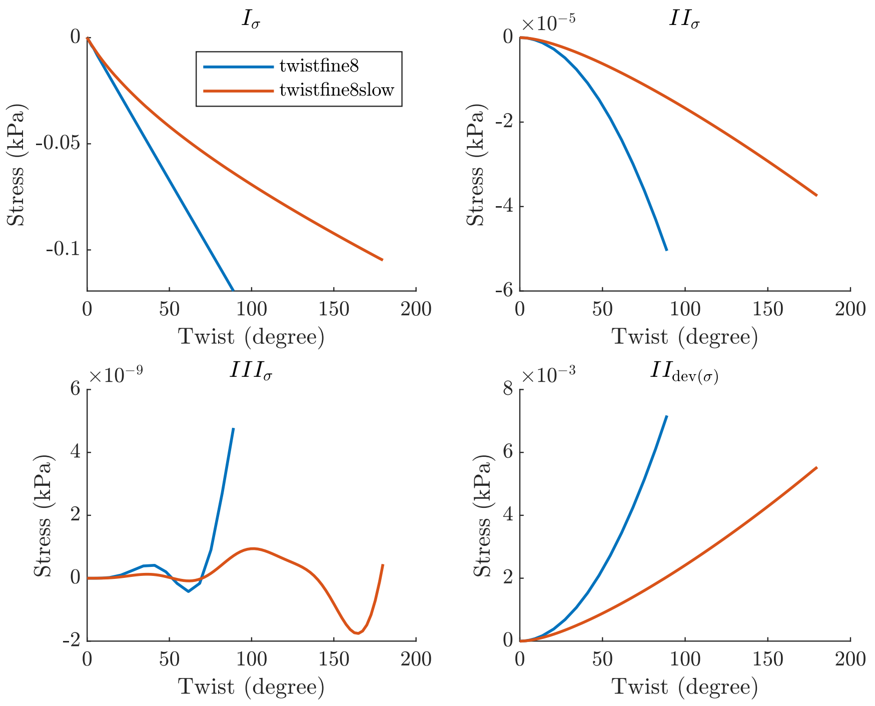
The response of the column twisting simulations is as shown in Fig. 5.5. In the case of *slow* simulations, the pressure was ramped at a rate of 2.5 kPa/s and 1.5 kPa/s for simulations with C3D8H and C3D20H elements respectively. For all other simulations, the pressure was ramped at a rate of 1 MPa/s. It is evident from the results that the invariants of the Cauchy stress tensor $I_{\boldsymbol{\sigma}}, II_{\boldsymbol{\sigma}}, III_{\boldsymbol{\sigma}}$ and $II_{\text{dev}(\boldsymbol{\sigma})}$ are affected by the rate of deformation or the rate of pressure application. The stress invariants for simulations with slower pressure application rate were always lower. This is because at slower rates of deformation or pressure application, the contribution of the non-equillibrium stress $\boldsymbol{\sigma}_{\text{NEQ}}$ to the total stress $\boldsymbol{\sigma}$ is lower, which is also the expected behaviour.

[1]20-noded 3D continuum element with hybrid formulation in ABAQUS



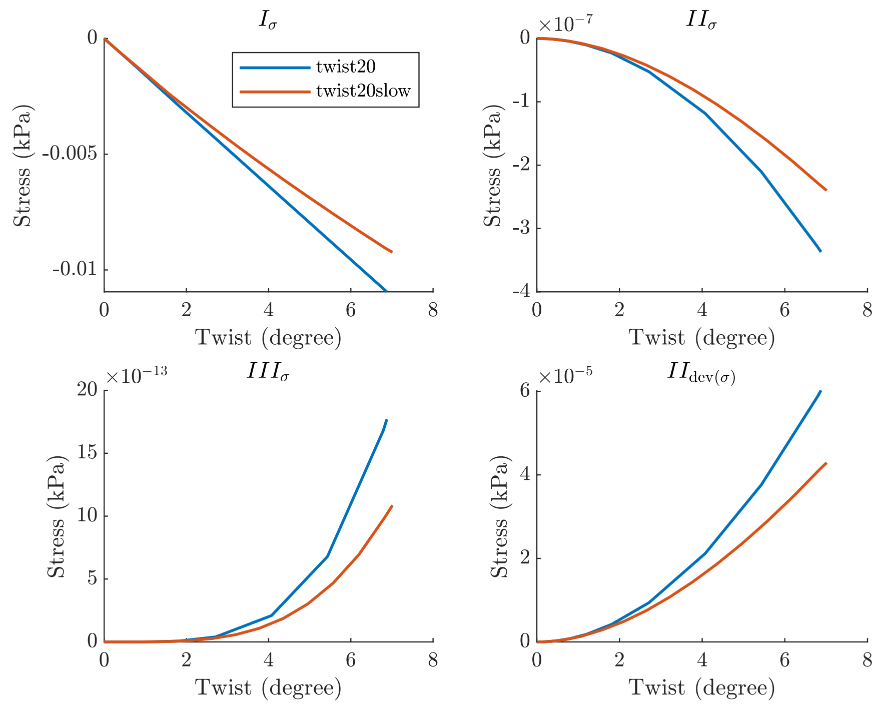
I_σ
twist8
twist8slow
II_σ
$\times 10^{-5}$
III_σ
$\times 10^{-9}$
$II_{\mathrm{dev}(\sigma)}$
Stress (kPa)
Twist (degree)

(a)



I_σ
twistfine8
twistfine8slow
II_σ
$\times 10^{-5}$
III_σ
$\times 10^{-9}$
$II_{\mathrm{dev}(\sigma)}$
$\times 10^{-3}$
Stress (kPa)
Twist (degree)

(b)



(c)

Figure 5.5: Column twisting simulation results in ABAQUS discretized with (a) 576 C3D8H, (b) 9000 C3D8H and (c) 576 C3D20H elements using viscoelastic material model with two relaxation mechanisms ($k = 2$) and material parameters $\boldsymbol{x}_p$ for $2\mathrm{EL}_{\mathrm{NR}}$ (Table 4.1)

Chapter 6

Conclusion

Hydrogels are being increasingly used in biomedical applications. It has therefore become important to understand their characteristics. In order to understand the suitability of hydrogels to a specific application, it is convenient to be able to regenerate the material response in a simulation with the help of a material model. Various mathematical models have been developed over the years to describe the characteristics of hydrogels. However, these have been based on the underlying physical and chemical phenomena. Since, hydrogels also exhibit viscoelastic properties, it was sought to characterize the the hydrogels using the finite viscoelasticity theory in this work. For this, user material (`UMAT`) subroutines for finite viscoelasticity theory using the Ogden class of strain energy functions with one, two and three relaxation mechanisms were implemented. The necessary theory required to implement the material model as well as the preliminary theory of continuum mechanics was briefly summarized. Furthermore, the step-by-step implementation details for the `UMAT` were also explained.

Thereafter, so that the implemented material model could characterize a hydrogel material, parameter identification was carried out using a non-linear least square optimization procedure along with a twin implementation of the `UMAT` in MATLAB. Here it was found that, the `UMAT` with a single relaxation mechanism was not able to replicate the material response of the hydrogel material from the experimental data. Additionally, although the optimal set parameters for `UMAT`s with two and three relaxation mechanisms were identified, it was found later on, that the models with these parameters generated a material response in ABAQUS simulations which disagreed with the experimental data. However, this does not altogether disregard the use of the finite viscoelasticity theory to characterize hydrogel materials, but suggests towards finding another approach for the parameter identification procedure. For future work in this regard, as an alternate, it could be possible to use the material response from the finite element simulation in ABAQUS for calculation of the cost function in the optimization procedure. Also, for material models that employ Ogden class of strain energy functions, it is recommended to perform the parameter identification procedure using experimental data from additional load cases, viz. biaxial tension and pure shear. Lastly, as an improvement, the `UMAT` could also be optimized to reduce the number of memory allocations for local variables in the models with two or more relaxation mechanisms to make the subroutine faster and efficient.

Bibliography

[1] J. Bonet, A. J. Gil, and R. Ortigosa. “A computational framework for polyconvex large strain elasticity”. In: *Computer Methods in Applied Mechanics and Engineering* 283 (Jan. 2015).

[2] Dassault Systèmes Simulia Corp. *ABAQUS Analysis User’s Guide v.6.14*. 2014.

[3] M. Itskov. *Lecture notes on Continuum Mechanics.* 2020.

[4] M. Itskov. *Tensor Algebra and Tensor Analysis for Engineers.* Cham, Switzerland: Springer International Publishing, 2018.

[5] B. Kleuter, A. Menzel, and P. Steinmann. “Generalized parameter identification for finite viscoelasticity”. In: *Computer Methods in Applied Mechanics and Engineering* 196.35 (July 2007).

[6] J. Kopp. “Efficient numerical diagonalization of hermitian 3x3 matrices”. In: *arXiv* (Oct. 2006).

[7] R. Long et al. “Time Dependent Behavior of a Dual Cross-Link Self-Healing Gel: Theory and Experiments”. In: *Macromolecules* 47.20 (Oct. 2014).

[8] S. Reese and S. Govindjee. “A theory of finite viscoelasticity and numerical aspects”. In: *Int. J. Solids Struct.* 35.26 (Sept. 1998).

[9] C. Truesdell and W. Noll. *The Non-Linear Field Theories of Mechanics.* Berlin, Germany: Springer, 1965.

[10] P. Wriggers. *Nonlinear Finite Element Methods.* Springer.

[11] O. C. Zienkiewicz, R. L. Taylor, and D. Fox. *The Finite Element Method for Solid and Structural Mechanics.* Oxford, England, UK: Butterworth-Heinemann, 2014.

Appendix A

UMAT Source Code

Below is the source code implementation of the viscoelastic material model with two relaxation mechanisms (viscous elements). Here, the steps described in Section 3.4 are followed, which is evident from the comments in the source.

```
!------------------------------------------------------------------------
!    UMAT FOR FINITE VISCOELASTIC THEORY - REESE & GOVINDJEE
!    WITH OGDEN HYPERELASTICITY MODEL
!    IMPLEMENTED AS PART OF MINI THESIS
!    - ALAN J CORREA
!------------------------------------------------------------------------
!    PROPS(1) - MU
!    PROPS(2) - ALPHA
!    PROPS(3) - KELAS
!    PROPS(4) - MUVIS
!    PROPS(5) - ALPHAVIS
!    PROPS(6) - KVIS
!    PROPS(7) - ETADEV
!    PROPS(8) - ETAVOL
!    PROPS(9) - MUVIS_2
!    PROPS(10) - ALPHAVIS_2
!    PROPS(11) - KVIS_2
!    PROPS(12) - ETADEV_2
!    PROPS(13) - ETAVOL_2
!------------------------------------------------------------------------
      SUBROUTINE UMAT(STRESS, STATEV, DDSDDE, SSE, SPD, SCD, RPL,
     1 DDSDDT, DRPLDE, DRPLDT, STRAN, DSTRAN, TIME, DTIME, TEMP, DTEMP,
     2 PREDEF, DPRED, CMNAME, NDI, NSHR, NTENS, NSTATV, PROPS, NPROPS,
     3 COORDS, DROT, PNEWDT, CELENT, DFGRD0, DFGRD1, NOEL, NPT, LAYER,
     4 KSPT, KSTEP, KINC)
!
      INCLUDE 'ABA_PARAM.INC'
!
      CHARACTER*8 CMNAME
      DIMENSION STRESS(NTENS), STATEV(NSTATV), DDSDDE(NTENS, NTENS),
     1 DDSDDT(NTENS), DRPLDE(NTENS), STRAN(NTENS), DSTRAN(NTENS),
     2 PREDEF(1), DPRED(1), PROPS(NPROPS), COORDS(3), DROT(3, 3),
     3 DFGRD0(3, 3), DFGRD1(3, 3)

!
! ----------------------------------------------------------------------
!     LOCAL ARRAYS
! ----------------------------------------------------------------------
!     BeOLD          - VISCOUS LEFT CAUCHY-GREEN DEFORMATION TENSOR (AT N-1)
!     BeOLDINV       - INVERSE(BeOLD)
!     CiOLD          - VISCOUS RIGHT CAUCHY-GREEN DEFORMATION TENSOR (AT N-1)
!     CiOLDINV       - INVERSE(CiOLD)
!     BeTR           - NEQ TRIAL LEFT CAUCHY-GREEN TENSOR
!     BeTR_          - COPY OF BeTR FOR EIGEN VALUE\VECTOR DECOMPOSITION
!     Be             - NEQ LEFT CAUCHY-GREEN TENSOR
!     TAUNEQ         - NEQ KIRCHOFF STRESS TENSOR
!     PVBeTR         - EIGEN VALUES OF BeTR
!     PDBeTR         - EIGEN VECTORS OF BeTR
!     EPSeTR         - TRIAL NEQ PRINCIPAL(EIGEN) STRAIN
!     PVBe           - EIGEN VALUES OF Be
!     EPSe           - NEQ LOGARITHMIC PRINCIPAL(EIGEN) STRAIN
!     PVBeBAR        - EIGEN VALUES OF NEQ DEVIATORIC LEFT-CAUCHY GREEN TENSOR
!     DEVTAU         - EIGEN VALUES OF DEVIATORIC NEQ KIRCHOFF STRESS TENSOR
!     RESVEC         - RESIDUAL VECTOR FOR NEWTON RAPHSON'S ITERATION
!     DDEVTAUDEPSe   - D(DEVTAU) / D(EPSe)
!     KMAT           - K MATRIX FOR NEWTON RAPHSON'S ITERATION
!     KINV           - INVERSE OF K MATRIX
!     DELEPSe        - DELTA EPSe FOR NEWTON RAPHSON'S ITERATION
!     PVTAU          - EIGEN VALUES OF TAUNEQ
!     PDTAU          - EIGEN VECTORS OF TAUNEQ
!     DPVTAUDEPSe    - D(PVTAU) / D(EPSe)
!     CALG           - NEQ ALGORITHMIC TANGENT MODULUS
!     SIGMANEQ       - NEQ CAUCHY STRESS TENSOR
!     L4NEQ          - NEQ INTERMEDIATE MATERIAL TANGENT STIFFNESS MODULUS
!     C4NEQ          - NEQ SPATIAL TANGENT STIFFNESS MODULUS
!     C4NEQJ         - NEQ SPATIAL TANGENT STIFFNESS MODULUS (FE)
!     CNEQ           - NEQ SPATIAL TANGENT STIFFNESS MODULUS (VOIGT)
!
!     ##### SECOND VISCOUS ELEMENT
!     BeOLD_2        - VISCOUS LEFT CAUCHY-GREEN DEFORMATION TENSOR (AT N-1)
!     BeOLDINV_2     - INVERSE(BeOLD)
!     CiOLD_2        - VISCOUS RIGHT CAUCHY-GREEN DEFORMATION TENSOR (AT N-1)
!     CiOLDINV_2     - INVERSE(CiOLD)
!     BeTR_2         - NEQ TRIAL LEFT CAUCHY-GREEN TENSOR
!     BeTR__2        - COPY OF BeTR FOR EIGEN VALUE\VECTOR DECOMPOSITION
!     Be_2           - NEQ LEFT CAUCHY-GREEN TENSOR
!     TAUNEQ_2       - NEQ KIRCHOFF STRESS TENSOR
!     PVBeTR_2       - EIGEN VALUES OF BeTR
!     PDBeTR_2       - EIGEN VECTORS OF BeTR
!     EPSeTR_2       - TRIAL NEQ PRINCIPAL(EIGEN) STRAIN
!     PVBe_2         - EIGEN VALUES OF Be
!     EPSe_2         - NEQ LOGARITHMIC PRINCIPAL(EIGEN) STRAIN
!     PVBeBAR_2      - EIGEN VALUES OF NEQ DEVIATORIC LEFT-CAUCHY GREEN TENSOR
!     DEVTAU_2       - EIGEN VALUES OF DEVIATORIC NEQ KIRCHOFF STRESS TENSOR
!     RESVEC_2       - RESIDUAL VECTOR FOR NEWTON RAPHSON'S ITERATION
!     DDEVTAUDEPSe_2- D(DEVTAU) / D(EPSe)
!     KMAT_2         - K MATRIX FOR NEWTON RAPHSON'S ITERATION
!     KINV_2         - INVERSE OF K MATRIX
!     DELEPSe_2      - DELTA EPSe FOR NEWTON RAPHSON'S ITERATION
!     PVTAU_2        - EIGEN VALUES OF TAUNEQ
!     PDTAU_2        - EIGEN VECTORS OF TAUNEQ
!     DPVTAUDEPSe_2  - D(PVTAU) / D(EPSe)
!     CALG_2         - NEQ ALGORITHMIC TANGENT MODULUS
!     SIGMANEQ_2     - NEQ CAUCHY STRESS TENSOR
!     L4NEQ_2        - NEQ INTERMEDIATE MATERIAL TANGENT STIFFNESS MODULUS
!     C4NEQ_2        - NEQ SPATIAL TANGENT STIFFNESS MODULUS
!     C4NEQJ_2       - NEQ SPATIAL TANGENT STIFFNESS MODULUS (FE)
!     CNEQ_2         - NEQ SPATIAL TANGENT STIFFNESS MODULUS (VOIGT)
!
!     BTOT           - EQ LEFT CAUCHY-GREEN TENSOR
!     BTOT_          - COPY OF BTOT FOR EIGEN VALUE/VECTOR DECOMPOSITION
!     PVBTOT         - EIGEN VALUES OF BTOT
!     PDBTOT         - EIGEN VECTORS OF BTOT
!     PVBBAR         - EIGEN VALUES OF EQ DEVIATORIC LEFT CAUCHY-GREEN TENSOR
!     PVTAUEQ        - EIGEN VALUES OF EQ KIRCHOFF STRESS TENSOR
!     SIGMAEQ        - EQ CAUCHY STRESS TENSOR
!     CAB            - CAB FOR EQ SPATIAL TANGENT STIFFNESS MODULUS
!     GAB            - GAB FOR EQ SPATIAL TANGENT STIFFNESS MODULUS
!     C4EQMAT        - COEFFICIENTS OF EQ SPATIAL TANGENT STIFFNESS MODULUS
!     C4EQ           - EQ SPATIAL TANGENT STIFFNESS MODULUS
!     C4EQJ          - EQ SPATIAL TANGENT STIFFNESS MODULUS (FE)
!     CEQ            - EQ SPATIAL TANGENT STIFFNESS MODULUS (VOIGT)
!     STRESSTOT      - TOTAL CAUCHY STRESS
!     IDT2           - 2ND ORDER IDENTTITY TENSOR
!  ---------------------------------------------------------------------
```

```
DOUBLE PRECISION, DIMENSION(3,3) :: IDT2

DOUBLE PRECISION, DIMENSION(3, 3) :: BeOLD
DOUBLE PRECISION, DIMENSION(3, 3) :: BeOLDINV
DOUBLE PRECISION, DIMENSION(3, 3) :: CiOLD
DOUBLE PRECISION, DIMENSION(3, 3) :: CiOLDINV
DOUBLE PRECISION, DIMENSION(3, 3) :: BeTR
DOUBLE PRECISION, DIMENSION(3, 3) :: BeTR_
DOUBLE PRECISION, DIMENSION(3, 3) :: Be
DOUBLE PRECISION, DIMENSION(3,3) :: TAUNEQ
DOUBLE PRECISION, DIMENSION(3) :: PVBeTR
DOUBLE PRECISION, DIMENSION(3,3) :: PDBeTR
DOUBLE PRECISION, DIMENSION(3) :: EPSeTR
DOUBLE PRECISION, DIMENSION(3) :: PVBe
DOUBLE PRECISION, DIMENSION(3) :: EPSe
DOUBLE PRECISION, DIMENSION(3) :: PVBeBAR
DOUBLE PRECISION, DIMENSION(3) :: DEVTAU
DOUBLE PRECISION, DIMENSION(3) :: RESVEC
DOUBLE PRECISION, DIMENSION(3,3) :: DDEVTAUDEPSe
DOUBLE PRECISION, DIMENSION(3,3) :: KMAT
DOUBLE PRECISION, DIMENSION(3,3) :: KINV
DOUBLE PRECISION, DIMENSION(3) :: DELEPSe
DOUBLE PRECISION, DIMENSION(3) :: PVTAU
DOUBLE PRECISION, DIMENSION(3,3) :: PDTAU
DOUBLE PRECISION, DIMENSION(3,3) :: DPVTAUDEPSe
DOUBLE PRECISION, DIMENSION(3,3) :: CALG
DOUBLE PRECISION, DIMENSION(3,3) :: SIGMANEQ
DOUBLE PRECISION, DIMENSION(3,3,3,3) :: L4NEQ
DOUBLE PRECISION, DIMENSION(3,3,3,3) :: C4NEQ
DOUBLE PRECISION, DIMENSION(3,3,3,3) :: C4NEQJ
DOUBLE PRECISION, DIMENSION(6,6) :: CNEQ

DOUBLE PRECISION, DIMENSION(3, 3) :: BeOLD_2
DOUBLE PRECISION, DIMENSION(3, 3) :: BeOLDINV_2
DOUBLE PRECISION, DIMENSION(3, 3) :: CiOLD_2
DOUBLE PRECISION, DIMENSION(3, 3) :: CiOLDINV_2
DOUBLE PRECISION, DIMENSION(3, 3) :: BeTR_2
DOUBLE PRECISION, DIMENSION(3, 3) :: BeTR__2
DOUBLE PRECISION, DIMENSION(3, 3) :: Be_2
DOUBLE PRECISION, DIMENSION(3,3) :: TAUNEQ_2
DOUBLE PRECISION, DIMENSION(3) :: PVBeTR_2
DOUBLE PRECISION, DIMENSION(3,3) :: PDBeTR_2
DOUBLE PRECISION, DIMENSION(3) :: EPSeTR_2
DOUBLE PRECISION, DIMENSION(3) :: PVBe_2
DOUBLE PRECISION, DIMENSION(3) :: EPSe_2
DOUBLE PRECISION, DIMENSION(3) :: PVBeBAR_2
DOUBLE PRECISION, DIMENSION(3) :: DEVTAU_2
DOUBLE PRECISION, DIMENSION(3) :: RESVEC_2
DOUBLE PRECISION, DIMENSION(3,3) :: DDEVTAUDEPSe_2
DOUBLE PRECISION, DIMENSION(3,3) :: KMAT_2
DOUBLE PRECISION, DIMENSION(3,3) :: KINV_2
DOUBLE PRECISION, DIMENSION(3) :: DELEPSe_2
DOUBLE PRECISION, DIMENSION(3) :: PVTAU_2
DOUBLE PRECISION, DIMENSION(3,3) :: PDTAU_2
DOUBLE PRECISION, DIMENSION(3,3) :: DPVTAUDEPSe_2
DOUBLE PRECISION, DIMENSION(3,3) :: CALG_2
DOUBLE PRECISION, DIMENSION(3,3) :: SIGMANEQ_2
DOUBLE PRECISION, DIMENSION(3,3,3,3) :: L4NEQ_2
DOUBLE PRECISION, DIMENSION(3,3,3,3) :: C4NEQ_2
DOUBLE PRECISION, DIMENSION(3,3,3,3) :: C4NEQJ_2
DOUBLE PRECISION, DIMENSION(6,6) :: CNEQ_2

DOUBLE PRECISION, DIMENSION(3,3) :: BTOT
DOUBLE PRECISION, DIMENSION(3,3) :: BTOT_
DOUBLE PRECISION, DIMENSION(3) :: PVBTOT
DOUBLE PRECISION, DIMENSION(3,3) :: PDBTOT
DOUBLE PRECISION, DIMENSION(3) :: PVBBAR
DOUBLE PRECISION, DIMENSION(3) :: PVTAUEQ
DOUBLE PRECISION, DIMENSION(3,3) :: SIGMAEQ
DOUBLE PRECISION, DIMENSION(3,3) :: CAB
DOUBLE PRECISION, DIMENSION(3,3) :: GAB
DOUBLE PRECISION, DIMENSION(3,3,3,3) :: C4EQMAT
DOUBLE PRECISION, DIMENSION(3,3,3,3) :: C4EQ
```

```
      DOUBLE PRECISION, DIMENSION(3,3,3,3) :: C4EQJ
      DOUBLE PRECISION, DIMENSION(6,6) :: CEQ
      DOUBLE PRECISION, DIMENSION(3,3) :: STRESSTOT
!
! ---------------------------------------------------------------------
!     LOCAL VARIABLES
! ---------------------------------------------------------------------
!     DET       - JACOBIAN OF TOTAL DEFORMATION GRADIENT
!     Je        - JACOBIAN OF NEQ DEFORMATION GRADIENT
!     NORMRES   - NORM OF RESIDUAL VECTOR
!     RESTOL    - TOLERANCE FOR NEWTON RHAPSON CONVERGENCE
!     EPS       - TOLERANCE FOR EQUIVALENCE OF FLOAT VALUES
!     ITER      - LOCAL ITERATION NUMBER
!
!     ##### SECOND VISCOUS ELEMENT
!     Je_2        - JACOBIAN OF NEQ DEFORMATION GRADIENT
!     NORMRES_2   - NORM OF RESIDUAL VECTOR
! ---------------------------------------------------------------------
      DOUBLE PRECISION DET

      DOUBLE PRECISION MU
      DOUBLE PRECISION ALPHA
      DOUBLE PRECISION KELAS
      DOUBLE PRECISION MUVIS
      DOUBLE PRECISION ALPHAVIS
      DOUBLE PRECISION KVIS
      DOUBLE PRECISION ETADEV
      DOUBLE PRECISION ETAVOL
      DOUBLE PRECISION Je
      DOUBLE PRECISION NORMRES

      DOUBLE PRECISION MUVIS_2
      DOUBLE PRECISION ALPHAVIS_2
      DOUBLE PRECISION KVIS_2
      DOUBLE PRECISION ETADEV_2
      DOUBLE PRECISION ETAVOL_2
      DOUBLE PRECISION Je_2
      DOUBLE PRECISION NORMRES_2

      INTEGER ITER, I, J, K, L, M, N
      LOGICAL OK_FLAG
      DOUBLE PRECISION :: RESTOL = 1.0D-12
      DOUBLE PRECISION :: EPS = 1.0D-12
!
      PARAMETER(ZERO=0.D0, ONE=1.D0, TWO=2.D0, THREE=3.D0, FOUR=4.D0,
     1 SIX=6.D0, NINE=9.0D0)
!
! ---------------------------------------------------------------------
!     STEP 1 - COMPUTING NECESSARY VARIABLES AND CONSTANTS
! ---------------------------------------------------------------------
!
!     DEFINE SECOND ORDER IDENTITY TENSOR
      IDT2 = ZERO
      DO I = 1,3
       IDT2(I,I) = ONE
      END DO
!
!     DEFINING INTERNAL STATE VARIABLE FOR FIRST ITERATION
      IF (KSTEP.EQ.1 .AND. KINC.EQ.1) THEN
            STATEV(1) = ONE
            STATEV(2) = ONE
            STATEV(3) = ONE
            STATEV(4) = ZERO
            STATEV(5) = ZERO
            STATEV(6) = ZERO
            STATEV(7) = ONE
            STATEV(8) = ONE
            STATEV(9) = ONE
            STATEV(10) = ZERO
            STATEV(11) = ZERO
            STATEV(12) = ZERO
      END IF
!
```

```
!      CALCULATE JACOBIAN OF DEFORMATION GRADIENT
       DET = DFGRD1(1,1)*DFGRD1(2,2)*DFGRD1(3,3)
     1      -DFGRD1(1,2)*DFGRD1(2,1)*DFGRD1(3,3)
       IF(NSHR.EQ.3) THEN
         DET=DET+DFGRD1(1,2)*DFGRD1(2,3)*DFGRD1(3,1)
     1          +DFGRD1(1,3)*DFGRD1(3,2)*DFGRD1(2,1)
     2          -DFGRD1(1,3)*DFGRD1(3,1)*DFGRD1(2,2)
     3          -DFGRD1(2,3)*DFGRD1(3,2)*DFGRD1(1,1)
       END IF
!
!      ##### FIRST VISCOUS ELEMENT
!
!      GET INTERNAL STATE-VARIABLE FROM PREVIOUS ITERATION
       BeOLD(1,1) = STATEV(1)
       BeOLD(1,2) = STATEV(4)
       BeOLD(1,3) = STATEV(5)
       BeOLD(2,1) = STATEV(4)
       BeOLD(2,2) = STATEV(2)
       BeOLD(2,3) = STATEV(6)
       BeOLD(3,1) = STATEV(5)
       BeOLD(3,2) = STATEV(6)
       BeOLD(3,3) = STATEV(3)
!
!      ASSIGN PROPERTIES TO CONSTANTS
       MU          = PROPS(1)
       ALPHA       = PROPS(2)
       KELAS       = PROPS(3)
       MUVIS       = PROPS(4)
       ALPHAVIS    = PROPS(5)
       KVIS        = PROPS(6)
       ETADEV      = PROPS(7)
       ETAVOL      = PROPS(8)
!
!      ##### SECOND VISCOUS ELEMENT
!
!      GET INTERNAL STATE-VARIABLE FROM PREVIOUS ITERATION
       BeOLD_2(1,1) = STATEV(7)
       BeOLD_2(1,2) = STATEV(10)
       BeOLD_2(1,3) = STATEV(11)
       BeOLD_2(2,1) = STATEV(10)
       BeOLD_2(2,2) = STATEV(8)
       BeOLD_2(2,3) = STATEV(12)
       BeOLD_2(3,1) = STATEV(11)
       BeOLD_2(3,2) = STATEV(12)
       BeOLD_2(3,3) = STATEV(9)
!
!      ASSIGN PROPERTIES TO CONSTANTS
       MUVIS_2       = PROPS(9)
       ALPHAVIS_2    = PROPS(10)
       KVIS_2        = PROPS(11)
       ETADEV_2      = PROPS(12)
       ETAVOL_2      = PROPS(13)
!
! --------------------------------------------------------------------
!      STEP 2 - CALCULATION OF TRIAL LEFT CAUCHY-GREEN TENSOR
! --------------------------------------------------------------------
!
!      ##### FIRST VISCOUS ELEMENT
!
       CALL M33INV(BeOLD, BeOLDINV, OK_FLAG)
       CiOLD = MATMUL(TRANSPOSE(DFGRD0), MATMUL(BeOLDINV, DFGRD0))
       CALL M33INV(CiOLD, CiOLDINV, OK_FLAG)
       BeTR  = MATMUL(DFGRD1, MATMUL(CiOLDINV, TRANSPOSE(DFGRD1)))
!
!      ##### SECOND VISCOUS ELEMENT
!
       CALL M33INV(BeOLD_2, BeOLDINV_2, OK_FLAG)
       CiOLD_2 = MATMUL(TRANSPOSE(DFGRD0), MATMUL(BeOLDINV_2, DFGRD0))
       CALL M33INV(CiOLD_2, CiOLDINV_2, OK_FLAG)
       BeTR_2  = MATMUL(DFGRD1, MATMUL(CiOLDINV_2, TRANSPOSE(DFGRD1)))
!
! --------------------------------------------------------------------
!      STEP 3 - LOCAL NEWTON RHAPSON FOR NEQ EVOLUTION EQUATION
! --------------------------------------------------------------------
!     IN:
!           BeTR
!           DTIME
!           MUVIS, ALPHAVIS, KVIS, ETADEV, ETAVOL
!
!     OUT:
!           Be, TAUNEQ
!           PVBeTR, PVTAU, PDTAU
!           CALG
! --------------------------------------------------------------------
!
!     ##### FIRST VISCOUS ELEMENT
!
!     GET EIGEN VALUES AND EIGEN VECTORS OF BeTR
      DO I = 1,3
        DO J = 1,3
          BeTR_(I,J) = BeTR(I,J)
        END DO
      END DO
      CALL DSYEVJ3(BeTR_, PDBeTR, PVBeTR)
      PDBeTR = TRANSPOSE(PDBeTR)
!
!     CALCULATE EPSeTR
      DO I = 1,3
        EPSeTR(I)=ONE/TWO*LOG(PVBeTR(I))
      END DO
!
!     PERFORM LOCAL ITERATION
!     1. Initialize Iteration Variables
      DO I = 1,3
        PVBe(I) = PVBeTR(I)
        EPSe(I) = EPSeTR(I)
      END DO
!     2. Newton Rhapsons Iteration with MAXITER=200
      DO ITER = 1, 200
!       - Calculating Jacobian
        Je = (PVBe(1)*PVBe(2)*PVBe(3))**(ONE/TWO)
!       - Calculating Deviatoric Principal Values of Be
        PVBeBAR(1) = Je**(-TWO/THREE)*PVBe(1)
        PVBeBAR(2) = Je**(-TWO/THREE)*PVBe(2)
        PVBeBAR(3) = Je**(-TWO/THREE)*PVBe(3)
!       - Calculating Principal Values of Deviatoric Kirchoff Stress
        DEVTAU(1) = MUVIS * ((TWO/THREE)*(PVBeBAR(1)**(ALPHAVIS/TWO))
     1                    -(ONE/THREE)*(PVBeBAR(2)**(ALPHAVIS/TWO))
     2                    -(ONE/THREE)*(PVBeBAR(3)**(ALPHAVIS/TWO)))
        DEVTAU(2) = MUVIS * ((TWO/THREE)*(PVBeBAR(2)**(ALPHAVIS/TWO))
     1                    -(ONE/THREE)*(PVBeBAR(3)**(ALPHAVIS/TWO))
     2                    -(ONE/THREE)*(PVBeBAR(1)**(ALPHAVIS/TWO)))
        DEVTAU(3) = MUVIS * ((TWO/THREE)*(PVBeBAR(3)**(ALPHAVIS/TWO))
     1                    -(ONE/THREE)*(PVBeBAR(1)**(ALPHAVIS/TWO))
     2                    -(ONE/THREE)*(PVBeBAR(2)**(ALPHAVIS/TWO)))
!       - Calculating Residual Vector
        DO N = 1, 3
          RESVEC(N) = EPSe(N) - EPSeTR(N)
     1              + DTIME * (ONE/(TWO*ETADEV)*DEVTAU(N)
     2                  +KVIS/(SIX*ETAVOL)*(Je*Je-ONE))
        END DO
!       - Calculating Norm of Residual Vector
        NORMRES = (RESVEC(1)**TWO
     1            +RESVEC(2)**TWO
     2            +RESVEC(3)**TWO)**(ONE/TWO)
!       - Terminate if Norm of Residual Vector is less than Tolerance
        IF ((ITER.GT.1).AND.ABS(NORMRES).LT.RESTOL) THEN
            EXIT
        END IF
!       - Calculating dDEVTAU/dEPSe
        DDEVTAUDEPSe(1,1) = MUVIS * ALPHAVIS
     1                 *((FOUR/NINE)*(PVBeBAR(1)**(ALPHAVIS/TWO))
     2                   +(ONE/NINE)*(PVBeBAR(2)**(ALPHAVIS/TWO)
     3                               +PVBeBAR(3)**(ALPHAVIS/TWO)))
        DDEVTAUDEPSe(2,2) = MUVIS * ALPHAVIS
     1                 *((FOUR/NINE)*(PVBeBAR(2)**(ALPHAVIS/TWO))
     2                      +(ONE/NINE)*(PVBeBAR(3)**(ALPHAVIS/TWO)
     3                                  +PVBeBAR(1)**(ALPHAVIS/TWO)))
        DDEVTAUDEPSe(3,3) = MUVIS * ALPHAVIS
     1                    *((FOUR/NINE)*(PVBeBAR(3)**(ALPHAVIS/TWO))
     2                      +(ONE/NINE)*(PVBeBAR(1)**(ALPHAVIS/TWO)
     3                                  +PVBeBAR(2)**(ALPHAVIS/TWO)))
        DDEVTAUDEPSe(1,2) = MUVIS * ALPHAVIS
     1                    *((-TWO/NINE)*(PVBeBAR(1)**(ALPHAVIS/TWO)
     2                                  +PVBeBAR(2)**(ALPHAVIS/TWO))
     3                      +(ONE/NINE)*(PVBeBAR(3)**(ALPHAVIS/TWO)))
        DDEVTAUDEPSe(1,3) = MUVIS * ALPHAVIS
     1                    *((-TWO/NINE)*(PVBeBAR(1)**(ALPHAVIS/TWO)
     2                                  +PVBeBAR(3)**(ALPHAVIS/TWO))
     3                      +(ONE/NINE)*(PVBeBAR(2)**(ALPHAVIS/TWO)))
        DDEVTAUDEPSe(2,3) = MUVIS * ALPHAVIS
     1                    *((-TWO/NINE)*(PVBeBAR(2)**(ALPHAVIS/TWO)
     2                                  +PVBeBAR(3)**(ALPHAVIS/TWO))
     3                      +(ONE/NINE)*(PVBeBAR(1)**(ALPHAVIS/TWO)))
        DDEVTAUDEPSe(2,1) = DDEVTAUDEPSe(1,2)
        DDEVTAUDEPSe(3,1) = DDEVTAUDEPSe(1,3)
        DDEVTAUDEPSe(3,2) = DDEVTAUDEPSe(2,3)
!       - Calculating K Matrix for Newton Rhapson
        DO I = 1,3
          DO J = 1,3
            KMAT(I,J) = IDT2(I,J)
     1              + DTIME * ((ONE/(TWO*ETADEV))*DDEVTAUDEPSe(I,J)
     2                        -(ONE/(THREE*ETAVOL)*KVIS*Je*Je))
          END DO
        END DO
!       - Calculating KINV
        CALL M33INV(KMAT, KINV, OK_FLAG)
!       - Calculating NEQ Strain Increment
        DELEPSe(1) = -(KINV(1,1)*RESVEC(1)
     1               +KINV(1,2)*RESVEC(2)
     2               +KINV(1,3)*RESVEC(3))

        DELEPSe(2) = -(KINV(2,1)*RESVEC(1)
     1               +KINV(2,2)*RESVEC(2)
     2               +KINV(2,3)*RESVEC(3))

        DELEPSe(3) = -(KINV(3,1)*RESVEC(1)
     1               +KINV(3,2)*RESVEC(2)
     2               +KINV(3,3)*RESVEC(3))
!       - Calculating Updated NEQ Strain
        DO I = 1,3
        EPSe(I) = EPSe(I) + DELEPSe(I)
        END DO
!       - Updating Eigen Value of Be
        DO I = 1,3
          PVBe(I) = EXP(TWO*EPSe(I))
        END DO
      END DO
!     3. Non-Covergence Test for Newton Rhapson
      IF (ITER.GT.200) THEN
            WRITE(*,*) "LOCAL ITERATION DID NOT CONVERGE"
      END IF
!
!     CALCULATE REQUIRED QUANTITIES
!     1. Be
      DO I=1,3
        DO J=1,3
            Be(J,I)=PVBe(1)*PDBeTR(1,J)*PDBeTR(1,I)
     1             +PVBe(2)*PDBeTR(2,J)*PDBeTR(2,I)
     2             +PVBe(3)*PDBeTR(3,J)*PDBeTR(3,I)
        END DO
      END DO
!
!     2. PVTAU
      DO I=1,3
            PVTAU(I) = DEVTAU(I) + KVIS/TWO*(Je*Je-ONE)
      END DO
!
!     3. PDTAU
```

```
      DO I=1,3
        DO J=1,3
          PDTAU(J,I) = PDBeTR(J,I)
        END DO
      END DO
!
!     4. TAUNEQ
      DO I=1,3
        DO J=1,3
          TAUNEQ(J,I)=PVTAU(1)*PDTAU(1,J)*PDTAU(1,I)
     1               +PVTAU(2)*PDTAU(2,J)*PDTAU(2,I)
     2               +PVTAU(3)*PDTAU(3,J)*PDTAU(3,I)
        END DO
      END DO
!
!     5. CALG
!     - Calculating Calculating dPVTAU/dEPSe
      DO I=1,3
        DO J=1,3
          DPVTAUDEPSe(J,I) = DDEVTAUDEPSe(J,I) + KVIS*Je*Je
        END DO
      END DO
!     - Calculating CALG
      CALG = MATMUL(DPVTAUDEPSe, KINV)
!
!     ##### SECOND VISCOUS ELEMENT
!
!     GET EIGEN VALUES AND EIGEN VECTORS OF BeTR
      DO I = 1,3
        DO J = 1,3
          BeTR__2(I,J) = BeTR(I,J)
        END DO
      END DO
      CALL DSYEVJ3(BeTR__2, PDBeTR_2, PVBeTR_2)
      PDBeTR_2 = TRANSPOSE(PDBeTR_2)
!
!     CALCULATE EPSeTR
      DO I = 1,3
        EPSeTR_2(I)=ONE/TWO*LOG(PVBeTR_2(I))
      END DO
!
!     PERFORM LOCAL ITERATION
!     1. Initialize Iteration Variables
      DO I = 1,3
        PVBe_2(I) = PVBeTR_2(I)
        EPSe_2(I) = EPSeTR_2(I)
      END DO
!     2. Newton Rhapsons Iteration with MAXITER=200
      DO ITER = 1, 200
!       - Calculating Jacobian
        Je_2 = (PVBe_2(1)*PVBe_2(2)*PVBe_2(3))**(ONE/TWO)
!       - Calculating Deviatoric Principal Values of Be
        PVBeBAR_2(1) = Je_2**(-TWO/THREE)*PVBe_2(1)
        PVBeBAR_2(2) = Je_2**(-TWO/THREE)*PVBe_2(2)
        PVBeBAR_2(3) = Je_2**(-TWO/THREE)*PVBe_2(3)
!       - Calculating Principal Values of Deviatoric Kirchoff Stress
        DEVTAU_2(1) = MUVIS_2 *
     1                ((TWO/THREE)*(PVBeBAR_2(1)**(ALPHAVIS_2/TWO))
     2                -(ONE/THREE)*(PVBeBAR_2(2)**(ALPHAVIS_2/TWO))
     3                -(ONE/THREE)*(PVBeBAR_2(3)**(ALPHAVIS_2/TWO)))
        DEVTAU_2(2) = MUVIS_2 *
     1                ((TWO/THREE)*(PVBeBAR_2(2)**(ALPHAVIS_2/TWO))
     2                -(ONE/THREE)*(PVBeBAR_2(3)**(ALPHAVIS_2/TWO))
     3                -(ONE/THREE)*(PVBeBAR_2(1)**(ALPHAVIS_2/TWO)))
        DEVTAU_2(3) = MUVIS_2 *
     1                ((TWO/THREE)*(PVBeBAR_2(3)**(ALPHAVIS_2/TWO))
     2                -(ONE/THREE)*(PVBeBAR_2(1)**(ALPHAVIS_2/TWO))
     3                -(ONE/THREE)*(PVBeBAR_2(2)**(ALPHAVIS_2/TWO)))
!       - Calculating Residual Vector
        DO N = 1, 3
          RESVEC_2(N) = EPSe_2(N) - EPSeTR_2(N)
     1                + DTIME *
     2                 (ONE/(TWO*ETADEV_2)*DEVTAU_2(N)
     2                   + KVIS_2/(SIX*ETAVOL_2)*(Je_2*Je_2-ONE))
        END DO
!       - Calculating Norm of Residual Vector
        NORMRES_2 = (RESVEC_2(1)**TWO
     1            + RESVEC_2(2)**TWO
     2            + RESVEC_2(3)**TWO)**(ONE/TWO)
!       - Terminate if Norm of Residual Vector is less than Tolerance
        IF ((ITER.GT.1).AND.ABS(NORMRES_2).LT.RESTOL) THEN
            EXIT
        END IF
!       - Calculating dDEVTAU/dEPSe
        DDEVTAUDEPSe_2(1,1) = MUVIS_2 * ALPHAVIS_2
     1                 *((FOUR/NINE)*(PVBeBAR_2(1)**(ALPHAVIS_2/TWO))
     2                   +(ONE/NINE)*(PVBeBAR_2(2)**(ALPHAVIS_2/TWO)
     3                              +PVBeBAR_2(3)**(ALPHAVIS_2/TWO)))
        DDEVTAUDEPSe_2(2,2) = MUVIS_2 * ALPHAVIS_2
     1                 *((FOUR/NINE)*(PVBeBAR_2(2)**(ALPHAVIS_2/TWO))
     2                   +(ONE/NINE)*(PVBeBAR_2(3)**(ALPHAVIS_2/TWO)
     3                              +PVBeBAR_2(1)**(ALPHAVIS_2/TWO)))
        DDEVTAUDEPSe_2(3,3) = MUVIS_2 * ALPHAVIS_2
     1                 *((FOUR/NINE)*(PVBeBAR_2(3)**(ALPHAVIS_2/TWO))
     2                   +(ONE/NINE)*(PVBeBAR_2(1)**(ALPHAVIS_2/TWO)
     3                              +PVBeBAR_2(2)**(ALPHAVIS_2/TWO)))
        DDEVTAUDEPSe_2(1,2) = MUVIS_2 * ALPHAVIS_2
     1                 *((-TWO/NINE)*(PVBeBAR_2(1)**(ALPHAVIS_2/TWO)
     2                              +PVBeBAR_2(2)**(ALPHAVIS_2/TWO))
     3                   +(ONE/NINE)*(PVBeBAR_2(3)**(ALPHAVIS_2/TWO)))
        DDEVTAUDEPSe_2(1,3) = MUVIS_2 * ALPHAVIS_2
     1                 *((-TWO/NINE)*(PVBeBAR_2(1)**(ALPHAVIS_2/TWO)
     2                              +PVBeBAR_2(3)**(ALPHAVIS_2/TWO))
     3                   +(ONE/NINE)*(PVBeBAR_2(2)**(ALPHAVIS_2/TWO)))
        DDEVTAUDEPSe_2(2,3) = MUVIS_2 * ALPHAVIS_2
     1                 *((-TWO/NINE)*(PVBeBAR_2(2)**(ALPHAVIS_2/TWO)
     2                              +PVBeBAR_2(3)**(ALPHAVIS_2/TWO))
     3                   +(ONE/NINE)*(PVBeBAR_2(1)**(ALPHAVIS_2/TWO)))
        DDEVTAUDEPSe_2(2,1) = DDEVTAUDEPSe_2(1,2)
        DDEVTAUDEPSe_2(3,1) = DDEVTAUDEPSe_2(1,3)
        DDEVTAUDEPSe_2(3,2) = DDEVTAUDEPSe_2(2,3)
!       - Calculating K Matrix for Newton Rhapson
        DO I = 1,3
          DO J = 1,3
            KMAT_2(I,J) = IDT2(I,J)
     1              + DTIME *
     2               ((ONE/(TWO*ETADEV_2))*DDEVTAUDEPSe_2(I,J)
     2               -(ONE/(THREE*ETAVOL_2)*KVIS_2*Je_2*Je_2))
          END DO
        END DO
!       - Calculating KINV
        CALL M33INV(KMAT_2, KINV_2, OK_FLAG)
!       - Calculating NEQ Strain Increment
        DELEPSe_2(1) = -(KINV_2(1,1)*RESVEC_2(1)
     1                  +KINV_2(1,2)*RESVEC_2(2)
     2                  +KINV_2(1,3)*RESVEC_2(3))

        DELEPSe_2(2) = -(KINV_2(2,1)*RESVEC_2(1)
     1                  +KINV_2(2,2)*RESVEC_2(2)
     2                  +KINV_2(2,3)*RESVEC_2(3))

        DELEPSe_2(3) = -(KINV_2(3,1)*RESVEC_2(1)
     1                  +KINV_2(3,2)*RESVEC_2(2)
     2                  +KINV_2(3,3)*RESVEC_2(3))
!       - Calculating Updated NEQ Strain
        DO I = 1,3
        EPSe_2(I) = EPSe_2(I) + DELEPSe_2(I)
        END DO
!       - Updating Eigen Value of Be
        DO I = 1,3
          PVBe_2(I) = EXP(TWO*EPSe_2(I))
        END DO
      END DO
!     3. Non-Covergence Test for Newton Rhapson
      IF (ITER.GT.200) THEN
            WRITE(*,*) "LOCAL ITERATION DID NOT CONVERGE"
      END IF
!
!     CALCULATE REQUIRED QUANTITIES
!     1. Be
      DO I=1,3
        DO J=1,3
            Be_2(J,I)=PVBe_2(1)*PDBeTR_2(1,J)*PDBeTR_2(1,I)
     1               +PVBe_2(2)*PDBeTR_2(2,J)*PDBeTR_2(2,I)
     2               +PVBe_2(3)*PDBeTR_2(3,J)*PDBeTR_2(3,I)
        END DO
      END DO
!
!     2. PVTAU
      DO I=1,3
            PVTAU_2(I) = DEVTAU_2(I) + KVIS_2/TWO*(Je_2*Je_2-ONE)
      END DO
!
!     3. PDTAU
      DO I=1,3
        DO J=1,3
          PDTAU_2(J,I) = PDBeTR_2(J,I)
        END DO
      END DO
!
!     4. TAUNEQ
      DO I=1,3
        DO J=1,3
          TAUNEQ_2(J,I)=PVTAU_2(1)*PDTAU_2(1,J)*PDTAU_2(1,I)
     1                 +PVTAU_2(2)*PDTAU_2(2,J)*PDTAU_2(2,I)
     2                 +PVTAU_2(3)*PDTAU_2(3,J)*PDTAU_2(3,I)
        END DO
      END DO
!
!     5. CALG
!     - Calculating Calculating dPVTAU/dEPSe
      DO I=1,3
        DO J=1,3
          DPVTAUDEPSe_2(J,I) = DDEVTAUDEPSe_2(J,I) + KVIS_2*Je_2*Je_2
        END DO
      END DO
!     - Calculating CALG
      CALG_2 = MATMUL(DPVTAUDEPSe_2, KINV_2)
!
! ---------------------------------------------------------------------
!     STEP 4 - UPDATE INTERNAL STATE VARIABLE - Be
! ---------------------------------------------------------------------
!     SET INTERNAL STATE-VARIABLE FROM FINAL LOCAL ITERATION
      STATEV(1) = Be(1,1)
      STATEV(2) = Be(2,2)
      STATEV(3) = Be(3,3)
      STATEV(4) = Be(1,2)
      STATEV(5) = Be(1,3)
      STATEV(6) = Be(2,3)
      STATEV(7) = Be_2(1,1)
      STATEV(8) = Be_2(2,2)
      STATEV(9) = Be_2(3,3)
      STATEV(10) = Be_2(1,2)
      STATEV(11) = Be_2(1,3)
      STATEV(12) = Be_2(2,3)
! ---------------------------------------------------------------------
!     STEP 5 - CALCULATION OF NEQ CAUCHY STRESS & ELASTICITY TENSOR
! ---------------------------------------------------------------------
!
!     ##### FIRST VISCOUS ELEMENT
!
!     CALCULATE NEQ CAUCHY STRESS TENSOR

      SIGMANEQ = ZERO
      DO J=1,3
        DO K=1,3
          SIGMANEQ(J,K) = TAUNEQ(J,K)/DET
        END DO
      END DO
```

```fortran
!
!     CALCULATE MATERIAL TENSOR (L4NEQ) IN INTERMEDIATE CONFIGURATION
!     EXPRESSED IN EIGEN BASIS
      L4NEQ = ZERO
!     1. CALCULATE COEFFICIENTS FOR AABB
      DO J=1,3
        DO K=1,3
          L4NEQ(J,J,K,K) = (CALG(J,K) - PVTAU(J)*TWO*IDT2(J,K))
     1                   /(PVBeTR(J)*PVBeTR(K))
        END DO
      END DO
!     2. CALCULATE COEFFICIENTS FOR ABAB / ABBA
      DO J=1,3
        DO K=1,3
          IF (J.EQ.K) THEN
            CYCLE
          END IF
!         - In case of Equal PVBeTR, Use Alternate Form
          IF (ABS(PVBeTR(J)-PVBeTR(K)).LT.EPS) THEN
            L4NEQ(J,K,J,K) = (L4NEQ(J,J,J,J)-L4NEQ(J,J,K,K))/TWO
          ELSE
            L4NEQ(J,K,J,K) = ((PVTAU(K)/PVBeTR(K))-(PVTAU(J)/PVBeTR(J)))
     1                     /(PVBeTR(K)-PVBeTR(J))
          END IF
          L4NEQ(J,K,K,J) = L4NEQ(J,K,J,K)
        END DO
      END DO
!     CALCULATE MATERIAL TENSOR (C4NEQ) IN CURRENT CONFIGURATION
!     EXPRESSED IN STANDARD BASIS
      C4NEQ = ZERO
      DO J=1,3
        DO K=1,3
          DO L=1,3
            DO M=1,3
              C4NEQ(J,K,L,M) =
     1          L4NEQ(1,1,1,1)*PVBeTR(1)*PVBeTR(1)
     1                        *PDBeTR(1,J)*PDBeTR(1,K)
     1                        *PDBeTR(1,L)*PDBeTR(1,M)
     1         +L4NEQ(1,1,2,2)*PVBeTR(1)*PVBeTR(2)
     1                        *PDBeTR(1,J)*PDBeTR(1,K)
     1                        *PDBeTR(2,L)*PDBeTR(2,M)
     1         +L4NEQ(1,1,3,3)*PVBeTR(1)*PVBeTR(3)
     1                        *PDBeTR(1,J)*PDBeTR(1,K)
     1                        *PDBeTR(3,L)*PDBeTR(3,M)
     1         +L4NEQ(2,2,1,1)*PVBeTR(2)*PVBeTR(1)
     1                        *PDBeTR(2,J)*PDBeTR(2,K)
     1                        *PDBeTR(1,L)*PDBeTR(1,M)
     1         +L4NEQ(2,2,2,2)*PVBeTR(2)*PVBeTR(2)
     1                        *PDBeTR(2,J)*PDBeTR(2,K)
     1                        *PDBeTR(2,L)*PDBeTR(2,M)
     1         +L4NEQ(2,2,3,3)*PVBeTR(2)*PVBeTR(3)
     1                        *PDBeTR(2,J)*PDBeTR(2,K)
     1                        *PDBeTR(3,L)*PDBeTR(3,M)
     1         +L4NEQ(3,3,1,1)*PVBeTR(3)*PVBeTR(1)
     1                        *PDBeTR(3,J)*PDBeTR(3,K)
     1                        *PDBeTR(1,L)*PDBeTR(1,M)
     1         +L4NEQ(3,3,2,2)*PVBeTR(3)*PVBeTR(2)
     1                        *PDBeTR(3,J)*PDBeTR(3,K)
     1                        *PDBeTR(2,L)*PDBeTR(2,M)
     1         +L4NEQ(3,3,3,3)*PVBeTR(3)*PVBeTR(3)
     1                        *PDBeTR(3,J)*PDBeTR(3,K)
     1                        *PDBeTR(3,L)*PDBeTR(3,M)
     1         +L4NEQ(1,2,1,2)*PVBeTR(1)*PVBeTR(2)
     1                        *PDBeTR(1,J)*PDBeTR(2,K)
     1                        *PDBeTR(1,L)*PDBeTR(2,M)
     1         +L4NEQ(1,2,2,1)*PVBeTR(1)*PVBeTR(2)
     1                        *PDBeTR(1,J)*PDBeTR(2,K)
     1                        *PDBeTR(2,L)*PDBeTR(1,M)
     1         +L4NEQ(1,3,1,3)*PVBeTR(1)*PVBeTR(3)
     1                        *PDBeTR(1,J)*PDBeTR(3,K)
     1                        *PDBeTR(1,L)*PDBeTR(3,M)
     1         +L4NEQ(1,3,3,1)*PVBeTR(1)*PVBeTR(3)
     1                        *PDBeTR(1,J)*PDBeTR(3,K)
     1                          *PDBeTR(3,L)*PDBeTR(1,M)
     1           +L4NEQ(2,1,2,1)*PVBeTR(2)*PVBeTR(1)
     1                          *PDBeTR(2,J)*PDBeTR(1,K)
     1                          *PDBeTR(2,L)*PDBeTR(1,M)
     1           +L4NEQ(2,1,1,2)*PVBeTR(2)*PVBeTR(1)
     1                          *PDBeTR(2,J)*PDBeTR(1,K)
     1                          *PDBeTR(1,L)*PDBeTR(2,M)
     1           +L4NEQ(2,3,2,3)*PVBeTR(2)*PVBeTR(3)
     1                          *PDBeTR(2,J)*PDBeTR(3,K)
     1                          *PDBeTR(2,L)*PDBeTR(3,M)
     1           +L4NEQ(2,3,3,2)*PVBeTR(2)*PVBeTR(3)
     1                          *PDBeTR(2,J)*PDBeTR(3,K)
     1                          *PDBeTR(3,L)*PDBeTR(2,M)
     1           +L4NEQ(3,1,3,1)*PVBeTR(3)*PVBeTR(1)
     1                          *PDBeTR(3,J)*PDBeTR(1,K)
     1                          *PDBeTR(3,L)*PDBeTR(1,M)
     1           +L4NEQ(3,1,1,3)*PVBeTR(3)*PVBeTR(1)
     1                          *PDBeTR(3,J)*PDBeTR(1,K)
     1                          *PDBeTR(1,L)*PDBeTR(3,M)
     1           +L4NEQ(3,2,3,2)*PVBeTR(3)*PVBeTR(2)
     1                          *PDBeTR(3,J)*PDBeTR(2,K)
     1                          *PDBeTR(3,L)*PDBeTR(2,M)
     1           +L4NEQ(3,2,2,3)*PVBeTR(3)*PVBeTR(2)
     1                          *PDBeTR(3,J)*PDBeTR(2,K)
     1                          *PDBeTR(2,L)*PDBeTR(3,M)
            END DO
          END DO
        END DO
      END DO
!
!     CALCLUATE JAUMANN RATE FORM OF MATERIAL TENSOR
      C4NEQJ = ZERO
      DO I=1,3
        DO J=1,3
          DO K=1,3
            DO L=1,3
              C4NEQJ(I,J,K,L) = C4NEQ(I,J,K,L)/DET
     1                          +(IDT2(I,K)*TAUNEQ(J,L)
     1                           +IDT2(I,L)*TAUNEQ(J,K)
     1                           +IDT2(J,K)*TAUNEQ(I,L)
     1                           +IDT2(J,L)*TAUNEQ(I,K))/(TWO*DET)
            END DO
          END DO
        END DO
      END DO
!
!     GENERATE ABAQUS NEQ TANGENT STIFFNESS MATRIX (VOIGT NOTATION)
      CNEQ = ZERO
      CNEQ(1,1) = C4NEQJ(1,1,1,1)
      CNEQ(1,2) = C4NEQJ(1,1,2,2)
      CNEQ(1,3) = C4NEQJ(1,1,3,3)
      CNEQ(1,4) = C4NEQJ(1,1,1,2)
      CNEQ(1,5) = C4NEQJ(1,1,1,3)
      CNEQ(1,6) = C4NEQJ(1,1,2,3)
      CNEQ(2,1) = CNEQ(1,2)
      CNEQ(2,2) = C4NEQJ(2,2,2,2)
      CNEQ(2,3) = C4NEQJ(2,2,3,3)
      CNEQ(2,4) = C4NEQJ(2,2,1,2)
      CNEQ(2,5) = C4NEQJ(2,2,1,3)
      CNEQ(2,6) = C4NEQJ(2,2,2,3)
      CNEQ(3,1) = CNEQ(1,3)
      CNEQ(3,2) = CNEQ(2,3)
      CNEQ(3,3) = C4NEQJ(3,3,3,3)
      CNEQ(3,4) = C4NEQJ(3,3,1,2)
      CNEQ(3,5) = C4NEQJ(3,3,1,3)
      CNEQ(3,6) = C4NEQJ(3,3,2,3)
      CNEQ(4,1) = CNEQ(1,4)
      CNEQ(4,2) = CNEQ(2,4)
      CNEQ(4,3) = CNEQ(3,4)
      CNEQ(4,4) = C4NEQJ(1,2,1,2)
      CNEQ(4,5) = C4NEQJ(1,2,1,3)
      CNEQ(4,6) = C4NEQJ(1,2,2,3)
      CNEQ(5,1) = CNEQ(1,5)
```

```fortran
      CNEQ(5,2) = CNEQ(2,5)
      CNEQ(5,3) = CNEQ(3,5)
      CNEQ(5,4) = CNEQ(4,5)
      CNEQ(5,5) = C4NEQJ(1,3,1,3)
      CNEQ(5,6) = C4NEQJ(1,3,2,3)
      CNEQ(6,1) = CNEQ(1,6)
      CNEQ(6,2) = CNEQ(2,6)
      CNEQ(6,3) = CNEQ(3,6)
      CNEQ(6,4) = CNEQ(4,6)
      CNEQ(6,5) = CNEQ(5,6)
      CNEQ(6,6) = C4NEQJ(2,3,2,3)
!
!     ##### SECOND VISCOUS ELEMENT
!
!     CALCULATE NEQ CAUCHY STRESS TENSOR

      SIGMANEQ_2 = ZERO
      DO J=1,3
        DO K=1,3
          SIGMANEQ_2(J,K) = TAUNEQ_2(J,K)/DET
        END DO
      END DO
!
!     CALCULATE MATERIAL TENSOR (L4NEQ) IN INTERMEDIATE CONFIGURATION
!     EXPRESSED IN EIGEN BASIS
      L4NEQ_2 = ZERO
!     1. CALCULATE COEFFICIENTS FOR AABB
      DO J=1,3
        DO K=1,3
          L4NEQ_2(J,J,K,K) = (CALG_2(J,K) - PVTAU_2(J)*TWO*IDT2(J,K))
     1                      /(PVBeTR_2(J)*PVBeTR_2(K))
        END DO
      END DO
!     2. CALCULATE COEFFICIENTS FOR ABAB / ABBA
      DO J=1,3
        DO K=1,3
          IF (J.EQ.K) THEN
            CYCLE
          END IF
!         - In case of Equal PVBeTR, Use Alternate Form
          IF (ABS(PVBeTR_2(J)-PVBeTR_2(K)).LT.EPS) THEN
            L4NEQ_2(J,K,J,K) = (L4NEQ_2(J,J,J,J)-L4NEQ_2(J,J,K,K))/TWO
          ELSE
            L4NEQ_2(J,K,J,K) = ((PVTAU_2(K)/PVBeTR_2(K))
     1                         -(PVTAU_2(J)/PVBeTR_2(J)))
     1                        /(PVBeTR(K)-PVBeTR(J))
          END IF
          L4NEQ_2(J,K,K,J) = L4NEQ_2(J,K,J,K)
        END DO
      END DO
!     CALCULATE MATERIAL TENSOR (C4NEQ) IN CURRENT CONFIGURATION
!     EXPRESSED IN STANDARD BASIS
      C4NEQ_2 = ZERO
      DO J=1,3
        DO K=1,3
          DO L=1,3
            DO M=1,3
              C4NEQ_2(J,K,L,M) =
     1          L4NEQ_2(1,1,1,1)*PVBeTR_2(1)*PVBeTR_2(1)
     1                          *PDBeTR_2(1,J)*PDBeTR_2(1,K)
     1                          *PDBeTR_2(1,L)*PDBeTR_2(1,M)
     1         +L4NEQ_2(1,1,2,2)*PVBeTR_2(1)*PVBeTR_2(2)
     1                          *PDBeTR_2(1,J)*PDBeTR_2(1,K)
     1                          *PDBeTR_2(2,L)*PDBeTR_2(2,M)
     1         +L4NEQ_2(1,1,3,3)*PVBeTR_2(1)*PVBeTR_2(3)
     1                          *PDBeTR_2(1,J)*PDBeTR_2(1,K)
     1                          *PDBeTR_2(3,L)*PDBeTR_2(3,M)
     1         +L4NEQ_2(2,2,1,1)*PVBeTR_2(2)*PVBeTR_2(1)
     1                          *PDBeTR_2(2,J)*PDBeTR_2(2,K)
     1                          *PDBeTR_2(1,L)*PDBeTR_2(1,M)
     1         +L4NEQ_2(2,2,2,2)*PVBeTR_2(2)*PVBeTR_2(2)
     1                          *PDBeTR_2(2,J)*PDBeTR_2(2,K)
     1                          *PDBeTR_2(2,L)*PDBeTR_2(2,M)
     1         +L4NEQ_2(2,2,3,3)*PVBeTR_2(2)*PVBeTR_2(3)
     1                          *PDBeTR_2(2,J)*PDBeTR_2(2,K)
     1                          *PDBeTR_2(3,L)*PDBeTR_2(3,M)
     1         +L4NEQ_2(3,3,1,1)*PVBeTR_2(3)*PVBeTR_2(1)
     1                          *PDBeTR_2(3,J)*PDBeTR_2(3,K)
     1                          *PDBeTR_2(1,L)*PDBeTR_2(1,M)
     1         +L4NEQ_2(3,3,2,2)*PVBeTR_2(3)*PVBeTR_2(2)
     1                          *PDBeTR_2(3,J)*PDBeTR_2(3,K)
     1                          *PDBeTR_2(2,L)*PDBeTR_2(2,M)
     1         +L4NEQ_2(3,3,3,3)*PVBeTR_2(3)*PVBeTR_2(3)
     1                          *PDBeTR_2(3,J)*PDBeTR_2(3,K)
     1                          *PDBeTR_2(3,L)*PDBeTR_2(3,M)
     1         +L4NEQ_2(1,2,1,2)*PVBeTR_2(1)*PVBeTR_2(2)
     1                          *PDBeTR_2(1,J)*PDBeTR_2(2,K)
     1                          *PDBeTR_2(1,L)*PDBeTR_2(2,M)
     1         +L4NEQ_2(1,2,2,1)*PVBeTR_2(1)*PVBeTR_2(2)
     1                          *PDBeTR_2(1,J)*PDBeTR_2(2,K)
     1                          *PDBeTR_2(2,L)*PDBeTR_2(1,M)
     1         +L4NEQ_2(1,3,1,3)*PVBeTR_2(1)*PVBeTR_2(3)
     1                          *PDBeTR_2(1,J)*PDBeTR_2(3,K)
     1                          *PDBeTR_2(1,L)*PDBeTR_2(3,M)
     1         +L4NEQ_2(1,3,3,1)*PVBeTR_2(1)*PVBeTR_2(3)
     1                          *PDBeTR_2(1,J)*PDBeTR_2(3,K)
     1                          *PDBeTR_2(3,L)*PDBeTR_2(1,M)
     1         +L4NEQ_2(2,1,2,1)*PVBeTR_2(2)*PVBeTR_2(1)
     1                          *PDBeTR_2(2,J)*PDBeTR_2(1,K)
     1                          *PDBeTR_2(2,L)*PDBeTR_2(1,M)
     1         +L4NEQ_2(2,1,1,2)*PVBeTR_2(2)*PVBeTR_2(1)
     1                          *PDBeTR_2(2,J)*PDBeTR_2(1,K)
     1                          *PDBeTR_2(1,L)*PDBeTR_2(2,M)
     1         +L4NEQ_2(2,3,2,3)*PVBeTR_2(2)*PVBeTR_2(3)
     1                          *PDBeTR_2(2,J)*PDBeTR_2(3,K)
     1                          *PDBeTR_2(2,L)*PDBeTR_2(3,M)
     1         +L4NEQ_2(2,3,3,2)*PVBeTR_2(2)*PVBeTR_2(3)
     1                          *PDBeTR_2(2,J)*PDBeTR_2(3,K)
     1                          *PDBeTR_2(3,L)*PDBeTR_2(2,M)
     1         +L4NEQ_2(3,1,3,1)*PVBeTR_2(3)*PVBeTR_2(1)
     1                          *PDBeTR_2(3,J)*PDBeTR_2(1,K)
     1                          *PDBeTR_2(3,L)*PDBeTR_2(1,M)
     1         +L4NEQ_2(3,1,1,3)*PVBeTR_2(3)*PVBeTR_2(1)
     1                          *PDBeTR_2(3,J)*PDBeTR_2(1,K)
     1                          *PDBeTR_2(1,L)*PDBeTR_2(3,M)
     1         +L4NEQ_2(3,2,3,2)*PVBeTR_2(3)*PVBeTR_2(2)
     1                          *PDBeTR_2(3,J)*PDBeTR_2(2,K)
     1                          *PDBeTR_2(3,L)*PDBeTR_2(2,M)
     1         +L4NEQ_2(3,2,2,3)*PVBeTR_2(3)*PVBeTR_2(2)
     1                          *PDBeTR_2(3,J)*PDBeTR_2(2,K)
     1                          *PDBeTR_2(2,L)*PDBeTR_2(3,M)
            END DO
          END DO
        END DO
      END DO
!
!     CALCLUATE JAUMANN RATE FORM OF MATERIAL TENSOR
      C4NEQJ_2 = ZERO
      DO I=1,3
        DO J=1,3
          DO K=1,3
            DO L=1,3
              C4NEQJ_2(I,J,K,L) = C4NEQ_2(I,J,K,L)/DET
     1                        +(IDT2(I,K)*TAUNEQ_2(J,L)
     1                         +IDT2(I,L)*TAUNEQ_2(J,K)
     1                         +IDT2(J,K)*TAUNEQ_2(I,L)
     1                         +IDT2(J,L)*TAUNEQ_2(I,K))/(TWO*DET)
            END DO
          END DO
        END DO
      END DO
!
!     GENERATE ABAQUS NEQ TANGENT STIFFNESS MATRIX (VOIGT NOTATION)
      CNEQ_2 = ZERO
      CNEQ_2(1,1) = C4NEQJ_2(1,1,1,1)
      CNEQ_2(1,2) = C4NEQJ_2(1,1,2,2)
      CNEQ_2(1,3) = C4NEQJ_2(1,1,3,3)
      CNEQ_2(1,4) = C4NEQJ_2(1,1,1,2)
      CNEQ_2(1,5) = C4NEQJ_2(1,1,1,3)
      CNEQ_2(1,6) = C4NEQJ_2(1,1,2,3)
      CNEQ_2(2,1) = CNEQ_2(1,2)
      CNEQ_2(2,2) = C4NEQJ_2(2,2,2,2)
      CNEQ_2(2,3) = C4NEQJ_2(2,2,3,3)
      CNEQ_2(2,4) = C4NEQJ_2(2,2,1,2)
      CNEQ_2(2,5) = C4NEQJ_2(2,2,1,3)
      CNEQ_2(2,6) = C4NEQJ_2(2,2,2,3)
      CNEQ_2(3,1) = CNEQ_2(1,3)
      CNEQ_2(3,2) = CNEQ_2(2,3)
      CNEQ_2(3,3) = C4NEQJ_2(3,3,3,3)
      CNEQ_2(3,4) = C4NEQJ_2(3,3,1,2)
      CNEQ_2(3,5) = C4NEQJ_2(3,3,1,3)
      CNEQ_2(3,6) = C4NEQJ_2(3,3,2,3)
      CNEQ_2(4,1) = CNEQ_2(1,4)
      CNEQ_2(4,2) = CNEQ_2(2,4)
      CNEQ_2(4,3) = CNEQ_2(3,4)
      CNEQ_2(4,4) = C4NEQJ_2(1,2,1,2)
      CNEQ_2(4,5) = C4NEQJ_2(1,2,1,3)
      CNEQ_2(4,6) = C4NEQJ_2(1,2,2,3)
      CNEQ_2(5,1) = CNEQ_2(1,5)
      CNEQ_2(5,2) = CNEQ_2(2,5)
      CNEQ_2(5,3) = CNEQ_2(3,5)
      CNEQ_2(5,4) = CNEQ_2(4,5)
      CNEQ_2(5,5) = C4NEQJ_2(1,3,1,3)
      CNEQ_2(5,6) = C4NEQJ_2(1,3,2,3)
      CNEQ_2(6,1) = CNEQ_2(1,6)
      CNEQ_2(6,2) = CNEQ_2(2,6)
      CNEQ_2(6,3) = CNEQ_2(3,6)
      CNEQ_2(6,4) = CNEQ_2(4,6)
      CNEQ_2(6,5) = CNEQ_2(5,6)
      CNEQ_2(6,6) = C4NEQJ_2(2,3,2,3)
! ---------------------------------------------------------------------
!     STEP 6 - CALCULATION OF EQ CAUCHY STRESS & ELASTICITY TENSOR
! ---------------------------------------------------------------------
!     CALCULATE LEFT CAUCHY-GREEN DEFORMATION TENSOR
      BTOT = MATMUL(DFGRD1, TRANSPOSE(DFGRD1))
!
!     SETUP VOIGT NOTATION FOR BTOT
      BTOTV(1) = BTOT(1,1)
      BTOTV(2) = BTOT(2,2)
      BTOTV(3) = BTOT(3,3)
      BTOTV(4) = BTOT(1,2)+BTOT(2,1)
      BTOTV(5) = BTOT(1,3)+BTOT(3,1)
      BTOTV(6) = BTOT(2,3)+BTOT(3,2)
!
!     GET EIGEN VALUES AND EIGEN VECTORS OF BTOT
      ! CALL SPRIND(BTOTV, PVBTOT, PDBTOT, 2, NDI, NSHR)
      DO I = 1,3
        DO J = 1,3
          BTOT_(I,J) = BTOT(I,J)
        END DO
      END DO
      CALL DSYEVJ3(BTOT_, PDBTOT, PVBTOT)
      PDBTOT = TRANSPOSE(PDBTOT)
!
!     CALCULATE EIGEN VALUES OF DEVIATORIC LEFT CAUCHY-GREEN
!     DEFORMATION TENSOR
      DO J=1,3
        PVBBAR(J) = DET**(-TWO/THREE)*PVBTOT(J)
      END DO
!
!     CALCULATING PRINCIPAL VALUES OF KIRCHOFF STRESS
      PVTAUEQ(1) = MU * ((TWO/THREE)*(PVBBAR(1)**(ALPHA/TWO))
     1                -  (ONE/THREE)*(PVBBAR(2)**(ALPHA/TWO))
     2                -  (ONE/THREE)*(PVBBAR(3)**(ALPHA/TWO)))
     3                +   KELAS/TWO*(DET*DET - ONE)
      PVTAUEQ(2) = MU * ((TWO/THREE)*(PVBBAR(2)**(ALPHA/TWO))
     1                -  (ONE/THREE)*(PVBBAR(3)**(ALPHA/TWO))
     2                -  (ONE/THREE)*(PVBBAR(1)**(ALPHA/TWO)))
     3                +   KELAS/TWO*(DET*DET - ONE)
```

```
       PVTAUEQ(3) = MU * ((TWO/THREE)*(PVBBAR(3)**(ALPHA/TWO))
     1                  -  (ONE/THREE)*(PVBBAR(1)**(ALPHA/TWO))
     2                  -  (ONE/THREE)*(PVBBAR(2)**(ALPHA/TWO)))
     3                  +   KELAS/TWO*(DET*DET - ONE)
!
!      CALCULATING EQ CAUCHY STRESS TENSOR
       DO J=1,3
         DO K=1,3
           SIGMAEQ(K,J)=(PVTAUEQ(1)*PDBTOT(1,K)*PDBTOT(1,J)
     1                  +PVTAUEQ(2)*PDBTOT(2,K)*PDBTOT(2,J)
     2                  +PVTAUEQ(3)*PDBTOT(3,K)*PDBTOT(3,J))/DET
         END DO
       END DO
!
!      CALCULATING CAB
       CAB(1,1) = MU * ALPHA
     1           *((FOUR/NINE)*(PVBBAR(1)**(ALPHA/TWO))
     2             +(ONE/NINE)*(PVBBAR(2)**(ALPHA/TWO)
     3                         +PVBBAR(3)**(ALPHA/TWO)))
     4           + KELAS*DET*DET
       CAB(2,2) = MU * ALPHA
     1           *((FOUR/NINE)*(PVBBAR(2)**(ALPHA/TWO))
     2             +(ONE/NINE)*(PVBBAR(3)**(ALPHA/TWO)
     3                         +PVBBAR(1)**(ALPHA/TWO)))
     4           + KELAS*DET*DET
       CAB(3,3) = MU * ALPHA
     1           *((FOUR/NINE)*(PVBBAR(3)**(ALPHA/TWO))
     2             +(ONE/NINE)*(PVBBAR(1)**(ALPHA/TWO)
     3                         +PVBBAR(2)**(ALPHA/TWO)))
     4           + KELAS*DET*DET
       CAB(1,2) = MU * ALPHA
     1           *((-TWO/NINE)*(PVBBAR(1)**(ALPHA/TWO)
     2                         +PVBBAR(2)**(ALPHA/TWO))
     3             +(ONE/NINE)*(PVBBAR(3)**(ALPHA/TWO)))
     4           + KELAS*DET*DET
       CAB(1,3) = MU * ALPHA
     1           *((-TWO/NINE)*(PVBBAR(1)**(ALPHA/TWO)
     2                         +PVBBAR(3)**(ALPHA/TWO))
     3             +(ONE/NINE)*(PVBBAR(2)**(ALPHA/TWO)))
     4           + KELAS*DET*DET
       CAB(2,3) = MU * ALPHA
     1           *((-TWO/NINE)*(PVBBAR(2)**(ALPHA/TWO)
     2                         +PVBBAR(3)**(ALPHA/TWO))
     3             +(ONE/NINE)*(PVBBAR(1)**(ALPHA/TWO)))
     4           + KELAS*DET*DET
       CAB(2,1) = CAB(1,2)
       CAB(3,1) = CAB(1,3)
       CAB(3,2) = CAB(2,3)
!
!      CALCULATING GAB
       DO J=1,3
         DO K=1,3
           IF(J.EQ.K) THEN
             GAB(J,K) = MU * ALPHA * PVBBAR(J)**(ALPHA/TWO)
           ELSE
             IF (ABS(PVBTOT(J)-PVBTOT(K)).LT.EPS) THEN
               GAB(J,K) = MU * ALPHA
     1                  *(TWO/THREE*PVBBAR(J)**(ALPHA/TWO)
     2                   +ONE/THREE*PVBBAR(K)**(ALPHA/TWO))
             ELSE
               GAB(J,K) = (PVTAUEQ(J)*PVBBAR(K) - PVTAUEQ(K)*PVBBAR(J))
     1                  / (PVBBAR(J) - PVBBAR(K))
             END IF
           END IF
         END DO
       END DO
!
!      CALCULATE MATERIAL TENSOR COEFFICIENTS (C4EQMAT)
!      EXPRESSED IN EIGEN BASIS
       C4EQMAT = ZERO
!      1. CALCULATE COEFFICIENTS FOR AABB
       DO J=1,3
         DO K=1,3
```

```
          C4EQMAT(J,J,K,K) = (CAB(J,K) - TWO*PVTAUEQ(J)*IDT2(J,K))
     1                        /DET
        END DO
      END DO
!     2. CALCULATE COEFFICIENTS FOR ABAB / ABBA
      DO J=1,3
        DO K=1,3
          IF (J.EQ.K) THEN
            CYCLE
          END IF
          C4EQMAT(J,K,J,K) = GAB(J,K) / (TWO*DET)
          C4EQMAT(J,K,K,J) = C4EQMAT(J,K,J,K)
        END DO
      END DO
!
!     CALCULATE MATERIAL TENSOR (C4EQ) IN CURRENT CONFIGURATION
!     EXPRESSED IN STANDARD BASIS
      C4EQ = ZERO
      DO J=1,3
        DO K=1,3
          DO L=1,3
            DO M=1,3
              C4EQ(J,K,L,M) =
     1          C4EQMAT(1,1,1,1)*PDBTOT(1,J)*PDBTOT(1,K)
     1                         *PDBTOT(1,L)*PDBTOT(1,M)
     1         +C4EQMAT(1,1,2,2)*PDBTOT(1,J)*PDBTOT(1,K)
     1                         *PDBTOT(2,L)*PDBTOT(2,M)
     1         +C4EQMAT(1,1,3,3)*PDBTOT(1,J)*PDBTOT(1,K)
     1                         *PDBTOT(3,L)*PDBTOT(3,M)
     1         +C4EQMAT(2,2,1,1)*PDBTOT(2,J)*PDBTOT(2,K)
     1                         *PDBTOT(1,L)*PDBTOT(1,M)
     1         +C4EQMAT(2,2,2,2)*PDBTOT(2,J)*PDBTOT(2,K)
     1                         *PDBTOT(2,L)*PDBTOT(2,M)
     1         +C4EQMAT(2,2,3,3)*PDBTOT(2,J)*PDBTOT(2,K)
     1                         *PDBTOT(3,L)*PDBTOT(3,M)
     1         +C4EQMAT(3,3,1,1)*PDBTOT(3,J)*PDBTOT(3,K)
     1                         *PDBTOT(1,L)*PDBTOT(1,M)
     1         +C4EQMAT(3,3,2,2)*PDBTOT(3,J)*PDBTOT(3,K)
     1                         *PDBTOT(2,L)*PDBTOT(2,M)
     1         +C4EQMAT(3,3,3,3)*PDBTOT(3,J)*PDBTOT(3,K)
     1                         *PDBTOT(3,L)*PDBTOT(3,M)
     1         +C4EQMAT(1,2,1,2)*PDBTOT(1,J)*PDBTOT(2,K)
     1                         *PDBTOT(1,L)*PDBTOT(2,M)
     1         +C4EQMAT(1,2,2,1)*PDBTOT(1,J)*PDBTOT(2,K)
     1                         *PDBTOT(2,L)*PDBTOT(1,M)
     1         +C4EQMAT(1,3,1,3)*PDBTOT(1,J)*PDBTOT(3,K)
     1                         *PDBTOT(1,L)*PDBTOT(3,M)
     1         +C4EQMAT(1,3,3,1)*PDBTOT(1,J)*PDBTOT(3,K)
     1                         *PDBTOT(3,L)*PDBTOT(1,M)
     1         +C4EQMAT(2,1,2,1)*PDBTOT(2,J)*PDBTOT(1,K)
     1                         *PDBTOT(2,L)*PDBTOT(1,M)
     1         +C4EQMAT(2,1,1,2)*PDBTOT(2,J)*PDBTOT(1,K)
     1                         *PDBTOT(1,L)*PDBTOT(2,M)
     1         +C4EQMAT(2,3,2,3)*PDBTOT(2,J)*PDBTOT(3,K)
     1                         *PDBTOT(2,L)*PDBTOT(3,M)
     1         +C4EQMAT(2,3,3,2)*PDBTOT(2,J)*PDBTOT(3,K)
     1                         *PDBTOT(3,L)*PDBTOT(2,M)
     1         +C4EQMAT(3,1,3,1)*PDBTOT(3,J)*PDBTOT(1,K)
     1                         *PDBTOT(3,L)*PDBTOT(1,M)
     1         +C4EQMAT(3,1,1,3)*PDBTOT(3,J)*PDBTOT(1,K)
     1                         *PDBTOT(1,L)*PDBTOT(3,M)
     1         +C4EQMAT(3,2,3,2)*PDBTOT(3,J)*PDBTOT(2,K)
     1                         *PDBTOT(3,L)*PDBTOT(2,M)
     1         +C4EQMAT(3,2,2,3)*PDBTOT(3,J)*PDBTOT(2,K)
     1                         *PDBTOT(2,L)*PDBTOT(3,M)
            END DO
          END DO
        END DO
      END DO
!
!     CALCLUATE JAUMANN RATE FORM OF MATERIAL TENSOR
      C4EQJ = ZERO
      DO I=1,3
```

```
        DO J=1,3
          DO K=1,3
            DO L=1,3
              C4EQJ(I,J,K,L) = C4EQ(I,J,K,L)/DET
     1                       +(IDT2(I,K)*SIGMAEQ(J,L)
     1                        +IDT2(I,L)*SIGMAEQ(J,K)
     1                        +IDT2(J,K)*SIGMAEQ(I,L)
     1                        +IDT2(J,L)*SIGMAEQ(I,K))/(TWO)
            END DO
          END DO
        END DO
      END DO
!
!     GENERATE ABAQUS EQ TANGENT STIFFNESS MATRIX (VOIGT NOTATION)
      CEQ = ZERO
      CEQ(1,1) = C4EQJ(1,1,1,1)
      CEQ(1,2) = C4EQJ(1,1,2,2)
      CEQ(1,3) = C4EQJ(1,1,3,3)
      CEQ(1,4) = C4EQJ(1,1,1,2)
      CEQ(1,5) = C4EQJ(1,1,1,3)
      CEQ(1,6) = C4EQJ(1,1,2,3)
      CEQ(2,1) = CEQ(1,2)
      CEQ(2,2) = C4EQJ(2,2,2,2)
      CEQ(2,3) = C4EQJ(2,2,3,3)
      CEQ(2,4) = C4EQJ(2,2,1,2)
      CEQ(2,5) = C4EQJ(2,2,1,3)
      CEQ(2,6) = C4EQJ(2,2,2,3)
      CEQ(3,1) = CEQ(1,3)
      CEQ(3,2) = CEQ(2,3)
      CEQ(3,3) = C4EQJ(3,3,3,3)
      CEQ(3,4) = C4EQJ(3,3,1,2)
      CEQ(3,5) = C4EQJ(3,3,1,3)
      CEQ(3,6) = C4EQJ(3,3,2,3)
      CEQ(4,1) = CEQ(1,4)
      CEQ(4,2) = CEQ(2,4)
      CEQ(4,3) = CEQ(3,4)
      CEQ(4,4) = C4EQJ(1,2,1,2)
      CEQ(4,5) = C4EQJ(1,2,1,3)
      CEQ(4,6) = C4EQJ(1,2,2,3)
      CEQ(5,1) = CEQ(1,5)
      CEQ(5,2) = CEQ(2,5)
      CEQ(5,3) = CEQ(3,5)
      CEQ(5,4) = CEQ(4,5)
      CEQ(5,5) = C4EQJ(1,3,1,3)
      CEQ(5,6) = C4EQJ(1,3,2,3)
      CEQ(6,1) = CEQ(1,6)
      CEQ(6,2) = CEQ(2,6)
      CEQ(6,3) = CEQ(3,6)
      CEQ(6,4) = CEQ(4,6)
      CEQ(6,5) = CEQ(5,6)
      CEQ(6,6) = C4EQJ(2,3,2,3)
!
! ---------------------------------------------------------------------
!     STEP 7 - CALCULATION OF TOTAL CAUCHY STRESS & ELASTICITY TENSOR
! ---------------------------------------------------------------------
!     CALCULATE TOTAL CAUCHY STRESS TENSOR
      DO J=1,3
        DO K=1,3
          STRESSTOT(J,K) = SIGMANEQ(J,K)
     1                    +SIGMANEQ_2(J,K)
     2                    +SIGMAEQ(J,K)
        END DO
      END DO

!     VOIGT NOTATION FOR CAUCHY STRESS
      STRESS(1) = STRESSTOT(1,1)
      STRESS(2) = STRESSTOT(2,2)
      STRESS(3) = STRESSTOT(3,3)
      STRESS(4) = STRESSTOT(1,2)
      STRESS(5) = STRESSTOT(1,3)
      STRESS(6) = STRESSTOT(2,3)
!
!     CALCULATE TOTAL ABAQUS TANGENT STIFFNESS MATRIX
```

```
      DO J=1,6
        DO K=1,6
          DDSDDE(J,K) = CNEQ(J,K) + CNEQ_2(J,K)+ CEQ(J,K)
        END DO
      END DO

      RETURN
      END SUBROUTINE UMAT

! ---------------------------------------------------------------------
!     UTILITY SUBROUTINES
! ---------------------------------------------------------------------
! ---------------------------------------------------------------------
!     M33INV  -  Compute the Inverse of a 3x3 Matrix.
!     SOURCE:  David G. Simpson - NASA Goddard Space Flight Center
!
!     A       - Input  - 3x3 Matrix to be Inverted
!     AINV    - Output - 3x3 Inverse of Matrix A
!     OK_FLAG - .TRUE. If A is non-singular and A is Inverted
! ---------------------------------------------------------------------
      SUBROUTINE M33INV (A, AINV, OK_FLAG)
            IMPLICIT NONE
            DOUBLE PRECISION, DIMENSION(3,3), INTENT(IN)  :: A
            DOUBLE PRECISION, DIMENSION(3,3), INTENT(OUT) :: AINV
            LOGICAL, INTENT(OUT) :: OK_FLAG

            DOUBLE PRECISION, PARAMETER :: EPS = 1.0D-16
            DOUBLE PRECISION :: DET
            DOUBLE PRECISION, DIMENSION(3,3) :: COFACTOR
!
!
            DET = A(1,1) * A(2,2) * A(3,3)
     1          - A(1,1) * A(2,3) * A(3,2)
     2          - A(1,2) * A(2,1) * A(3,3)
     3          + A(1,2) * A(2,3) * A(3,1)
     4          + A(1,3) * A(2,1) * A(3,2)
     5          - A(1,3) * A(2,2) * A(3,1)
!
            IF (ABS(DET) .LE. EPS) THEN
            AINV = 0.0D0
            OK_FLAG = .FALSE.
            RETURN
            END IF
!
            COFACTOR(1,1) = +(A(2,2) * A(3,3) - A(2,3) * A(3,2))
            COFACTOR(1,2) = -(A(2,1) * A(3,3) - A(2,3) * A(3,1))
            COFACTOR(1,3) = +(A(2,1) * A(3,2) - A(2,2) * A(3,1))
            COFACTOR(2,1) = -(A(1,2) * A(3,3) - A(1,3) * A(3,2))
            COFACTOR(2,2) = +(A(1,1) * A(3,3) - A(1,3) * A(3,1))
            COFACTOR(2,3) = -(A(1,1) * A(3,2) - A(1,2) * A(3,1))
            COFACTOR(3,1) = +(A(1,2) * A(2,3) - A(1,3) * A(2,2))
            COFACTOR(3,2) = -(A(1,1) * A(2,3) - A(1,3) * A(2,1))
            COFACTOR(3,3) = +(A(1,1) * A(2,2) - A(1,2) * A(2,1))
!
            AINV = TRANSPOSE(COFACTOR) / DET
!
            OK_FLAG = .TRUE.
            RETURN
!
      END SUBROUTINE M33INV

* ----------------------------------------------------------------------------
      SUBROUTINE DSYEVJ3(A, Q, W)
* ----------------------------------------------------------------------------
* Calculates the eigenvalues and normalized eigenvectors of a symmetric 3x3
* matrix A using the Jacobi algorithm.
* The upper triangular part of A is destroyed during the calculation,
* the diagonal elements are read but not destroyed, and the lower
* triangular elements are not referenced at all.
* ----------------------------------------------------------------------------
* Parameters:
*   A: The symmetric input matrix
*   Q: Storage buffer for eigenvectors
*    W: Storage buffer for eigenvalues
* ----------------------------------------------------------------------------
*      .. Arguments ..
      DOUBLE PRECISION A(3,3)
      DOUBLE PRECISION Q(3,3)
      DOUBLE PRECISION W(3)

*      .. Parameters ..
      INTEGER          N
      PARAMETER        ( N = 3 )

*      .. Local Variables ..
      DOUBLE PRECISION SD, SO
      DOUBLE PRECISION S, C, T
      DOUBLE PRECISION G, H, Z, THETA
      DOUBLE PRECISION THRESH
      INTEGER          I, X, Y, R

*      Initialize Q to the identitity matrix
*      --- This loop can be omitted if only the eigenvalues are desired ---
      DO 10 X = 1, N
        Q(X,X) = 1.0D0
        DO 11, Y = 1, X-1
          Q(X, Y) = 0.0D0
          Q(Y, X) = 0.0D0
   11    CONTINUE
   10  CONTINUE

*      Initialize W to diag(A)
      DO 20 X = 1, N
        W(X) = A(X, X)
   20  CONTINUE

*      Calculate SQR(tr(A))
      SD = 0.0D0
      DO 30 X = 1, N
        SD = SD + ABS(W(X))
   30  CONTINUE
      SD = SD**2

*      Main iteration loop
      DO 40 I = 1, 50
*        Test for convergence
        SO = 0.0D0
        DO 50 X = 1, N
          DO 51 Y = X+1, N
            SO = SO + ABS(A(X, Y))
   51      CONTINUE
   50    CONTINUE
        IF (SO .EQ. 0.0D0) THEN
          RETURN
        END IF

        IF (I .LT. 4) THEN
          THRESH = 0.2D0 * SO / N**2
        ELSE
          THRESH = 0.0D0
        END IF

*        Do sweep
        DO 60 X = 1, N
          DO 61 Y = X+1, N
            G = 100.0D0 * ( ABS(A(X, Y)) )
            IF ( I .GT. 4 .AND. ABS(W(X)) + G .EQ. ABS(W(X))
     $                     .AND. ABS(W(Y)) + G .EQ. ABS(W(Y)) ) THEN
              A(X, Y) = 0.0D0
            ELSE IF (ABS(A(X, Y)) .GT. THRESH) THEN
*              Calculate Jacobi transformation
              H = W(Y) - W(X)
              IF ( ABS(H) + G .EQ. ABS(H) ) THEN
                T = A(X, Y) / H
              ELSE
                THETA = 0.5D0 * H / A(X, Y)
                IF (THETA .LT. 0.0D0) THEN
                  T = -1.0D0 / (SQRT(1.0D0 + THETA**2) - THETA)
                ELSE
                  T = 1.0D0 / (SQRT(1.0D0 + THETA**2) + THETA)
                END IF
              END IF

              C = 1.0D0 / SQRT( 1.0D0 + T**2 )
              S = T * C
              Z = T * A(X, Y)

*              Apply Jacobi transformation
              A(X, Y) = 0.0D0
              W(X)    = W(X) - Z
              W(Y)    = W(Y) + Z
              DO 70 R = 1, X-1
                T       = A(R, X)
                A(R, X) = C * T - S * A(R, Y)
                A(R, Y) = S * T + C * A(R, Y)
   70          CONTINUE
              DO 80, R = X+1, Y-1
                T       = A(X, R)
                A(X, R) = C * T - S * A(R, Y)
                A(R, Y) = S * T + C * A(R, Y)
   80          CONTINUE
              DO 90, R = Y+1, N
                T       = A(X, R)
                A(X, R) = C * T - S * A(Y, R)
                A(Y, R) = S * T + C * A(Y, R)
   90          CONTINUE

*              Update eigenvectors
*              --- This loop can be omitted if only the eigenvalues are desired
              DO 100, R = 1, N
                T       = Q(R, X)
                Q(R, X) = C * T - S * Q(R, Y)
                Q(R, Y) = S * T + C * Q(R, Y)
 100          CONTINUE
            END IF
   61      CONTINUE
   60    CONTINUE
   40  CONTINUE

      PRINT *, "DSYEVJ3: No convergence."

      END SUBROUTINE
* End of subroutine DSYEVJ3
```