Nonlinear anisotropic hyperelastic arterial wall material models into ANSYS

User Manual & Theory - v0.1

Nicolas Mesnier*

June 15, 2011

Abstract

This document aims to serve as *user manual* to use and implement anisotropic hyperelastic material models to describe the arterial wall behaviour. Embedded into the classical geometric formulation of continuum mechanics, we provide an explicit derivation of three constitutive models. Their implementation into a user material routine written in modern Fortran is also presented and the Fortran code listing for the user material routine `usermat3d()` is given.

*Permanent e-mail: nicolas.mesnier@gmail.com

Contents

1 Introduction **3**

2 Installation **3**
- 2.1 Installation . . . 3
- 2.2 Recompilation . . . 4
  - 2.2.1 Installation des compilateurs . . . 4
  - 2.2.2 La re-compilation . . . 5
  - 2.2.3 Le premier bug . . . 5
  - 2.2.4 Le cadeau de la v12 . . . 6

3 User manual **6**
- 3.1 Linking `usermat()` . . . 6
- 3.2 Elements restrictions . . . 7
- 3.3 ADPL specifications . . . 7
  - 3.3.1 Model 1 - Holzapfel et al. (2000) . . . 7
  - 3.3.2 Model 2 - Baek et al. (2007) . . . 8
  - 3.3.3 Model 3 - Holzapfel et al. (2005) . . . 8

4 Theory and implementation **9**
- 4.1 Theoretical background . . . 9
  - 4.1.1 Variational formulation . . . 9
  - 4.1.2 Corotational kinematics . . . 10
  - 4.1.3 Corotational tangent stiffness . . . 11
- 4.2 Explicit tangent stiffness . . . 11
  - 4.2.1 Holzapfel (2000) sedf . . . 11
  - 4.2.2 Holzapfel (2005) sedf . . . 12
- 4.3 Implementation . . . 14
  - 4.3.1 Element orientation and material anisotropy 14
  - 4.3.2 Voigt notation . . . 14

A Ansys `usermat3d()` subroutine **16**

1 Introduction

Since the arteries may be regarded as a fibre-reinforced composite, we are interested in the numerical implementation of specific anisotropic hyperelastic behaviour laws into ANSYS. In order to describe the anisotropic elastic stress response of arteries, we decided to use the strain energy density function proposed by Holzapfel et al. [2, 3] and Baek et al. [1]. Consequently, the following anisotropic hyperelastic material models are currently implemented:

1. Anisotropic two conjugated fibres family of Holzapfel et al. [2],
2. Anisotropic four fibres family of Baek et al. [1], in which two fibres family are conjugated,
3. Anisotropic two conjugated fibres family of Holzapfel et al. [3].

For that purpose, we first present the way to use these material models with ANSYS. In a second step, we recall some specificity of the finite element method and particularly the way to define a user material (in ANSYS). A special attention is provide on the material tangent, naturally given in the spatial configuration and which need to be recast in the corotated frame. For such purpose, we recall the algorithm for determination of the corotated frame rotation via the polar decomposition of the deformation gradient, as well as the conversion of the stress, material tangent and change of basis operations to Voigt notation. After while we derived the material models and finally we give the Fortran code listing for the user material routine `usermat3d()`.

2 Installation

Dans cette partie, nous avons regroupé les informations permettant d'utiliser ANSYS en version personnalisée sur une distribution linux 32 bits de base debian, comme ubuntu. Les petits détails pas toujours évidents à régler pour y parvenir sont ici mentionnés.

2.1 Installation

Pour utiliser ANSYS, il faut installer la version 32 bits linux. Immédiatement après, il est nécessaire d'installer les librairies `mesa` et `libmotif3`

```
sudo apt-get install libglu1-mesa
sudo apt-get install libmotif3
```

et de créer quelques liens symboliques dans le répertoire `/usr/lib` du genre

```
sudo ln -s /usr/lib/libGLU.so.1.*.* /usr/lib/libGLU.so
sudo ln -s /usr/lib/libXm.so.3.*.* /usr/lib/libXm.so
sudo ln -s /usr/lib/libXi.so.6.*.* /usr/lib/libXi.so
sudo ln -s /usr/lib/libXext.so.6.*.* /usr/lib/libXext.so
sudo ln -s /usr/lib/libXmu.so.*.* /usr/lib/libXmu.so
sudo ln -s /usr/lib/libXp.so.6.*.* /usr/lib/libXp.so
sudo ln -s /usr/lib/libXt.so.6.*.* /usr/lib/libXt.so
sudo ln -s /usr/lib/libX11.so.6.*.* /usr/lib/libX11.so
```

associés aux paquets dépendants de `motif`, notamment `libxi6, libxext6, libxmu6, libxp6, libxt6, libx11-6`. Ensuite, il est nécessaire de faire pointer la librairie openGL vers celle de votre carte graphique. En ce qui concerne mes machines, c'est du nvidia et j'ai créé le lien

```
sudo ln -s /usr/lib/nvidia-current/libGL.so /usr/lib/libGL.so
```

Avec tout ça, vous lancer ansys en mode graphique avec la commande

```
/usr/local/ansys_inc/vXX0/ansys/bin/launcherXX0
```

où vous prendrez soin de remplacer les "XX" par votre version d'ANSYS.

2.2 Recompilation

Pour pouvoir recompiler ANSYS, par exemple pour utiliser son propre modèle de comportement, il est nécessaire d'avoir installé les compilateurs intel cpp V9.1 et intel fortran V9.1 (tel que certifié par Ansys).

2.2.1 Installation des compilateurs

J'ai téléchargé sur le site intel
http://software.intel.com/en-us/articles/intel-compilers/
les versions:

- intel cpp 32 bits version 9.1.047,
- intel fortran 32 bits version 9.1.043.

Une fois les archives disponibles, avant de procéder à l'installation des compilateurs, il est nécessaire d'installer les pacquets

```
sudo apt-get install alien g++-multilib libstdc++5 build-essential
```

et de faire quelques modifications banales du genre remplacer le lien de `sh` vers `dash`, visible par

```
moi@mon-pc:~$ls -l /bin/sh
lrwxrwxrwx 1 root root 9 date heure /bin/sh -> /bin/dash
```

vers `bash` selon

```
sudo rm /bin/sh
sudo ln -s /bin/bash /bin/sh
```

Après quoi, il est nécessaire de décompresser l'archive de chacun des compilateurs et de créer un `*.deb` adapté à l'installation sur base debian à partir du `*.rpm` (RedHat) fourni par intel. Pour le compilateur fortran, il faut exécuter la séquence

```
tar xvzf l_fc_c_9.1.043_ia32.tar.gz
cd l_fc_c_9.1.043_ia32/data
sudo alien -cv intel-ifort91043-9.1.043-1.i386.rpm
sudo dpkg -i intel-ifort91043-9.1.043-1_i386.deb
```

et de façon équivalente pour le compilateur cpp

```
tar xvzf l_cc_c_9.1.047_ia32.tar.gz
cd l_cc_c_9.1.047_ia32/data
sudo alien -cv intel-icc91047-9.1.047-1.i386.rpm
sudo dpkg -i ntel-icc91047-9.1.047-1.i386.deb
```

Bien que les deux compilateurs soient maintenant installés, les fichiers se réfèrent à `<INSTALLDIR>` et non à `/opt/intel/...` comme il faudrait. Il est donc nécessaire d'aller remplacer toutes les occurrences pour les deux compilateurs. Comme nous ne sommes pas (encore) fous, on lance une simple commande `sed` en root:

```
sudo sed -i 's/&lt;INSTALLDIR&gt;/\/opt\/intel\/fc\/9.1.043/g' \
                          /opt/intel/fc/9.1.043/bin/ifortvars.sh
sudo sed -i 's/&lt;INSTALLDIR&gt;/\/opt\/intel\/cc\/9.1.047/g'\
                          /opt/intel/cc/9.1.047/bin/iccvars.sh
```

Maintenant, il ne reste plus qu'à copier les fichiers de licence gracieusement offerts par intel pour toute utilisation non-commerciale

```
cp quelque_part/NCOM_L_CMP_FOR*.lic /opt/intel/fc/9.1.043/licenses/.
cp quelque_part/NCOM_L_CMP_CPP*.lic /opt/intel/cc/9.1.047/licenses/.
```

et d'ajouter deux lignes au fichier `~/.bashrc` pour être certain que le shell hérite bien des variables des compilateurs intel:

```
cd ~
echo "source /opt/intel/fc/9.1.043/bin/ifortvars.sh" 1>>.bashrc
echo "source /opt/intel/cc/9.1.047/bin/iccvars.sh" 1>>.bashrc
```

2.2.2 La re-compilation

Les fichiers sources qu'ANSYS vous autorise à recompiler sont dans le répertoire `/usr/local/ansys_inc/vXX0/ansys/customize`. Je vous encourage vivement à copier ce répertoire et le mettre dans un autre répertoire, par exemple celui de vos fichiers sources de simulation. Pour ma part, j'en ai profité pour le renommer `ansys-custom`. Les *User Programmable Features* d'ANSYS résident dans le sous-répertoire `user`. Pour mes travaux, je me suis amusé à éditer le code fortran `ansys-custom/user/usermat3d.F` associé à la définition de modèles de comportement mécanique. Ainsi, une fois que vous avez codé ce que vous souhaitiez, vous recompilez ANSYS avec

```
./ansys-custom/user/ANSCUSTOM
```

et tapez deux fois `y` pour créer un exécutable distribuable qui portera le doux nom de `ansys-custom/user/ansyscust.eXX0`, "XX" étant toujours associé à votre version.

2.2.3 Le premier bug

Un des premiers éléments auxquels vous êtes confronté après avoir installé les compilateurs et essayé votre première compilation est le message d'erreur:

```
/usr/ansys_inc/vxx0/ansys/lib/linia32/libansys.so:
       undefined reference to `glPolygonOffsetEXT'
```

où vous prendrez soin de remplacer les "xx" par la version d'ANSYS. Le palliatif qui a fonctionné est de créer un *wrapper*[1] vers la bonne fonction sous la forme

```
#include <GL/gl.h>

void glPolygonOffsetEXT(float factor, float bias) {
glPolygonOffset(factor, bias);
}
```

que vous enregistrez par exemple sous le nom `wrapper.c` et que vous compilez ensuite sous forme de librairie:

```
gcc -fPIC -shared -o wrapper.so wrapper.c
```

Partant de ce point, il est nécessaire d'inclure l'appel à cette librairie lors de la compilation. En créant un répertoire `doc` avec le source et la librairie `wrapper.so` dans votre répertoire `ansys-custom`, il vous faut éditer le fichier `user/ANSCUSTOM` et ajouter la ligne

```
                -L ../doc -lwrapper \
```

au niveau de l'appel aux librairies graphiques `lopnsx11=`... dans la commande relative aux systèmes linux 32 bits.

2.2.4 Le cadeau de la v12

Avec la v12, lorsque vous essayez de compiler vos anciens codes, vous êtes confronté à de nouveaux petits problèmes. Pour ma part, ils ont été résolu en créant simplement des liens symboliques associés aux librairies MPI fournies par ANSYS; explicitement

```
sudo ln -s /usr/local/ansys_inc/v120/ansys/hpmpi/\
      linia32/lib/linux_ia32/libmpio.so.1 /usr/lib/.
sudo ln -s /usr/local/ansys_inc/v120/ansys/hpmpi/\
      linia32/lib/linux_ia32/libmpi.so.1 /usr/lib/.
```

Avec ces liens, tout fonctionne (enfin) correctement.

3 User manual

3.1 Linking `usermat()`

Pour utiliser votre ANSYS recompilé, vous disposez de deux moyens. Le premier est le mode graphique que vous lancez avec la commande

```
/usr/local/ansys\_inc/vXX0/ansys/bin/launcherXX0
```

et pour lequel vous prenez soin dans le second onglet de préciser l'exécutable que vous souhaitez utiliser avant de lancer la commande `run`. Sur un plan pratique, vous cliquez sur `browse` et allez chercher le fichier exécutable ANSYS

```
quelque_part/ansys-custom/user/ansyscust.eXX0
```

[1] Les détails sont disponibles sur http://www.nvnews.net/vbulletin/archive/index.php/t-68080.html.

qui vous intéresse. Si vide, ce sera ANSYS par défaut.

Le second moyen, plus pratique pour les gros calculs, est d'appeler ANSYS en batch. Dans ce cas, il suffit d'utiliser la séquence de commandes

```
ansys=/usr/local/ansys_inc/vXX0/ansys/bin/ansysXX0
ansyscust=quelque_part/ansys-custom/user/ansyscust.eXX0
file=your-job-file.inp
"$ansys"  -p AA_T_ME -j "job-file" -s read -l en-us
         -b <"$file" -custom "$ansyscust" | tee -i 'job-file.out'
```

ici associée à l'exécution du job ANSYS appelé `job-file` et dont les instructions sont données dans le fichier `your-job-file.inp`.

3.2 Elements restrictions

The user material routine, `usermat()`, is an ANSYS user-programmable feature for use with the 18x family elements which allow user to write their own material constitutive equations within. In our specification of the code, it can be use with `plane182`, `plane183`, `solid185` and `solid186`. This subroutine is called at all material integration points of these elements during the solution phase. The input parameters for the `usermat` subroutine is defined by command `tb,user`.

3.3 ADPL specifications

Recall that ADPL is nothing but *ANSYS Parametric Design Language*. This is the classical code used to define a simulation file. Then, when you want to use one of the three material behaviour law of the arterial wall, you need the following specifications:

3.3.1 Model 1 - Holzapfel et al. (2000)

The strain energy density proposed by Holzapfel et al. [2] is given by

$$\psi = \frac{\mu}{2}\left(\bar{I}_1 - 3\right) + \frac{\kappa}{2}\left(J-1\right)^2 + \sum_{\alpha=1}^{2} \frac{k_1}{2\,k_2}\left(\exp\left[k_2\left(\bar{I}_4^{\alpha} - 1\right)^2\right] - 1\right) \ . \qquad (1)$$

To define such material model with the ADPL, you need to furnish to the `usermat()` routine the following definitions:

```
mat-nb=1           ! material number
tb,user,mat-nb,1,6
tbdata,1,1         ! Holzapfel et al. (2000)
tbdata,2,15.02     ! mu
tbdata,3,1e5       ! k (bulk modulus)
tbdata,4,38.57     ! k1
tbdata,5,85.03     ! k2
tbdata,6,67        ! beta, deg
tb,state,mat-nb,,4 ! state var
```

3.3.2 Model 2 - Baek et al. (2007)

The strain energy density proposed by Baek et al. [1] is given by

$$\psi = \frac{\mu}{2}\left(\bar{I}_1 - 3\right) + \frac{\kappa}{2}\left(J-1\right)^2 + \sum_{\alpha=1}^{2}\frac{k_1}{2\,k_2}\left(\exp\left[k_2\left(\bar{I}_4^{\alpha}-1\right)^2\right]-1\right) + \frac{k_1^{\theta}}{2\,k_2^{\theta}}\left(\exp\left[k_2^{\theta}\left(\bar{I}_4^{\theta}-1\right)^2\right]-1\right) + \frac{k_1^{z}}{2\,k_2^{z}}\left(\exp\left[k_2^{z}\left(\bar{I}_4^{z}-1\right)^2\right]-1\right) \quad . \quad (2)$$

To define such material model with the ADPL, you need to furnish to the `usermat()` routine the following definitions:

```
mat-nb=1           ! material number
tb,user,mat-nb,1,10
tbdata,1,2         ! Baek et al. (2007)
tbdata,2,75.53     ! mu
tbdata,3,1e5       ! k (bulk modulus)
tbdata,4,6.25      ! k1\theta
tbdata,5,0.1370    ! k2\theta
tbdata,6,12.71     ! k1z
tbdata,7,0.0290    ! k2z
tbdata,8,0.016     ! k1\alpha
tbdata,9,1.4390    ! k2\alpha
tbdata,10,58.6000  ! beta, deg
tb,state,mat-nb,,4 ! state var
```

3.3.3 Model 3 - Holzapfel et al. (2005)

The strain energy density proposed by Holzapfel et al. [3] is given by

$$\psi = \frac{\mu}{2}\left(\bar{I}_1 - 3\right) + \frac{\kappa}{2}\left(J-1\right)^2 + \frac{k_1}{k_2}\left[\exp\left(k_2\left[\left(1-\rho\right)\left(\bar{I}_1-3\right)^2 + \rho\left(\bar{I}_4-1\right)^2\right]\right)-1\right] \quad . \quad (3)$$

To define such material model with the ADPL, you need to furnish to the `usermat()` routine the following definitions:

```
mat-nb=1           ! material number
tb,user,mat-nb,1,7
tbdata,1,3         ! Holzapfel et al. (2005)
tbdata,2,15.02     ! mu
tbdata,3,1e5       ! k (bulk modulus)
tbdata,4,38.57     ! k1
tbdata,5,85.03     ! k2
tbdata,6,0.55      ! rho
tbdata,7,67        ! beta, deg
tb,state,mat-nb,,4 ! state var
```

4 Theory and implementation

In this section we first recall the basis to express spatial quantities into the corotational frame. After while, the basis of the finite element method are recalled so as to have the necessary theoretical background to understand the implementation of a user material into the finite element commercial software ANSYS by using the user subroutine `usermat3d()`.

4.1 Theoretical background

We are interested in an anisotropic strain energy density function (sedf) of the form

$$\psi : (\mathbf{g}, \mathbf{b}, \mathbf{a}_\alpha) \to \mathbb{R} \tag{4}$$

per unit volume of the material manifold $\mathscr{B}$, where $\mathbf{g}$ is the spatial (euclidean) metric and $\mathbf{b}$ is introduced as relevant deformation measure. It is defined as the inverse of the convected material metric $\mathbf{G}$, explicitly

$$\mathbf{b} = (\varphi_* \mathbf{G})^{-1} = F^i{}_A \, F^j{}_B G^{AB} \; \partial_i \otimes \partial_j \; , \tag{5}$$

and customary denoted the left Cauchy-Green strain tensor. To model anisotropy due to fibres behaviour, we introduce the family of symmetric second order structural tensor

$$\mathbf{a}_\alpha = \varphi_* \left(\boldsymbol{f}_\alpha \otimes \boldsymbol{f}_\alpha \right) = F^i_A \, f^A_\alpha \, f^B_\alpha \, F^j_B \; \partial_i \otimes \partial_j \; , \quad \alpha = 1, .., n \tag{6}$$

in which the unit vector $\boldsymbol{f}_\alpha$ describe the α-fibre direction on the material manifold $\mathscr{B}$. Following the theory of invariants and classical decomposition methods, we may rewrite the generic sedf in the form

$$\psi = \psi_{\text{vol}} \left(J \right) + \psi_{\text{mat}} \left(\bar{I}_1 \right) + \sum_{\alpha \in \mathcal{B}} \psi^\alpha_{\text{fib}} \left(\bar{I}^\alpha_4 \right) \tag{7}$$

with the definition of invariants

$$J = \det \mathbf{F} \; , \quad \bar{I}_1 = J^{-\frac{2}{3}} \; g_{ij} \; b^{ij} \; , \quad \bar{I}^\alpha_4 = J^{-\frac{2}{3}} \; g_{ij} \, a^{ij}_\alpha . \tag{8}$$

where we used the abbreviation $\bar{\bullet} = J^{-\frac{2}{3}} \bullet$.

4.1.1 Variational formulation

In quasi-static finite elasticity, the classical (covariant) variations $\delta_\xi \psi \left[\partial \varphi \right] = d_\varepsilon \psi \left[\partial \left(\phi^\varepsilon \circ \varphi \right) \right] |_{\varepsilon = 0}$, where ϕ^ε is the flow generated by the vector field $\xi = \xi^i \partial_i$, may refer to the Lie-derivative

$$\mathfrak{L}_\xi \psi = \frac{\partial \psi}{\partial \mathbf{g}} : \mathfrak{L}_\xi \mathbf{g} = \boldsymbol{\tau} : \mathbf{d} \quad \text{with } \boldsymbol{\tau} = 2 \, \partial_{\mathbf{g}} \, \psi \tag{9}$$

where $\mathbf{d} = \text{sym} \left(\mathbf{l} \right)$ and with $\mathbf{l} = \nabla_i \xi_j dx^i \otimes dx^j$. Thus, the Lie derivative of the Kirshhoff stress may be expressed as

$$\mathfrak{L}_\xi \boldsymbol{\tau} = \frac{\partial \boldsymbol{\tau}}{\partial \mathbf{g}} : \mathfrak{L}_\xi \mathbf{g} = \mathbf{c} : \mathbf{d}. \tag{10}$$

In the above relation, $\mathbf{c}$ is the tangent stiffness, defined by

$$(\mathbf{c})^{ijkl} = 2\frac{\partial \tau^{ij}}{\partial g_{kl}}. \tag{11}$$

On an other hand, let us recall that the Lie derivative of the Kirshhoff stress classically reads

$$\mathfrak{L}_\xi \boldsymbol{\tau} = d_t \boldsymbol{\tau} - \mathbf{l}\cdot\boldsymbol{\tau} - \boldsymbol{\tau}\cdot\mathbf{l}^t. \tag{12}$$

with $\mathbf{l} = \mathbf{d} + \boldsymbol{\omega}$, $\boldsymbol{\omega} = \text{skew}\,(\mathbf{l})$.

4.1.2 Corotational kinematics

Let us introduce the transformation $\boldsymbol{\varphi}$ that maps material points $\boldsymbol{X}$ in the material configuration $\mathcal{B}_0$ onto a position $\mathbf{x} = \boldsymbol{\varphi}\,(\boldsymbol{X}, t)$ in the spatial configuration $\mathcal{B}_t$. The corresponding deformation gradient $\mathbf{F} = \partial_{\boldsymbol{X}}\boldsymbol{\varphi}\,(\boldsymbol{X}, t)$,

$$\mathbf{F} = \partial_A \varphi^i \;\partial_i \otimes dX^A, \tag{13}$$

with Jacobian $J = \det \mathbf{F} > 0$, denotes the tangent map from the material tangent space $T\mathcal{B}_0$ to the spatial tangent space $T\mathcal{B}_t$. As each $GL\,(3,\mathbb{R})$-tensor, it may be recasted into a unique polar decomposition

$$\mathbf{F} = R^i_a \; U^a_A \; \partial_i \otimes dX^A \tag{14}$$

where $\mathbf{U} = U^a_A \; \partial_a \otimes dX^A$ is the stretch tensor. Thus, given a deformation gradient, the frame rotation $\mathbf{R}$ is determined from the closed-form solution of the inverse stretch,

$$\mathbf{U}^{-1} = \frac{1}{III_U}\left(\mathbf{C} - I_U \;\mathbf{U} + II_U \;\mathbf{I}\right) \tag{15}$$

where I_U, II_U and III_U are the principal invariants of $\mathbf{U}$. Since the eigenvalues of $\mathbf{U}$ are the square root of the eigenvalues of $\mathbf{C} = \varphi^*\mathbf{g}$ given by the roots of the characteristic polynomial

$$P(\lambda) = -\lambda^3 + \lambda^2\, I_1 - \lambda\; I_2 + I_3 \tag{16}$$

where the principal invariants are

$$I_1 = \text{tr}\,(\mathbf{C}) \;\;,\quad I_2 = \frac{1}{2}\left(I_1^2 - \text{tr}\left(\mathbf{C}^2\right)\right) \;\;,\quad I_3 = \det(\mathbf{C})\,. \tag{17}$$

Since $\{C_{AB}\}$ is a symmetric, positive-definite, rank 3 matrix, Eq.(16) has 3 positive real roots given by

$$\lambda_i = \frac{I_1}{3} + \frac{2}{3}\sqrt{I_1^2 - 2I_2}\;\cos\left(\frac{\beta + 2\pi(i-1)}{3}\right)\;, \tag{18}$$

$$\text{with}\quad \beta = \arccos\left(\frac{2I_1^3 - 9\,I_1\,I_2 + 27\,I_3}{2\sqrt{\left(I_1^2 - 2I_2\right)^3}}\right). \tag{19}$$

This formulation is implemented by means of the `polarRU()` subrountine. For further details, the reader may refers to Simo and Hughes [4, box 7.1 page 743].

4.1.3 Corotational tangent stiffness

Now introducing the corotated Kirhhoff stress $\hat{\boldsymbol{\tau}}$, that is explicitly the pull-back of the Kirshhoff stress into the corotated frame

$$\hat{\boldsymbol{\tau}} = \mathbf{R}^* (\boldsymbol{\tau}) = R^{-a}_{\ i}\, \tau^{ij}\, R^{-b}_{\ j}\, \partial_a \otimes \partial_b \tag{20}$$

$\mathbf{R} = \mathbf{F} \cdot \mathbf{U}^{-1}$ being the rotation of the corotated frame, it is straightforward to see that the push-forward of the Lie derivative of the corotated Kirhhoff stress is equivalent to the Lie derivative of the Kirhhoff stress, augmented by convective terms. That is explicitly in components,

$$\mathbf{R}_* \left(\mathfrak{L}_\xi \hat{\boldsymbol{\tau}}\right) = \mathfrak{L}_\xi \boldsymbol{\tau} + \mathbf{d} \cdot \boldsymbol{\tau} + \boldsymbol{\tau} \cdot \mathbf{d} = J\mathcal{C} : \mathbf{d}. \tag{21}$$

In the above relation, $\mathcal{C}$ is the Jaumann tangent stiffness, defined with minor symmetries by

$$(\mathcal{C})^{ijkl} = \frac{1}{J}(\mathbf{c})^{ijkl} + \frac{1}{2J}\left(g^{ik}\tau^{jl} + \tau^{ik}g^{jl} + g^{il}\tau^{jk} + \tau^{il}g^{jk}\right). \tag{22}$$

Finally, one obtains the corotated fourth order tangent operator $\hat{\mathcal{C}}$ needed to be calculated at each iteration and at each integration point by simple pull-back in the corotated frame of all its components; explicitly

$$(\hat{\mathcal{C}})^{abcd} = R^{-a}_{\ i}\ R^{-b}_{\ j}\ (\mathcal{C})^{ijkl}\ R^{-c}_{\ k}\ R^{-d}_{\ l}\ . \tag{23}$$

Then, since we customary used isoparametric finite elements, we must rewrite all these equation into full Cartesian coordinates, i.e. $G_{AB} = \delta_{AB}$ and $g_{ij} = \delta_{ij}$.

4.2 Explicit tangent stiffness

In this section we give the explicit tangent stiffness related to the chosen material models. Since Baek et al. [1] is a simple four fibres extension of Holzapfel et al. [2] model, we just concentrated our efforts on Holzapfel et al. [2] and Holzapfel et al. [3] models.

In what follow, we use the classical derivations of the invariants in terms of the spatial metric

$$\frac{\partial J}{g_{ij}} = \frac{J}{2}\, g^{ij}\ , \quad \frac{\partial \bar{I}_1}{g_{ij}} = \bar{b}^{ij} - \frac{\bar{I}_1}{3} g^{ij}\ , \quad \frac{\partial \bar{I}^\alpha_4}{g_{ij}} = \bar{a}^{ij}_\alpha - \frac{\bar{I}^\alpha_4}{3} g^{ij}\ , \tag{24}$$

and the useful definition of (minus) the fourth rank identity tensor, computed from the well-known relationship

$$\frac{\partial g^{ij}}{g_{kl}} = -\frac{1}{2}\ \left(g^{ik}\, g^{jl} + g^{il}\, g^{ik}\right)\ . \tag{25}$$

4.2.1 Holzapfel (2000) sedf

The strain energy density proposed by Holzapfel et al. [2] is given by

$$\psi = \psi_{\text{vol}}\left(J\right) + \psi_{\text{mat}}\left(\bar{I}_1\right) + \sum_{\alpha \in \mathcal{B}} \psi^\alpha_{\text{fib}}\left(\bar{I}^\alpha_4\right) \tag{26}$$

with the three parts having explicit forms

$$\psi_{\text{vol}} = \frac{\kappa}{2}\left(J-1\right)^2 \ , \tag{27a}$$

$$\psi_{\text{mat}} = \frac{\mu}{2}\left(\bar{I}_1 - 3\right) \ , \tag{27b}$$

$$\psi^{\alpha}_{\text{fib}} = \frac{k^{\alpha}_1}{2\,k^{\alpha}_2}\left(\exp\left[k^{\alpha}_2\left(\bar{I}^{\alpha}_4 - 1\right)^2\right] - 1\right) \ . \tag{27c}$$

Thus, the elastic stress response is governed by the three contributions

$$\tau^{ij}_{\text{vol}} = \kappa\ J\left(J-1\right)\ g^{ij} \ , \tag{28a}$$

$$\tau^{ij}_{\text{mat}} = \mu\ \left(\text{dev}\,\bar{\mathbf{b}}\right)^{ij} \ , \tag{28b}$$

$$\tau^{ij}_{\text{fib}} = \sum_{\alpha\in\mathcal{B}} 2\left(\psi^{\alpha}_{\text{fib}}\right)_{,\alpha}\ \left(\text{dev}\,\bar{\mathbf{a}}\right)^{ij} \ . \tag{28c}$$

where

$$\left(\text{dev}\,\bar{\mathbf{b}}\right)^{ij} = \bar{b}^{ij} - \frac{\bar{I}_1}{3}\,g^{ij} \ , \tag{29}$$

$$\left(\text{dev}\,\bar{\mathbf{a}}\right)^{ij} = \bar{a}^{ij} - \frac{\bar{I}_4}{3}\,g^{ij} \ , \tag{30}$$

$$\left(\psi^{\alpha}_{\text{fib}}\right)_{,\alpha} = k^{\alpha}_1\left(\bar{I}^{\alpha}_4 - 1\right)\ \exp\left[k^{\alpha}_2\left(\bar{I}^{\alpha}_4 - 1\right)^2\right] \ . \tag{31}$$

Thus, using the decoupled representation of the Kirshhoff stress tensor, its derivatives straightforward leads to the relations for the volumic and isotropic deviatoric parts:

$$\left(\mathbf{c}_{\text{vol}}\right)^{ijkl} = \kappa\ J\left[\left(2J-1\right)g^{ij}\ g^{kl} - \left(J-1\right)\ \left(g^{ik}\,g^{jl} + g^{il}\,g^{jk}\right)\right] \ , \tag{32a}$$

$$\left(\mathbf{c}_{\text{mat}}\right)^{ijkl} = \frac{2}{3}\mu\ \left[\frac{\bar{I}_1}{2}\left(g^{ik}\,g^{jl} + g^{il}\,g^{jk}\right) + \frac{\bar{I}_1}{3}g^{ij}\ g^{kl} - \left(g^{ij}\,\bar{b}^{kl} + \bar{b}^{ij}\,g^{kl}\right)\right] . \tag{32b}$$

Finally, for the anisotropic deviatoric part, we used both derivation on J and $\mathbf{a}$ and obtained

$$\begin{aligned}\left(\mathbf{c}_{\text{fib}}\right)^{ijkl} = \sum_{\alpha\in\mathcal{B}} \Big\{ & 4\left(\psi^{\alpha}_{\text{fib}}\right)_{,\alpha\alpha}\ \left(\text{dev}\,\bar{\mathbf{a}}\right)^{ij}\ \left(\text{dev}\,\bar{\mathbf{a}}\right)^{kl} \\ & + \frac{4}{3}\left(\psi^{\alpha}_{\text{fib}}\right)_{,\alpha}\ \left[\frac{\bar{I}^{\alpha}_4}{2}\left(g^{ik}\,g^{jl} + g^{il}\,g^{jk}\right) + \frac{\bar{I}^{\alpha}_4}{3}g^{ij}\ g^{kl} - \left(g^{ij}\,\bar{a}^{kl}_{\alpha} + \bar{a}^{ij}_{\alpha}\,g^{kl}\right)\right]\Big\}\end{aligned} \tag{33}$$

where

$$\left(\psi^{\alpha}_{\text{fib}}\right)_{,\alpha\alpha} = k^{\alpha}_1\left(1 + 2\ k^{\alpha}_2\left(\bar{I}^{\alpha}_4 - 1\right)^2\right)\ \exp\left[k^{\alpha}_2\left(\bar{I}^{\alpha}_4 - 1\right)^2\right] \ . \tag{34}$$

4.2.2 Holzapfel (2005) sedf

The strain energy density proposed by Holzapfel et al. [3] is given by

$$\psi = \psi_{\text{vol}}\left(J\right) + \psi_{\text{mat}}\left(\bar{I}_1\right) + \psi_{\text{fib}}\left(\bar{I}_1, \bar{I}_4\right) \tag{35}$$

with the three parts having explicit forms

$$\psi_{\rm vol} = \frac{\kappa}{2}\left(J-1\right)^2 \ , \tag{36a}$$

$$\psi_{\rm mat} = \frac{\mu}{2}\left(\bar{I}_1 - 3\right) \ , \tag{36b}$$

$$\psi_{\rm fib} = \frac{k_1}{k_2}\left[\exp\left(k_2\left[(1-\rho)\left(\bar{I}_1-3\right)^2 + \rho\left(\bar{I}_4-1\right)^2\right]\right) - 1\right] \ . \tag{36c}$$

Thus, the elastic stress response is governed by the three contributions

$$\tau^{ij}_{\rm vol} = \kappa\ J\left(J-1\right)\ g^{ij} \ , \tag{37a}$$

$$\tau^{ij}_{\rm mat} = \mu\ \left(\mathrm{dev}\,\bar{\mathbf{b}}\right)^{ij} \ , \tag{37b}$$

$$\tau^{ij}_{\rm fib} = \left\{\left(\psi_{\rm fib}\right)_{,\bar{I}_1}\left(\mathrm{dev}\,\bar{\mathbf{b}}\right)^{ij} + \left(\psi_{\rm fib}\right)_{,\bar{I}_4}\left(\mathrm{dev}\,\bar{\mathbf{a}}\right)^{ij}\right\} \ . \tag{37c}$$

where

$$\left(\mathrm{dev}\,\bar{\mathbf{b}}\right)^{ij} = \bar{b}^{ij} - \frac{\bar{I}_1}{3}\,g^{ij} \ , \tag{38a}$$

$$\left(\mathrm{dev}\,\bar{\mathbf{a}}\right)^{ij} = \bar{a}^{ij} - \frac{\bar{I}_4}{3}\,g^{ij} \ , \tag{38b}$$

$$\left(\psi_{\rm fib}\right)_{,\bar{I}_1} = (1-\rho)\left(\bar{I}_1-3\right)\ \mathcal{F} \ , \tag{38c}$$

$$\left(\psi_{\rm fib}\right)_{,\bar{I}_4} = \rho\left(\bar{I}_4-1\right)\ \mathcal{F} \ , \tag{38d}$$

$$\mathcal{F} = 4\ k_1\ \exp\left(k_2\left[(1-\rho)\left(\bar{I}_1-3\right)^2 + \rho\left(\bar{I}_4-1\right)^2\right]\right) \ . \tag{38e}$$

Thus, using the decoupled representation of the Kirshhoff stress tensor, its derivatives straightforward leads to the relations for the volumic and isotropic deviatoric parts:

$$\left(\mathbf{c}_{\rm vol}\right)^{ijkl} = \kappa\ J\left[(2J-1)\,g^{ij}\ g^{kl} - (J-1)\ \left(g^{ik}\,g^{jl} + g^{il}\,g^{jk}\right)\right] \ , \tag{39a}$$

$$\left(\mathbf{c}_{\rm mat}\right)^{ijkl} = \frac{2}{3}\mu\ \left[\frac{\bar{I}_1}{2}\left(g^{ik}\,g^{jl} + g^{il}\,g^{jk}\right) + \frac{\bar{I}_1}{3}g^{ij}\ g^{kl} - \left(g^{ij}\,\bar{b}^{kl} + \bar{b}^{ij}\,g^{kl}\right)\right] . \tag{39b}$$

Finally, for the anisotropic deviatoric part, we used both derivation on J and $\mathbf{a}$ and obtained

$$\begin{aligned}\left(\mathbf{c}_{\rm fib}\right)^{ijkl} = {} & \left(\psi_{\rm fib}\right)_{,\bar{I}_1\bar{I}_1}\left(\mathrm{dev}\,\bar{\mathbf{b}}\right)^{ij}\left(\mathrm{dev}\,\bar{\mathbf{b}}\right)^{kl} + \left(\psi_{\rm fib}\right)_{,\bar{I}_4\bar{I}_4}\left(\mathrm{dev}\,\bar{\mathbf{a}}\right)^{ij}\left(\mathrm{dev}\,\bar{\mathbf{a}}\right)^{kl} \\ & + \left(\psi_{\rm fib}\right)_{,\bar{I}_1\bar{I}_4}\left[\left(\mathrm{dev}\,\bar{\mathbf{b}}\right)^{ij}\left(\mathrm{dev}\,\bar{\mathbf{a}}\right)^{kl} + \left(\mathrm{dev}\,\bar{\mathbf{a}}\right)^{ij}\left(\mathrm{dev}\,\bar{\mathbf{b}}\right)^{kl}\right] \\ & - \frac{2}{3}\left(\psi_{\rm fib}\right)_{,\bar{I}_1}\left(g^{ij}\,\bar{b}^{kl} + \bar{b}^{ij}\,g^{kl}\right) - \frac{2}{3}\left(\psi_{\rm fib}\right)_{,\bar{I}_4}\left(g^{ij}\,\bar{a}^{kl} + \bar{a}^{ij}\,g^{kl}\right) \\ & + \left[\frac{\bar{I}_1}{3}\left(\psi_{\rm fib}\right)_{,\bar{I}_1} + \frac{\bar{I}_4}{3}\left(\psi_{\rm fib}\right)_{,\bar{I}_4}\right] \times \left[\left(g^{ik}g^{jl} + g^{il}g^{jk}\right) + \frac{2}{3}g^{ij}g^{kl}\right]\end{aligned} \tag{40}$$

where

$$\left(\psi_{\rm fib}\right)_{,\bar{I}_1\bar{I}_1} = 2\ (1-\rho)\left[1 + 2\ k_2\,(1-\rho)\left(\bar{I}_1-3\right)^2\right]\ \mathcal{F} \ , \tag{41a}$$

$$\left(\psi_{\rm fib}\right)_{,\bar{I}_1\bar{I}_4} = 2\,\rho\,(1-\rho)\left(\bar{I}_1-3\right)\ \left(\bar{I}_4-1\right)\mathcal{F} \ , \tag{41b}$$

$$\left(\psi_{\rm fib}\right)_{,\bar{I}_4\bar{I}_4} = 2\,\rho\left[1 + 2\ k_2\ \rho\left(\bar{I}_4-1\right)^2\right]\ \mathcal{F} \ . \tag{41c}$$

4.3 Implementation

4.3.1 Element orientation and material anisotropy

In this section, we present the approach used to determine the anisotropic directions. To determine the initial element orientation $\boldsymbol{f}_0$, we assume a "sufficiently good mesh" so as to base its determination on elements orientation, i.e. on the element sides. For 2-dimensional elements, such as `plane 182` or `plane 183` [see Fig.(1)], the fibre direction is determined trough the simple relation

$$\boldsymbol{f}_0 = \frac{(\boldsymbol{X}_i - \boldsymbol{X}_j) + (\boldsymbol{X}_l - \boldsymbol{X}_k)}{\| (\boldsymbol{X}_i - \boldsymbol{X}_j) + (\boldsymbol{X}_l - \boldsymbol{X}_k) \|} , \tag{42}$$

where the subset identified the corresponding node. Similarly, for 3-dimensional elements, such as `solid 185` or `solid 186` [see Fig.(1)], the fibre direction is determined via

$$\boldsymbol{f}_0 = \frac{(\boldsymbol{X}_i - \boldsymbol{X}_j) + (\boldsymbol{X}_l - \boldsymbol{X}_k) + (\boldsymbol{X}_m - \boldsymbol{X}_n) + (\boldsymbol{X}_p - \boldsymbol{X}_o)}{\| (\boldsymbol{X}_i - \boldsymbol{X}_j) + (\boldsymbol{X}_l - \boldsymbol{X}_k) + (\boldsymbol{X}_m - \boldsymbol{X}_n) + (\boldsymbol{X}_p - \boldsymbol{X}_o) \|} . \tag{43}$$

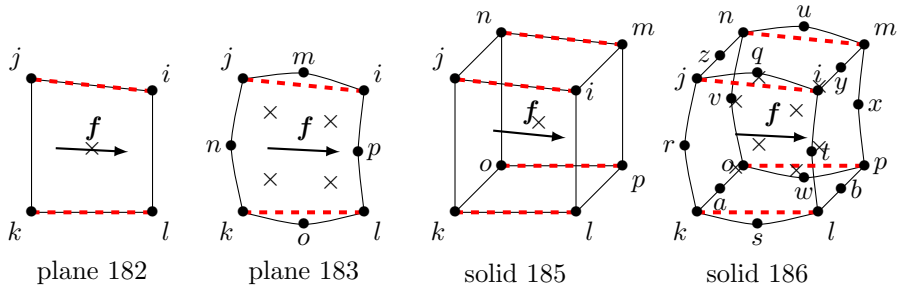

Figure 1: Illustration of determination of ANSYS elements orientation based on the median of highlighted red sides.

Notice that since ANSYS worked with updated coordinates, we first have to get the current node position as well as its current corresponding displacement so as to determine the initial position with

$$\boldsymbol{X}_i = \boldsymbol{x}_i - \boldsymbol{u}_i. \tag{44}$$

Then, once $\boldsymbol{f}_0$ is determined, we can compute the local anisotropic directions $\boldsymbol{f}_0^\alpha$, which then has to be pushed-forward into the spatial direction with

$$\boldsymbol{f}^\alpha = \mathbf{F} \cdot \boldsymbol{f}_0^\alpha , \tag{45}$$

from which we construct the structural tensor

$$\bar{\mathbf{a}}^\alpha = J^{-\frac{2}{3}} \boldsymbol{f}^\alpha \otimes \boldsymbol{f}^\alpha . \tag{46}$$

4.3.2 Voigt notation

Let us first recall that using the symmetry properties of second order tensor, we may introduce the Voigt notation by converting pairs of indices to a single

index via $\{11\ 22\ 33\ 12\ 23\ 31\} = \{11\ 22\ 33\ 21\ 32\ 13\} \Rightarrow \{1\ 2\ 3\ 4\ 5\ 6\}$ such that the Jaumann rate of the Cauchy stress may reads in Voigt notation

$$\{\overset{\nabla}{\boldsymbol{\sigma}}\} = [\ \mathcal{C}\ ] \cdot \{\mathbf{d}\} \tag{47}$$

with $\{\mathbf{d}\}$ the Voigt notation of the rate of deformation tensor $\mathbf{d} = \mathrm{sym}\,(\nabla_x \boldsymbol{\varphi})$, explicitly defined by

$$\{\mathbf{d}\} = \left\{\ d_{11} \quad d_{22} \quad d_{33} \quad 2d_{12} \quad 2d_{23} \quad 2d_{13}\ \right\}^T. \tag{48}$$

Accordingly, the fourth order Jaumann tangent stiffness may be recast in the following form

$$[\,\mathcal{C}\,] = \begin{bmatrix} \mathcal{C}_{1111} & \mathcal{C}_{1122} & \mathcal{C}_{1133} & \mathcal{C}_{1112} & \mathcal{C}_{1123} & \mathcal{C}_{1113} \\ \propto & \mathcal{C}_{2222} & \mathcal{C}_{2233} & \mathcal{C}_{2212} & \mathcal{C}_{2223} & \mathcal{C}_{2213} \\ \propto & \propto & \mathcal{C}_{3333} & \mathcal{C}_{3312} & \mathcal{C}_{3323} & \mathcal{C}_{3313} \\ \propto & \propto & \propto & \mathcal{C}_{1212} & \mathcal{C}_{1223} & \mathcal{C}_{1213} \\ \propto & \propto & \propto & \propto & \mathcal{C}_{2323} & \mathcal{C}_{2313} \\ \propto & \propto & \propto & \propto & \propto & \mathcal{C}_{1313} \end{bmatrix}, \tag{49}$$

where the symbol $\propto$ stands for symmetric terms. Notice that the stress in the tangent stiffness are

$$[\,\mathcal{C}\,] - \frac{1}{J}[\,\mathsf{c}\,] = \frac{1}{2}\begin{bmatrix} 4\sigma_1 & 0 & 0 & 2\sigma_4 & 0 & 2\sigma_6 \\ \propto & 4\sigma_2 & 0 & 2\sigma_4 & 2\sigma_5 & 0 \\ \propto & \propto & 4\sigma_3 & 0 & 2\sigma_5 & 2\sigma_6 \\ \propto & \propto & \propto & \sigma_1+\sigma_2 & \sigma_6 & \sigma_5 \\ \propto & \propto & \propto & \propto & \sigma_2+\sigma_3 & \sigma_4 \\ \propto & \propto & \propto & \propto & \propto & \sigma_1+\sigma_3 \end{bmatrix}. \tag{50}$$

Thus, once the Cauchy stress $\{\boldsymbol{\sigma}\}$ and the Jaumann tangent stiffness $[\,\mathcal{C}\,]$ are determined, they need to be expressed into the corotated frame. Denoting $[\,\mathbf{Q}\,]$ the change of basis matrix, it is defined from the orthogonal rotation tensor $\mathbf{R}^{-1} \equiv \mathbf{R}^{\mathrm{T}}$ by

$$[\,\mathbf{Q}\,] = \begin{bmatrix} [\,\mathbf{Q}_{11}\,] & [\,\mathbf{Q}_{12}\,] \\ [\,\mathbf{Q}_{21}\,] & [\,\mathbf{Q}_{22}\,] \end{bmatrix}. \tag{51}$$

with

$$[\,\mathbf{Q}_{11}\,] = \begin{bmatrix} R_{11}^2 & R_{21}^2 & R_{31}^2 \\ R_{12}^2 & R_{22}^2 & R_{32}^2 \\ R_{13}^2 & R_{23}^2 & R_{33}^2 \end{bmatrix}, \tag{52}$$

$$[\,\mathbf{Q}_{12}\,] = 2\begin{bmatrix} R_{11}R_{21} & R_{21}R_{31} & R_{11}R_{31} \\ R_{12}R_{22} & R_{22}R_{32} & R_{12}R_{32} \\ R_{13}R_{23} & R_{23}R_{33} & R_{13}R_{33} \end{bmatrix}, \tag{53}$$

$$[\,\mathbf{Q}_{21}\,] = \begin{bmatrix} R_{11}R_{12} & R_{21}R_{22} & R_{31}R_{32} \\ R_{12}R_{13} & R_{22}R_{23} & R_{32}R_{33} \\ R_{13}R_{11} & R_{23}R_{21} & R_{33}R_{31} \end{bmatrix}, \tag{54}$$

$$[\,\mathbf{Q}_{22}\,] = \begin{bmatrix} R_{11}R_{22}+R_{21}R_{12} & R_{21}R_{32}+R_{31}R_{22} & R_{11}R_{32}+R_{31}R_{12} \\ R_{12}R_{23}+R_{22}R_{13} & R_{22}R_{33}+R_{32}R_{23} & R_{12}R_{33}+R_{32}R_{13} \\ R_{13}R_{21}+R_{23}R_{11} & R_{23}R_{31}+R_{33}R_{21} & R_{13}R_{31}+R_{33}R_{11} \end{bmatrix}. \tag{55}$$

Thus, we may obtain the corotated Cauchy stress and Jaumann tangent stiffness with the relations

$$\{\hat{\boldsymbol{\sigma}}\} = [\,\mathbf{Q}\,] \cdot \{\boldsymbol{\sigma}\}\ , \tag{56}$$

$$[\,\hat{\mathcal{C}}\,] = [\,\mathbf{Q}\,] \cdot [\,\mathcal{C}\,] \cdot [\,\mathbf{Q}\,]^T\ . \tag{57}$$

This formulation is implemented by means of the `trans_matrx_6()` subrountine. This is all what is needed to implement a behaviour law into ANSYS by means of the user subroutine `usermat3d()`.

References

[1] Baek, S., Gleason, R., Rajagopal, K., Humphrey, J., 2007. Theory of small on large: Potential utility in computations of fluid–solid interactions in arteries. Computer Methods in Applied Mechanics and Engineering 196 (31-32), 3070–3078.

[2] Holzapfel, G., Gasser, T., Ogden, R., 2000. A New Constitutive Framework for Arterial Wall Mechanics and a Comparative Study of Material Models. Journal of Elasticity 61 (1), 1–48.

[3] Holzapfel, G. A., Sommer, G., Gasser, C. T., Regitnig, P., Nov 2005. Determination of layer-specific mechanical properties of human coronary arteries with nonatherosclerotic intimal thickening and related constitutive modeling. Am J Physiol Heart Circ Physiol 289 (5), H2048–H2058.

[4] Simo, J., Hughes, T., 2000. Computational inelasticity. Springer.

A Ansys `usermat3d()` subroutine

```
!***********************************************************************
! *** anisotropic hyperelastic models ***
! author:        Nicolas Mesnier
! creation:      06/02/2010
! last modified: 07/03/2011
!-----------------------------------------------------------------------
!  Models:
!      1 - Holzapfel 2000 hyperelastic anisotropic compressible law [1]
!      2 - model 1 with two additional fibre family [2]
!      3 - Holzapfel 2005 hyperelastic anisotropic compressible law [3]
!
!  References :
!     [1] Holzapfel et al, J Elasticity 61 (2000)
!     [2] Baek et al, Comp Meth Appl Mech Eng 196 (2007)
!     [3] Holzapfel et al, AJP 289 (2005)
!
!***********************************************************************
*deck,usermat3d  parallel          user                                  nmesnier
      subroutine usermat3d(matId, elemId,kDomIntPt, kLayer, kSectPt,
     &                  ldstep,isubst,keycut,
     &                  nDirect,nShear,ncomp,nStatev,nProp,
     &                  Time,dTime,Temp,dTemp,
     &                  stress,ustatev,dsdePl,sedEl,sedPl,epseq,
     &                  Strain,dStrain, epsPl, prop, coords,
     &                  var0, F0, F1,
     &                  tsstif, epsZZ,
     &                  var1, var2, var3, var4, var5,
     &                  var6, var7, var8)
!-----------------------------------------------------------------------
      IMPLICIT NONE
      integer,parameter :: RK=KIND(1.D0)        ! real kind
! external procedures
      external polarRU                          ! polar decomposition subroutine
      external trans_matrx_6                    ! 6D voigt rotation matrix subroutine
      external elmget                           ! ANSYS fct, getting element nodes
      external ndgxyz                           ! ANSYS fct, get node position
      external dspget                           ! ANSYS fct, get node solution
! input arguments
      integer, intent(IN) :: matID              ! material ID number
      integer, intent(IN) :: elemID             ! element ID number
      integer, intent(IN) :: kDomIntPt          ! integration point
      integer, intent(IN) :: kLayer             ! layer number
      integer, intent(IN) :: kSectPt            ! section number
      integer, intent(IN) :: ldstep             ! load step number
      integer, intent(IN) :: isubst             ! substep number
      integer, intent(IN) :: ncomp              ! no. stress components
      integer, intent(IN) :: nDirect            ! no. direct stress components
      integer, intent(IN) :: nShear             ! no. shear stress components
      integer, intent(IN) :: nStatev            ! size of state variable array
      integer, intent(IN) :: nProp              ! size of property array
      real(RK),intent(IN) :: Time               ! time at begining of increment
      real(RK),intent(IN) :: dTime              ! time increment
      real(RK),intent(IN) :: Temp               ! temperature at begining of increment
      real(RK),intent(IN) :: dTemp              ! temperature increment
      real(RK),intent(IN) :: Strain(ncomp)      ! strain at begining of inc
      real(RK),intent(IN) :: dStrain(ncomp)     ! strain increment
      real(RK),intent(IN) :: epsPl(ncomp)       ! woRK space
      real(RK),intent(IN) :: prop(nProp)        ! mat props, user input
      real(RK),intent(IN) :: coords(3)          ! coordinates of material point
      real(RK),intent(IN) :: F1(3,3)            ! deformation gradient at t+dt
      real(RK),intent(IN) :: F0(3,3)            ! deformation gradient at t
      real(RK),intent(IN) :: var0(*),var1,var2,var3,var4 !place holder
      real(RK),intent(IN) :: var5,var6,var7,var8 !place holder
! input output arguments
      real(RK),intent(INOUT) :: sedEl           ! stored (elastic) energy
      real(RK),intent(INOUT) :: sedPl           ! dissipated (plastic) energy
      real(RK),intent(INOUT) :: epseq           ! equivalent plastic strain
      real(RK),intent(INOUT) :: epsZZ           ! plane stress thickness strain
      real(RK),intent(INOUT) :: stress(ncomp)   ! stress
      real(RK),intent(INOUT) :: ustatev(nStatev) !state variables
! output arguments
      integer, intent(OUT) :: keycut            ! time step reduction
      real(RK),intent(OUT) :: dsdePl(ncomp,ncomp) !tangent stiffness
      real(RK),intent(OUT) :: tsstif(2)         ! transverse shear stiffness
```

```
!-----------------------------------------------------------------------
      integer  :: i,j,k,l
      real(RK), parameter :: pi =  3.14159265358979324_RK
      real(RK) :: mu,bm,K1(4),K2(4),beta,rho    ! material parameters
      real(RK) :: model                         ! material model
!-----------------------------------------------------------------------
      real(RK) :: detF,detF23                   ! Jacobian and factor
      real(RK) :: Bbar(6)                       ! isochoric left cauchy green
!-----------------------------------------------------------------------
      integer  :: elmget,ndgxyz,dspget          ! ANSYS fct
      integer  :: tmp,Item(2),ElemData(10)
      integer  :: nodes(8)                      ! node vector, depends of elements
      real(RK) :: Disp(2)                       ! node displacement
      real(RK) :: ActnodePos(3)                 ! current position of a node
      real(RK) :: nodesPos(4,2)                 ! 2D nodes positions
      real(RK) :: uElemOrientVec(2)             ! unormed element orientation
      real(RK) :: normElemOrient                ! norm of uElemOrientVec
      real(RK) :: ElemOrientVec(2)              ! element orientation
      real(RK) :: fib(4,3),spafib(4,3)          ! orthoradial, longitudinal and crossed material and
spatial fibers
      real(RK) :: A(4,6)                        ! spatial structural tensor
!-----------------------------------------------------------------------
      real(RK) :: I1bar                         ! first invariant
      real(RK) :: I4bar(4)                      ! fourth invariants
!-----------------------------------------------------------------------
      real(RK) :: I1bar3                        ! I1/3
      real(RK) :: dw1(4),dw2(4)                 ! first and second sedf derivatives w.r.t. I4bar(i)
      real(RK) :: expf(4)                       ! fiber exponential factor
      real(RK) :: dw2f(4)                       ! fiber factor for second derivatives
      real(RK) :: C1,C2                         ! constants
!-----------------------------------------------------------------------
      real(RK) :: sigma(6)                      ! cauchy stress
      real(RK) :: M(6,6)                        ! material jacobian
      real(RK) :: stressfull(6)                 ! full 3D stress
      real(RK) :: dsdePlfull(6,6)               ! full 3D tangent stiffness
      real(RK) :: R(3,3), Q(6,6)                ! rotations
      real(RK) :: U(3,3)                        ! stretch
!-----------------------------------------------------------------------
      real(RK) :: EVR(3)                         ! E_R vector
      real(RK) :: EVT(3)                         ! E_Theta vector
      real(RK) :: normspaevt                     ! norm e_theta vector
      real(RK) :: spaevt(2)                      ! spatial element orientation
!***********************************************************************
!-----------------------------------------------------------------------
! *** kinematics
!-----------------------------------------------------------------------
! Jacobian
      detF = F1(1,1) * (F1(2,2)*F1(3,3) - F1(2,3)*F1(3,2))
     &     + F1(1,2) * (F1(2,3)*F1(3,1) - F1(2,1)*F1(3,3))
     &     + F1(1,3) * (F1(2,1)*F1(3,2) - F1(2,2)*F1(3,1))
! time step reduction condition
      if(detF<=0._RK)then
        keycut=1
        return
      else
        detF23 = detF**(-2._RK/3._RK)
      endif
! isochoric left Cauchy Green
      Bbar(1)=(F1(1,1)*F1(1,1)+F1(1,2)*F1(1,2)+F1(1,3)*F1(1,3))*detF23 ! Bbar(1,1)
      Bbar(2)=(F1(2,1)*F1(2,1)+F1(2,2)*F1(2,2)+F1(2,3)*F1(2,3))*detF23 ! Bbar(2,2)
      Bbar(3)=(F1(3,1)*F1(3,1)+F1(3,2)*F1(3,2)+F1(3,3)*F1(3,3))*detF23 ! Bbar(3,3)
      Bbar(4)=(F1(2,1)*F1(1,1)+F1(2,2)*F1(1,2)+F1(2,3)*F1(1,3))*detF23 ! Bbar(2,1)
      Bbar(5)=(F1(3,1)*F1(2,1)+F1(3,2)*F1(2,2)+F1(3,3)*F1(2,3))*detF23 ! Bbar(3,2)
      Bbar(6)=(F1(3,1)*F1(1,1)+F1(3,2)*F1(1,2)+F1(3,3)*F1(1,3))*detF23 ! Bbar(3,1)
! -----------------------------------------------------------------------
! *** get elements orientation
      if (Time.lt.1._RK) then
!-----------------------------------------------------------------------
      tmp = elmget(elemId,ElemData,nodes)       ! get element nodes
      Item(:) = (/1,2/)                         ! solution
! get node position
      do i=1,4
        tmp = ndgxyz(nodes(i),ActnodePos)
        tmp = dspget(nodes(i),2,Item,Disp)
```

```
        nodesPos(i,1:2)=ActnodePos(1:2)-Disp(1:2)
        ActnodePos(1:3) = 0._RK
      end do
! element orientation unormed vector
      uElemOrientVec(:) = (nodesPos(3,1:2)-nodesPos(2,1:2))
     &                  + (nodesPos(4,1:2)-nodesPos(1,1:2))
! element orientation normed version
      normElemOrient = sqrt(uElemOrientVec(1)*uElemOrientVec(1)
     &                    + uElemOrientVec(2)*uElemOrientVec(2))
      ElemOrientVec(1:2) = uElemOrientVec(1:2)/normElemOrient
! ------------------------------------------------------------------------
      ustatev(1) = ElemOrientVec(1)
      ustatev(2) = ElemOrientVec(2)
! ------------------------------------------------------------------------
      else
! ------------------------------------------------------------------------
      ElemOrientVec(1) = ustatev(1)
      ElemOrientVec(2) = ustatev(2)
! ------------------------------------------------------------------------
      end if
      EVR(1:3) = (/ElemOrientVec(2),-ElemOrientVec(1),0._RK/)
      EVT(1:3) = (/ElemOrientVec(1),ElemOrientVec(2),0._RK/)
!-------------------------------------------------------------------------
! *** model choice
!-------------------------------------------------------------------------
      model = prop(1)
      if(model.eq.1._RK)then
!************************************************************************ << model 1
! model 1:       anisotropic hyperelastic two fibre families model
! author:        Nicolas Mesnier
! creation date: 06/02/2010
! last modified: 19/07/2010
!-------------------------------------------------------------------------
!  The behaviour is described by the holzapfel hyperelastic anisotropic
!  compressible law [1]
!       W = Wiso + Wvol + Waniso
!  where each contribution is defined as:
!       Wiso = mu /2 * (I1 - 3 )
!       Wvol = K / 2 * (J - 1 ) ^2
!       Waniso = k1i / (2*k2)* { exp[ k2 * (I4 - 1)^2 ] - 1 }
!  with the two fiber family:
!       fib1 = cos(beta) Et + sin(beta) Ez
!       fib2 = cos(beta) Et - sin(beta) Ez
!  Input material properties:
!       mu   : shear modulus
!       k    : bulk modulus
!       k1   : stress-like
!       k2   : dimensionless
!       beta : angle in degrees, 0° < beta < 90°
!  References :
!     [1] G.Holzapfel et al, J Elasticity 61 (2000)
!************************************************************************
!-------------------------------------------------------------------------
! *** input material parameters
!-------------------------------------------------------------------------
      mu    = prop(2)           ! shear modulus
      bm    = prop(3)           ! bulk modulus
      K1(1) = prop(4)           ! aniso parameter (stress-like)
      K2(1) = prop(5)           ! aniso parameter (dimensionless)
      K1(2) = prop(4)
      K2(2) = prop(5)
      beta  = prop(6)*pi/180._RK ! fiber angle (radians)
!-------------------------------------------------------------------------
! *** material anisotropy
!-------------------------------------------------------------------------
! init fibers
      fib(:,:) = 0._RK
! crossed fibers
      fib(1,1:3) = (/ElemOrientVec(1)*cos(beta),
     &               ElemOrientVec(2)*cos(beta),sin(beta)/)
      fib(2,1:3) = (/ElemOrientVec(1)*cos(beta),
     &               ElemOrientVec(2)*cos(beta),-sin(beta)/)
      fib(3,1:3) = (/ElemOrientVec(1)*cos(beta),
     &               ElemOrientVec(2)*cos(beta),sin(beta)/)
```

```
      fib(4,1:3) = (/ElemOrientVec(1)*cos(beta),
     &               ElemOrientVec(2)*cos(beta),-sin(beta)/)
!-----------------------------------------------------------------------
! init fibers
!      fib(:,:) = 0._RK
! orthoradial fiber
!      fib(3,1:2) = (/ElemOrientVec(1),ElemOrientVec(2)/)
! crossed fibers
!      fib(1,1:3) = (/fib(3,1)*cos(beta),fib(3,2)*cos(beta),sin(beta)/)
!      fib(2,1:3) = (/fib(3,1)*cos(beta),fib(3,2)*cos(beta),-sin(beta)/)
! spatial fibers
      do k=1,4
        spafib(k,:)= (/
     &          F1(1,1)*fib(k,1)+F1(1,2)*fib(k,2)+F1(1,3)*fib(k,3),
     &          F1(2,1)*fib(k,1)+F1(2,2)*fib(k,2)+F1(2,3)*fib(k,3),
     &          F1(3,1)*fib(k,1)+F1(3,2)*fib(k,2)+F1(3,3)*fib(k,3)/)
      end do
! spatial anisotropic tensors
      do k=1,4
        A(k,:) = (/ detF23*spafib(k,1)*spafib(k,1),
     &              detF23*spafib(k,2)*spafib(k,2),
     &              detF23*spafib(k,3)*spafib(k,3),
     &              detF23*spafib(k,2)*spafib(k,1),
     &              detF23*spafib(k,3)*spafib(k,2),
     &              detF23*spafib(k,3)*spafib(k,1)/)
      end do
!-----------------------------------------------------------------------
! *** invariants
!-----------------------------------------------------------------------
! 1st invariant
      I1bar = (Bbar(1) + Bbar(2) + Bbar(3))
      I1bar3 = I1bar/3._RK      ! I1bar / 3
! 4th invariants
      I4bar(:) = (/A(1,1)+A(1,2)+A(1,3),
     &            A(2,1)+A(2,2)+A(2,3),
     &            A(3,1)+A(3,2)+A(3,3),
     &            A(4,1)+A(4,2)+A(4,3)/)
!-----------------------------------------------------------------------
! *** anisotropic sedf derivatives
!-----------------------------------------------------------------------
      do k=1,2
! fiber exponential factor
        expf(k)  = exp(K2(k)*(I4bar(k)-1._RK)**2)
! fiber factor for second derivatives
        dw2f(k)  = 1._RK + 2._RK* K2(k)*(I4bar(k)-1._RK)**2
! sedf derivative w.r.t. I4bar(i)
        dw1(k) = K1(k)*(I4bar(k)-1._RK)*expf(k)
! sedf derivative w.r.t. I4bar(i), I4bar(i)
        dw2(k) = K1(k)*dw2f(k)*expf(k)
      end do
!-----------------------------------------------------------------------
! *** update sedf
!-----------------------------------------------------------------------
      sedEl = mu*(I1bar-3._RK) + bm*(detF-1._RK)**2
      do k=1,2
        sedEl = sedEl + K1(k)/K2(k)*(expf(k)-1._RK)
      end do
      sedEl = sedEl/2._RK /detF
      sedPl = 0._RK
!-----------------------------------------------------------------------
! *** cauchy stress in spatial frame
!-----------------------------------------------------------------------
! isotropic contributions
      sigma(1:3) = mu *(Bbar(1:3)-I1bar3)/ detF + bm*(detF-1._RK)
      sigma(4:6) = mu*(Bbar(4:6)) / detF
! add anisotropic parts
      do k=1,2
        sigma(1:3) = sigma(1:3)
     & + 2._RK*dw1(k)/detF*(A(k,1:3)-I4bar(k)/3._RK)
        sigma(4:6) = sigma(4:6) + 2._RK*dw1(k)/detF*A(k,4:6)
      end do
!-----------------------------------------------------------------------
! *** Jaumann tangent stiffness in spatial frame
!-----------------------------------------------------------------------
```

```
! factors
      C1 = 2._RK/3._RK*mu/detF
      C2 = 3._RK/4._RK*C1
! isotropic deviatoric components
!-------------------------------------------------------------------------
      M(1,1) = C1 * (Bbar(1) + I1bar3)
      M(2,2) = C1 * (Bbar(2) + I1bar3)
      M(3,3) = C1 * (Bbar(3) + I1bar3)
!-----
      M(2,1) =-C1 * (Bbar(1) + Bbar(2) - I1bar3)
      M(3,1) =-C1 * (Bbar(1) + Bbar(3) - I1bar3)
      M(3,2) =-C1 * (Bbar(2) + Bbar(3) - I1bar3)
!-----
      M(4,1) = C1 * Bbar(4) / 2._RK
      M(6,1) = C1 * Bbar(6) / 2._RK
      M(4,2) = C1 * Bbar(4) / 2._RK
      M(5,2) = C1 * Bbar(5) / 2._RK
      M(5,3) = C1 * Bbar(5) / 2._RK
      M(6,3) = C1 * Bbar(6) / 2._RK
!-----
      M(5,1) =-C1 * Bbar(5)
      M(6,2) =-C1 * Bbar(6)
      M(4,3) =-C1 * Bbar(4)
!-----
      M(4,4) = C2 * (Bbar(1) + Bbar(2))
      M(5,5) = C2 * (Bbar(2) + Bbar(3))
      M(6,6) = C2 * (Bbar(1) + Bbar(3))
!-----
      M(5,4) = C2 * Bbar(6)
      M(6,4) = C2 * Bbar(5)
      M(6,5) = C2 * Bbar(4)
! anisotropic deviatoric components
!-------------------------------------------------------------------------
      do k=1,2
!-----
      M(1,1)=M(1,1)+4._RK/detF*(dw2(k)*(A(k,1)-I4bar(k)/3._RK)**2
     &      + dw1(k)/3._RK*(A(k,1)+I4bar(k)/3._RK))
      M(2,2)=M(2,2)+4._RK/detF*(dw2(k)*(A(k,2)-I4bar(k)/3._RK)**2
     &      + dw1(k)/3._RK*(A(k,2)+I4bar(k)/3._RK))
      M(3,3)=M(3,3)+4._RK/detF*(dw2(k)*(A(k,3)-I4bar(k)/3._RK)**2
     &      + dw1(k)/3._RK*(A(k,3)+I4bar(k)/3._RK))
!-----
      M(2,1)=M(2,1)+4._RK/detF*(
     &   dw2(k)*(A(k,2)-I4bar(k)/3._RK)*(A(k,1)-I4bar(k)/3._RK)
     & + dw1(k)/3._RK*(I4bar(k)/3._RK-A(k,2)-A(k,1)))
      M(3,1)=M(3,1)+4._RK/detF*(
     &   dw2(k)*(A(k,3)-I4bar(k)/3._RK)*(A(k,1)-I4bar(k)/3._RK)
     & + dw1(k)/3._RK*(I4bar(k)/3._RK-A(k,3)-A(k,1)))
      M(3,2)=M(3,2)+4._RK/detF*(
     &   dw2(k)*(A(k,3)-I4bar(k)/3._RK)*(A(k,2)-I4bar(k)/3._RK)
     & + dw1(k)/3._RK*(I4bar(k)/3._RK-A(k,3)-A(k,2)))
!-----
      M(4,1)=M(4,1)+2._RK/detF*(
     &   2._RK*dw2(k)*(A(k,1)-I4bar(k)/3._RK)*A(k,4)
     & + dw1(k)/3._RK*A(k,4))
      M(6,1)=M(6,1)+2._RK/detF*(
     &   2._RK*dw2(k)*(A(k,1)-I4bar(k)/3._RK)*A(k,6)
     & + dw1(k)/3._RK*A(k,6))
      M(4,2)=M(4,2)+2._RK/detF*(
     &   2._RK*dw2(k)*(A(k,2)-I4bar(k)/3._RK)*A(k,4)
     & + dw1(k)/3._RK*A(k,4))
      M(5,2)=M(5,2)+2._RK/detF*(
     &   2._RK*dw2(k)*(A(k,2)-I4bar(k)/3._RK)*A(k,5)
     & + dw1(k)/3._RK*A(k,5))
      M(5,3)=M(5,3)+2._RK/detF*(
     &   2._RK*dw2(k)*(A(k,3)-I4bar(k)/3._RK)*A(k,5)
     & + dw1(k)/3._RK*A(k,5))
      M(6,3)=M(6,3)+2._RK/detF*(
     &   2._RK*dw2(k)*(A(k,3)-I4bar(k)/3._RK)*A(k,6)
     & + dw1(k)/3._RK*A(k,6))
!-----
      M(5,1)=M(5,1)+4._RK/detF*(dw2(k)*(A(k,1)-I4bar(k)/3._RK)*A(k,5)
     & - dw1(k)/3._RK*A(k,5))
      M(6,2)=M(6,2)+4._RK/detF*(dw2(k)*(A(k,2)-I4bar(k)/3._RK)*A(k,6)
```

```
     & - dw1(k)/3._RK*A(k,6))
      M(4,3)=M(4,3)+4._RK/detF*(dw2(k)*(A(k,3)-I4bar(k)/3._RK)*A(k,4)
     & - dw1(k)/3._RK*A(k,4))
!-----
      M(4,4)=M(4,4)+1._RK/detF*(4._RK*dw2(k)*(A(k,4))**2
     & + dw1(k)*(A(k,1)+A(k,2)))
      M(5,5)=M(5,5)+1._RK/detF*(4._RK*dw2(k)*(A(k,5))**2
     & + dw1(k)*(A(k,2)+A(k,3)))
      M(6,6)=M(6,6)+1._RK/detF*(4._RK*dw2(k)*(A(k,6))**2
     & + dw1(k)*(A(k,1)+A(k,3)))
!-----
      M(5,4)=M(5,4)+1._RK/detF*(4._RK*dw2(k)*A(k,4)*A(k,5)
     & + dw1(k)*A(k,6))
      M(6,4)=M(6,4)+1._RK/detF*(4._RK*dw2(k)*A(k,4)*A(k,6)
     & + dw1(k)*A(k,5))
      M(6,5)=M(6,5)+1._RK/detF*(4._RK*dw2(k)*A(k,5)*A(k,6)
     & + dw1(k)*A(k,4))
      end do
!**********************************************************************
      end if
      if(model.eq.2._RK)then
!********************************************************************** << model 2
! model 2:       anisotropic hyperelastic four fibre families model
! author:        Nicolas Mesnier
! creation date: 06/02/2010
! last modified: 19/07/2010
!-----------------------------------------------------------------------
!  The behaviour is described by the holzapfel hyperelastic anisotropic
!  compressible law [1] with two additional fibre family [2]
!       W = Wiso + Wvol + Waniso
!  where each contribution is defined as:
!       Wiso = mu /2 * ( I1 - 3 )
!       Wvol = KK /2 * ( J - 1 ) ^2
!       Waniso(i) = k1i / (2*k2i) * { exp[ k2i * (I4i - 1)^2 ] - 1 }
!  with the four fiber family:
!       fib1 = Et
!       fib2 = Ez
!       fib3 = cos(beta) Et + sin(beta) Ez
!       fib4 = cos(beta) Et - sin(beta) Ez
!  Input material properties:
!       mu   : shear modulus
!       k    : bulk modulus
!       k1i  : stress-like,   k13 = k14
!       k2i  : dimensionless, k23 = k24
!       beta : angle in degrees, 0° < beta < 90°
!  References :
!     [1] G.Holzapfel et al, J Elasticity 61 (2000)
!     [2] Baek et al, Comp Meth Appl Mech Eng 196 (2007)
!**********************************************************************
!-----------------------------------------------------------------------
! *** input material parameters
!-----------------------------------------------------------------------
      mu    = prop(2)           ! shear modulus
      bm    = prop(3)           ! bulk modulus
      K1(1) = prop(4)           ! aniso parameter (stress-like)
      K2(1) = prop(5)           ! aniso parameter (dimensionless)
      K1(2) = prop(6)
      K2(2) = prop(7)
      K1(3) = prop(8)
      K2(3) = prop(9)
      K1(4) = prop(8)
      K2(4) = prop(9)
      beta  = prop(10)*pi/180._RK ! fiber angle (radians)
!-----------------------------------------------------------------------
! *** material anisotropy
!-----------------------------------------------------------------------
! init fibers
      fib(:,:) = 0._RK
! orthoradial fiber
      fib(1,1:2) = (/ElemOrientVec(1),ElemOrientVec(2)/)
! longitudinadial fiber
      fib(2,3) = 1._RK
! crossed fibers
      fib(3,1:3) = (/fib(1,1)*cos(beta),fib(1,2)*cos(beta),sin(beta)/)
```

```
      fib(4,1:3) = (/fib(1,1)*cos(beta),fib(1,2)*cos(beta),-sin(beta)/)
! spatial fibers
      do k=1,4
        spafib(k,:)= (/
     &          F1(1,1)*fib(k,1)+F1(1,2)*fib(k,2)+F1(1,3)*fib(k,3),
     &          F1(2,1)*fib(k,1)+F1(2,2)*fib(k,2)+F1(2,3)*fib(k,3),
     &          F1(3,1)*fib(k,1)+F1(3,2)*fib(k,2)+F1(3,3)*fib(k,3)/)
      end do
! spatial anisotropic tensors
      do k=1,4
        A(k,:) = (/ detF23*spafib(k,1)*spafib(k,1),
     &              detF23*spafib(k,2)*spafib(k,2),
     &              detF23*spafib(k,3)*spafib(k,3),
     &              detF23*spafib(k,2)*spafib(k,1),
     &              detF23*spafib(k,3)*spafib(k,2),
     &              detF23*spafib(k,3)*spafib(k,1)/)
      end do
!-----------------------------------------------------------------------
! *** invariants
!-----------------------------------------------------------------------
! 1st invariant
      I1bar = (Bbar(1) + Bbar(2) + Bbar(3))
      I1bar3 = I1bar/3._RK      ! I1bar / 3
! 4th invariants
      I4bar(:) = (/A(1,1)+A(1,2)+A(1,3),
     &            A(2,1)+A(2,2)+A(2,3),
     &            A(3,1)+A(3,2)+A(3,3),
     &            A(4,1)+A(4,2)+A(4,3)/)
!-----------------------------------------------------------------------
! *** anisotropic sedf derivatives
!-----------------------------------------------------------------------
      do k=1,4
! fiber exponential factor
        expf(k)  = exp(K2(k)*(I4bar(k)-1._RK)**2)
! fiber factor for second derivatives
        dw2f(k)  = 1._RK + 2._RK* K2(k)*(I4bar(k)-1._RK)**2
! sedf derivative w.r.t. I4bar(i)
        dw1(k) = K1(k)*(I4bar(k)-1._RK)*expf(k)
! sedf derivative w.r.t. I4bar(i), I4bar(i)
        dw2(k) = K1(k)*dw2f(k)*expf(k)
      end do
!-----------------------------------------------------------------------
! *** update sedf
!-----------------------------------------------------------------------
      sedEl = mu*(I1bar-3._RK) + bm*(detF-1._RK)**2
      do k=1,4
        sedEl = sedEl + K1(k)/K2(k)*(expf(k)-1._RK)
      end do
      sedEl = sedEl/2._RK /detF
      sedPl = 0._RK
!-----------------------------------------------------------------------
! *** cauchy stress in spatial frame
!-----------------------------------------------------------------------
! isotropic contributions
      sigma(1:3) = mu *(Bbar(1:3)-I1bar3)/ detF + bm*(detF-1._RK)
      sigma(4:6) = mu*(Bbar(4:6)) / detF
! add anisotropic parts
      do k=1,4
        sigma(1:3) = sigma(1:3)
     & + 2._RK*dw1(k)/detF*(A(k,1:3)-I4bar(k)/3._RK)
        sigma(4:6) = sigma(4:6) + 2._RK*dw1(k)/detF*A(k,4:6)
      end do
!-----------------------------------------------------------------------
! *** Jaumann tangent stiffness in spatial frame
!-----------------------------------------------------------------------
! factors
      C1 = 2._RK/3._RK*mu/detF
      C2 = 3._RK/4._RK*C1
! isotropic deviatoric components
!-----------------------------------------------------------------------
      M(1,1) = C1 * (Bbar(1) + I1bar3)
      M(2,2) = C1 * (Bbar(2) + I1bar3)
      M(3,3) = C1 * (Bbar(3) + I1bar3)
!-----
```

```
      M(2,1) =-C1 * (Bbar(1) + Bbar(2) - I1bar3)
      M(3,1) =-C1 * (Bbar(1) + Bbar(3) - I1bar3)
      M(3,2) =-C1 * (Bbar(2) + Bbar(3) - I1bar3)
!-----
      M(4,1) = C1 * Bbar(4) / 2._RK
      M(6,1) = C1 * Bbar(6) / 2._RK
      M(4,2) = C1 * Bbar(4) / 2._RK
      M(5,2) = C1 * Bbar(5) / 2._RK
      M(5,3) = C1 * Bbar(5) / 2._RK
      M(6,3) = C1 * Bbar(6) / 2._RK
!-----
      M(5,1) =-C1 * Bbar(5)
      M(6,2) =-C1 * Bbar(6)
      M(4,3) =-C1 * Bbar(4)
!-----
      M(4,4) = C2 * (Bbar(1) + Bbar(2))
      M(5,5) = C2 * (Bbar(2) + Bbar(3))
      M(6,6) = C2 * (Bbar(1) + Bbar(3))
!-----
      M(5,4) = C2 * Bbar(6)
      M(6,4) = C2 * Bbar(5)
      M(6,5) = C2 * Bbar(4)
! anisotropic deviatoric components
!-----------------------------------------------------------------------
      do k=1,4
!-----
      M(1,1)=M(1,1)+4._RK/detF*(dw2(k)*(A(k,1)-I4bar(k)/3._RK)**2
     &      + dw1(k)/3._RK*(A(k,1)+I4bar(k)/3._RK))
      M(2,2)=M(2,2)+4._RK/detF*(dw2(k)*(A(k,2)-I4bar(k)/3._RK)**2
     &      + dw1(k)/3._RK*(A(k,2)+I4bar(k)/3._RK))
      M(3,3)=M(3,3)+4._RK/detF*(dw2(k)*(A(k,3)-I4bar(k)/3._RK)**2
     &      + dw1(k)/3._RK*(A(k,3)+I4bar(k)/3._RK))
!-----
      M(2,1)=M(2,1)+4._RK/detF*(
     &   dw2(k)*(A(k,2)-I4bar(k)/3._RK)*(A(k,1)-I4bar(k)/3._RK)
     & + dw1(k)/3._RK*(I4bar(k)/3._RK-A(k,2)-A(k,1)))
      M(3,1)=M(3,1)+4._RK/detF*(
     &   dw2(k)*(A(k,3)-I4bar(k)/3._RK)*(A(k,1)-I4bar(k)/3._RK)
     & + dw1(k)/3._RK*(I4bar(k)/3._RK-A(k,3)-A(k,1)))
      M(3,2)=M(3,2)+4._RK/detF*(
     &   dw2(k)*(A(k,3)-I4bar(k)/3._RK)*(A(k,2)-I4bar(k)/3._RK)
     & + dw1(k)/3._RK*(I4bar(k)/3._RK-A(k,3)-A(k,2)))
!-----
      M(4,1)=M(4,1)+2._RK/detF*(
     &   2._RK*dw2(k)*(A(k,1)-I4bar(k)/3._RK)*A(k,4)
     & + dw1(k)/3._RK*A(k,4))
      M(6,1)=M(6,1)+2._RK/detF*(
     &   2._RK*dw2(k)*(A(k,1)-I4bar(k)/3._RK)*A(k,6)
     & + dw1(k)/3._RK*A(k,6))
      M(4,2)=M(4,2)+2._RK/detF*(
     &   2._RK*dw2(k)*(A(k,2)-I4bar(k)/3._RK)*A(k,4)
     & + dw1(k)/3._RK*A(k,4))
      M(5,2)=M(5,2)+2._RK/detF*(
     &   2._RK*dw2(k)*(A(k,2)-I4bar(k)/3._RK)*A(k,5)
     & + dw1(k)/3._RK*A(k,5))
      M(5,3)=M(5,3)+2._RK/detF*(
     &   2._RK*dw2(k)*(A(k,3)-I4bar(k)/3._RK)*A(k,5)
     & + dw1(k)/3._RK*A(k,5))
      M(6,3)=M(6,3)+2._RK/detF*(
     &   2._RK*dw2(k)*(A(k,3)-I4bar(k)/3._RK)*A(k,6)
     & + dw1(k)/3._RK*A(k,6))
!-----
      M(5,1)=M(5,1)+4._RK/detF*(dw2(k)*(A(k,1)-I4bar(k)/3._RK)*A(k,5)
     & - dw1(k)/3._RK*A(k,5))
      M(6,2)=M(6,2)+4._RK/detF*(dw2(k)*(A(k,2)-I4bar(k)/3._RK)*A(k,6)
     & - dw1(k)/3._RK*A(k,6))
      M(4,3)=M(4,3)+4._RK/detF*(dw2(k)*(A(k,3)-I4bar(k)/3._RK)*A(k,4)
     & - dw1(k)/3._RK*A(k,4))
!-----
      M(4,4)=M(4,4)+1._RK/detF*(4._RK*dw2(k)*(A(k,4))**2
     & + dw1(k)*(A(k,1)+A(k,2)))
      M(5,5)=M(5,5)+1._RK/detF*(4._RK*dw2(k)*(A(k,5))**2
     & + dw1(k)*(A(k,2)+A(k,3)))
      M(6,6)=M(6,6)+1._RK/detF*(4._RK*dw2(k)*(A(k,6))**2
```

```
     & + dw1(k)*(A(k,1)+A(k,3)))
!-----
      M(5,4)=M(5,4)+1._RK/detF*(4._RK*dw2(k)*A(k,4)*A(k,5)
     & + dw1(k)*A(k,6))
      M(6,4)=M(6,4)+1._RK/detF*(4._RK*dw2(k)*A(k,4)*A(k,6)
     & + dw1(k)*A(k,5))
      M(6,5)=M(6,5)+1._RK/detF*(4._RK*dw2(k)*A(k,5)*A(k,6)
     & + dw1(k)*A(k,4))
      end do
!=======================================================================
      end if
      if(model.eq.3._RK)then
!======================================================================= << model 3
! model 3:       anisotropic hyperelastic two fibre families model
! author:        Nicolas Mesnier
! creation date: 03/05/2010
! last modified: 20/07/2010
!-----------------------------------------------------------------------
!  The behaviour is described by the holzapfel hyperelastic anisotropic
!  compressible law [3] with two additional fibre family [2]
!       W = Wiso + Wvol + Waniso
!  where each contribution is defined as:
!       Wiso = mu /2 * ( I1 - 3 )
!       Wvol = KK /2 * ( J - 1 ) ^2
!       Waniso = k1/k2*{ exp[ k2 *{(1-rho)*(I1-3)^2+rho*(I4-1)^2}] -1}
!  with the two fiber family:
!       fib1 = cos(beta) Et + sin(beta) Ez
!       fib2 = cos(beta) Et - sin(beta) Ez
!  Input material properties:
!       mu   : shear modulus
!       k    : bulk modulus
!       k1   : stress-like
!       k2   : dimensionless
!       rho  : dimensionless, 0 < rho < 1
!       beta : angle in degrees, 0° < beta < 90°
!  References :
!     [3] Holzapfel et al, AJP 289 (2005)
!=======================================================================
!-----------------------------------------------------------------------
! *** input material parameters
!-----------------------------------------------------------------------
      mu    = prop(2)           ! shear modulus
      bm    = prop(3)           ! bulk modulus
      K1(1) = prop(4)           ! aniso parameter (stress-like)
      K2(1) = prop(5)           ! aniso parameter (dimensionless)
      rho   = prop(6)           ! aniso proportion (dimensionless)
      beta  = prop(7)*pi/180._RK ! fiber angle (radians)
!-----------------------------------------------------------------------
! *** material anisotropy
!-----------------------------------------------------------------------
! init fibers
      fib(:,:) = 0._RK
! crossed fiber
      fib(1,1:3) = (/ElemOrientVec(1)*cos(beta),
     &               ElemOrientVec(2)*cos(beta),
     &               sin(beta)/)
      fib(2,1:3) = (/ElemOrientVec(1)*cos(beta),
     &               ElemOrientVec(2)*cos(beta),
     &               -sin(beta)/)
! spatial fibers
      do k=1,2
        spafib(k,:)= (/
     &          F1(1,1)*fib(k,1)+F1(1,2)*fib(k,2)+F1(1,3)*fib(k,3),
     &          F1(2,1)*fib(k,1)+F1(2,2)*fib(k,2)+F1(2,3)*fib(k,3),
     &          F1(3,1)*fib(k,1)+F1(3,2)*fib(k,2)+F1(3,3)*fib(k,3)/)
      end do
! spatial anisotropic tensors
      do k=1,2
        A(k,:) = (/ detF23*spafib(k,1)*spafib(k,1),
     &              detF23*spafib(k,2)*spafib(k,2),
     &              detF23*spafib(k,3)*spafib(k,3),
     &              detF23*spafib(k,2)*spafib(k,1),
     &              detF23*spafib(k,3)*spafib(k,2),
     &              detF23*spafib(k,3)*spafib(k,1)/)
```

```
      end do
      A(3,1:6) = (A(1,1:6)+A(2,1:6))/2._RK
      A(1,1:6) = A(3,1:6)
!-----------------------------------------------------------------------
! *** invariants
!-----------------------------------------------------------------------
! 1st invariant
      I1bar = (Bbar(1) + Bbar(2) + Bbar(3))
      I1bar3 = I1bar/3._RK      ! I1bar / 3
! 4th invariants
      I4bar(1) = A(1,1)+A(1,2)+A(1,3)
!-----------------------------------------------------------------------
! *** anisotropic sedf derivatives
!-----------------------------------------------------------------------
! fiber exponential factor
      expf(1)  = 4._RK*K1(1)*
     &           exp(K2(1)*((1._RK-rho)*(I1bar-3._RK)**2
     &           +rho*(I4bar(1)-1._RK)**2))
! fiber factor for second derivatives
      dw2f(1) = 1._RK+2._RK* K2(1)*(1._RK-rho)*(I1bar-3._RK)**2
      dw2f(2) = 1._RK+2._RK* K2(1)*rho*(I4bar(1)-1._RK)**2
! sedf derivative w.r.t. I1bar and I4bar
      dw1(1) = (1._RK-rho)*(I1bar-3._RK)*expf(1)/detF
      dw1(2) = rho*(I4bar(1)-1._RK)*expf(1)/detF
! sedf derivative w.r.t. I1barI1bar, I4barI4bar, I1barI4bar
      dw2(1) = 2._RK*(1._RK-rho)*dw2f(1)*expf(1)/detF
      dw2(2) = 2._RK*rho*dw2f(2)*expf(1)/detF
      dw2(3) = 4._RK*rho*(1._RK-rho)*K2(1)
     &         *(I1bar-3._RK)*(I4bar(1)-1._RK)*expf(1)/detF
!-----------------------------------------------------------------------
! *** update sedf
!-----------------------------------------------------------------------
      sedEl = mu*(I1bar-3._RK) + bm*(detF-1._RK)**2
     &      + 2._RK*K1(1)/K2(1)*
     &      (exp(K2(1)*((1._RK-rho)*(I1bar-3._RK)**2
     &      +rho*(I4bar(1)-1._RK)**2))-1._RK)
      sedEl = sedEl/2._RK /detF
      sedPl = 0._RK
!-----------------------------------------------------------------------
! *** cauchy stress in spatial frame
!-----------------------------------------------------------------------
      sigma(1:3) = (mu/detF+dw1(1)) *(Bbar(1:3)-I1bar3)
     &             + dw1(2)*(A(1,1:3)-I4bar(1)/3._RK)
     &             + bm*(detF-1._RK)
      sigma(4:6) = (mu/detF+dw1(1)) *(Bbar(4:6))
     &             + dw1(2)*(A(1,4:6))
!-----------------------------------------------------------------------
! *** Jaumann tangent stiffness in spatial frame
!-----------------------------------------------------------------------
! factors
      C1 = 2._RK/3._RK*mu/detF
      C2 = 3._RK/4._RK*C1
! isotropic deviatoric components
!-----------------------------------------------------------------------
      M(1,1) = C1 * (Bbar(1) + I1bar3)
      M(2,2) = C1 * (Bbar(2) + I1bar3)
      M(3,3) = C1 * (Bbar(3) + I1bar3)
!-----
      M(2,1) =-C1 * (Bbar(1) + Bbar(2) - I1bar3)
      M(3,1) =-C1 * (Bbar(1) + Bbar(3) - I1bar3)
      M(3,2) =-C1 * (Bbar(2) + Bbar(3) - I1bar3)
!-----
      M(4,1) = C1 * Bbar(4) / 2._RK
      M(6,1) = C1 * Bbar(6) / 2._RK
      M(4,2) = C1 * Bbar(4) / 2._RK
      M(5,2) = C1 * Bbar(5) / 2._RK
      M(5,3) = C1 * Bbar(5) / 2._RK
      M(6,3) = C1 * Bbar(6) / 2._RK
!-----
      M(5,1) =-C1 * Bbar(5)
      M(6,2) =-C1 * Bbar(6)
      M(4,3) =-C1 * Bbar(4)
!-----
      M(4,4) = C2 * (Bbar(1) + Bbar(2))
```

```
      M(5,5) = C2 * (Bbar(2) + Bbar(3))
      M(6,6) = C2 * (Bbar(1) + Bbar(3))
!-----
      M(5,4) = C2 * Bbar(6)
      M(6,4) = C2 * Bbar(5)
      M(6,5) = C2 * Bbar(4)
! anisotropic deviatoric components
!-------------------------------------------------------------------------
      M(1,1)=M(1,1) + dw2(1)*(Bbar(1)-I1bar3)**2
     &              + dw2(2)*(A(1,1)-I4bar(1)/3._RK)**2
     & +2._RK*dw2(3)*(Bbar(1)-I1bar3)*(A(1,1)-I4bar(1)/3._RK)
     & +4._RK/3._RK*(dw1(1)*(2._RK*I1bar3-Bbar(1))
     &              +dw1(2)*(2._RK*I4bar(1)/3._RK-A(1,1)))
      M(2,2)=M(2,2) + dw2(1)*(Bbar(2)-I1bar3)**2
     &              + dw2(2)*(A(1,2)-I4bar(1)/3._RK)**2
     & +2._RK*dw2(3)*(Bbar(2)-I1bar3)*(A(1,2)-I4bar(1)/3._RK)
     & +4._RK/3._RK*(dw1(1)*(2._RK*I1bar3-Bbar(2))
     &              +dw1(2)*(2._RK*I4bar(1)/3._RK-A(1,2)))
      M(3,3)=M(3,3) + dw2(1)*(Bbar(3)-I1bar3)**2
     &              + dw2(2)*(A(1,3)-I4bar(1)/3._RK)**2
     & +2._RK*dw2(3)*(Bbar(3)-I1bar3)*(A(1,3)-I4bar(1)/3._RK)
     & +4._RK/3._RK*(dw1(1)*(2._RK*I1bar3-Bbar(3))
     &              +dw1(2)*(2._RK*I4bar(1)/3._RK-A(1,3)))
!-----
      M(2,1)=M(2,1) + dw2(1)*(Bbar(1)-I1bar3)*(Bbar(2)-I1bar3)
     &      + dw2(2)*(A(1,1)-I4bar(1)/3._RK)*(A(1,2)-I4bar(1)/3._RK)
     &      + dw2(3)*(Bbar(1)-I1bar3)*(A(1,2)-I4bar(1)/3._RK)
     &      + dw2(3)*(Bbar(2)-I1bar3)*(A(1,1)-I4bar(1)/3._RK)
     &      +2._RK/3._RK*dw1(1)*(I1bar3-Bbar(1)-Bbar(2))
     &      +2._RK/3._RK*dw1(2)*(I4bar(1)/3._RK-A(1,1)-A(1,2))
      M(3,1)=M(3,1) + dw2(1)*(Bbar(1)-I1bar3)*(Bbar(3)-I1bar3)
     &      + dw2(2)*(A(1,1)-I4bar(1)/3._RK)*(A(1,3)-I4bar(1)/3._RK)
     &      + dw2(3)*(Bbar(1)-I1bar3)*(A(1,3)-I4bar(1)/3._RK)
     &      + dw2(3)*(Bbar(3)-I1bar3)*(A(1,1)-I4bar(1)/3._RK)
     &      +2._RK/3._RK*dw1(1)*(I1bar3-Bbar(1)-Bbar(3))
     &      +2._RK/3._RK*dw1(2)*(I4bar(1)/3._RK-A(1,1)-A(1,3))
      M(3,2)=M(3,2) + dw2(1)*(Bbar(2)-I1bar3)*(Bbar(3)-I1bar3)
     &      + dw2(2)*(A(1,2)-I4bar(1)/3._RK)*(A(1,3)-I4bar(1)/3._RK)
     &      + dw2(3)*(Bbar(2)-I1bar3)*(A(1,3)-I4bar(1)/3._RK)
     &      + dw2(3)*(Bbar(3)-I1bar3)*(A(1,2)-I4bar(1)/3._RK)
     &      +2._RK/3._RK*dw1(1)*(I1bar3-Bbar(2)-Bbar(3))
     &      +2._RK/3._RK*dw1(2)*(I4bar(1)/3._RK-A(1,2)-A(1,3))
!-----
      M(4,1)=M(4,1) + dw2(1)*(Bbar(1)-I1bar3)*Bbar(4)
     &      + dw2(2)*(A(1,1)-I4bar(1)/3._RK)*A(1,4)
     &      + dw2(3)*(Bbar(1)-I1bar3)*A(1,4)
     &      + dw2(3)*(A(1,1)-I4bar(1)/3._RK)*Bbar(4)
     &      -2._RK/3._RK*(dw1(1)*Bbar(4)+dw1(2)*A(1,4))
      M(6,1)=M(6,1) + dw2(1)*(Bbar(1)-I1bar3)*Bbar(6)
     &      + dw2(2)*(A(1,1)-I4bar(1)/3._RK)*A(1,6)
     &      + dw2(3)*(Bbar(1)-I1bar3)*A(1,6)
     &      + dw2(3)*(A(1,1)-I4bar(1)/3._RK)*Bbar(6)
     &      -2._RK/3._RK*(dw1(1)*Bbar(6)+dw1(2)*A(1,6))
      M(4,2)=M(4,2) + dw2(1)*(Bbar(2)-I1bar3)*Bbar(4)
     &      + dw2(2)*(A(1,2)-I4bar(1)/3._RK)*A(1,4)
     &      + dw2(3)*(Bbar(2)-I1bar3)*A(1,4)
     &      + dw2(3)*(A(1,2)-I4bar(1)/3._RK)*Bbar(4)
     &      -2._RK/3._RK*(dw1(1)*Bbar(4)+dw1(2)*A(1,4))
      M(5,2)=M(5,2) + dw2(1)*(Bbar(2)-I1bar3)*Bbar(5)
     &      + dw2(2)*(A(1,2)-I4bar(1)/3._RK)*A(1,5)
     &      + dw2(3)*(Bbar(2)-I1bar3)*A(1,5)
     &      + dw2(3)*(A(1,2)-I4bar(1)/3._RK)*Bbar(5)
     &      -2._RK/3._RK*(dw1(1)*Bbar(5)+dw1(2)*A(1,5))
      M(5,3)=M(5,3) + dw2(1)*(Bbar(3)-I1bar3)*Bbar(5)
     &      + dw2(2)*(A(1,3)-I4bar(1)/3._RK)*A(1,5)
     &      + dw2(3)*(Bbar(3)-I1bar3)*A(1,5)
     &      + dw2(3)*(A(1,3)-I4bar(1)/3._RK)*Bbar(5)
     &      -2._RK/3._RK*(dw1(1)*Bbar(5)+dw1(2)*A(1,5))
      M(6,3)=M(6,3) + dw2(1)*(Bbar(3)-I1bar3)*Bbar(6)
     &      + dw2(2)*(A(1,3)-I4bar(1)/3._RK)*A(1,6)
     &      + dw2(3)*(Bbar(3)-I1bar3)*A(1,6)
     &      + dw2(3)*(A(1,3)-I4bar(1)/3._RK)*Bbar(6)
     &      -2._RK/3._RK*(dw1(1)*Bbar(6)+dw1(2)*A(1,6))
!-----
```

```
      M(5,1)=M(5,1) + dw2(1)*(Bbar(1)-I1bar3)*Bbar(5)
     &      + dw2(2)*(A(1,1)-I4bar(1)/3._RK)*A(1,5)
     &      + dw2(3)*(Bbar(1)-I1bar3)*A(1,5)
     &      + dw2(3)*(A(1,1)-I4bar(1)/3._RK)*Bbar(5)
     &      -2._RK/3._RK*(dw1(1)*Bbar(5)+dw1(2)*A(1,5))
      M(6,2)=M(6,2) + dw2(1)*(Bbar(2)-I1bar3)*Bbar(6)
     &      + dw2(2)*(A(1,2)-I4bar(1)/3._RK)*A(1,6)
     &      + dw2(3)*(Bbar(2)-I1bar3)*A(1,6)
     &      + dw2(3)*(A(1,2)-I4bar(1)/3._RK)*Bbar(6)
     &      -2._RK/3._RK*(dw1(1)*Bbar(6)+dw1(2)*A(1,6))
      M(4,3)=M(4,3) + dw2(1)*(Bbar(3)-I1bar3)*Bbar(4)
     &      + dw2(2)*(A(1,3)-I4bar(1)/3._RK)*A(1,4)
     &      + dw2(3)*(Bbar(3)-I1bar3)*A(1,4)
     &      + dw2(3)*(A(1,3)-I4bar(1)/3._RK)*Bbar(4)
     &      -2._RK/3._RK*(dw1(1)*Bbar(4)+dw1(2)*A(1,4))
!-----
      M(4,4)=M(4,4) + dw2(1)*(Bbar(4))**2
     &      + dw2(2)*(A(1,4))**2
     &      + 2._RK*dw2(3)*Bbar(4)*A(1,4)
     &      +1._RK/3._RK*(dw1(1)*I1bar+dw1(2)*I4bar(1))
      M(5,5)=M(5,5) + dw2(1)*(Bbar(5))**2
     &      + dw2(2)*(A(1,5))**2
     &      + 2._RK*dw2(3)*Bbar(5)*A(1,5)
     &      +1._RK/3._RK*(dw1(1)*I1bar+dw1(2)*I4bar(1))
      M(6,6)=M(6,6) + dw2(1)*(Bbar(6))**2
     &      + dw2(2)*(A(1,6))**2
     &      + 2._RK*dw2(3)*Bbar(6)*A(1,6)
     &      +1._RK/3._RK*(dw1(1)*I1bar+dw1(2)*I4bar(1))
!-----
      M(5,4)=M(5,4) + dw2(1)*Bbar(4)*Bbar(5)
     &      + dw2(2)*A(1,4)*A(1,5)
     &      + dw2(3)*(Bbar(4)*A(1,5)+Bbar(5)*A(1,4))
      M(6,4)=M(6,4) + dw2(1)*Bbar(4)*Bbar(6)
     &      + dw2(2)*A(1,4)*A(1,6)
     &      + dw2(3)*(Bbar(4)*A(1,6)+Bbar(6)*A(1,4))
      M(6,5)=M(6,5) + dw2(1)*Bbar(5)*Bbar(6)
     &      + dw2(2)*A(1,5)*A(1,6)
     &      + dw2(3)*(Bbar(5)*A(1,6)+Bbar(6)*A(1,5))
!**********************************************************************
      endif
!----------------------------------------------------------------------
! symmetries
!----------------------------------------------------------------------
      M(1,2) = M(2,1)
      M(1,3) = M(3,1)
      M(2,3) = M(3,2)
      M(1,4) = M(4,1)
      M(2,4) = M(4,2)
      M(3,4) = M(4,3)
      M(1,5) = M(5,1)
      M(2,5) = M(5,2)
      M(3,5) = M(5,3)
      M(4,5) = M(5,4)
      M(1,6) = M(6,1)
      M(2,6) = M(6,2)
      M(3,6) = M(6,3)
      M(4,6) = M(6,4)
      M(5,6) = M(6,5)
!----------------------------------------------------------------------
! isotropic volumic components
!----------------------------------------------------------------------
      M(1:3,1:3) = M(1:3,1:3) + bm*(2._RK*detF - 1._RK)
!----------------------------------------------------------------------
! change from spatial to corotated frame
!----------------------------------------------------------------------
! rotation from polar decomposition
      call polarRU(F1,R,U)
! dim 6 rotation matrix
      call trans_matrx_6(Q,transpose(R))
! corotated stress
      stressfull(:) = matmul(Q,sigma)
! corotated tangent
      dsdePlfull(:,:) = matmul(Q,matmul(M,transpose(Q)))
      if (ncomp .eq. 4) then
```

```
        stress(:) = stressfull(1:4)
        dsdePl = dsdePlfull(1:4,1:4)
      else
        stress = stressfull
        dsdePl = dsdePlfull
      end if
! transverse shear stiffness
      tsstif(:) = dsdePlfull(6,6)
!-----------------------------------------------------------------------
! state variables
!-----------------------------------------------------------------------
! *** orientation vector E_theta
!     ustatev(1) = EVT(1)
!     ustatev(2) = EVT(2)
!-----------------------------------------------------------------------
      normspaevt = sqrt( (F1(1,1)*EVT(1)+F1(1,2)*EVT(2))**2
     &                 + (F1(2,1)*EVT(1)+F1(2,2)*EVT(2))**2 )
      spaevt(1) = (F1(1,1)*EVT(1)+F1(1,2)*EVT(2))/normspaevt
      spaevt(2) = (F1(2,1)*EVT(1)+F1(2,2)*EVT(2))/normspaevt
! *** radial stress sigma_rr
      ustatev(3) = sigma(1)*(spaevt(2))**2
     &           + sigma(2)*(spaevt(1))**2
     &           - 2._RK * sigma(4)*spaevt(1)*spaevt(2)
! *** circonferential stress sigma_tt
      ustatev(4) = sigma(1)*(spaevt(1))**2
     &           + sigma(2)*(spaevt(2))**2
     &           + 2._RK * sigma(4)*spaevt(1)*spaevt(2)
!-----------------------------------------------------------------------
      end subroutine usermat3d
!-----------------------------------------------------------------------
!***********************************************************************
! *** subroutines ***
! author:               JM GeRKen, ANSYS, Inc.
! creation date:        17/12/2008
!***********************************************************************
!-----------------------------------------------------------------------
      subroutine polarRU(F,R,U)
!-----------------------------------------------------------------------
! *** Polar decomposition, F=RU
!  References :
!    JC Simo & TJR Hughes, Computational Inelasticity, 1998, Springer
!-----------------------------------------------------------------------
      IMPLICIT NONE
      integer,parameter :: RK=KIND(1.D0) ! real kind
! arguments
      real(RK),intent(IN) :: F(3,3) ! matrix to decompose
      real(RK),intent(OUT) :: R(3,3) !rotation
      real(RK),intent(OUT) :: U(3,3) !stretch
! local
      real(RK), parameter :: pi   =  3.14159265358979324_RK
      real(RK),parameter ::I(3,3)=reshape((/1._RK,0._RK,0._RK,
     &                                      0._RK,1._RK,0._RK,
     &                                      0._RK,0._RK,1._RK/),(/3,3/))
      real(RK) :: C(3,3),CC(3,3) ! right Cauchy-Green, squared
      real(RK) :: Ui(3,3) ! U inverse
      real(RK) :: IC, IIC, IIIC ! invariants of C
      real(RK) :: IU, IIU, IIIU ! invariants of U
      real(RK) :: l1,l2,l3 ! eigenvalues
      real(RK) :: p,q,m,n,t,D ! constants
!-----------------------------------------------------------------------
! right Cauchy-Green
      C(:,:) = matmul(transpose(F),F)
      CC(:,:) = matmul(C,C)
! invariants
      IC = C(1,1) + C(2,2) + C(3,3)
      IIC = 0.5_RK*(IC**2  - (CC(1,1) + CC(2,2) + CC(3,3)))
      IIIC = C(1,1) * (C(2,2)*C(3,3) - C(2,3)*C(3,2))
     &     + C(1,2) * (C(2,3)*C(3,1) - C(2,1)*C(3,3))
     &     + C(1,3) * (C(2,1)*C(3,2) - C(2,2)*C(3,1))
! eigenvalues of sqrt(C)
      p = IIC - (IC**2)/3._RK
      q = -(2._RK/27._RK)*IC**3+IC*IIC/3._RK-IIIC
      if(abs(p)<epsilon(1._RK))then
        l1 = sqrt( abs(-abs(q)**(1._RK/3._RK) + IC/3._RK) )
```

```
        l2 = l1
        l3 = l2
      else
        m = 2._RK*sqrt(abs(p)/3._RK)
        n = 3._RK*q/(m*p)
        if(abs(n)>1._RK) n = sign(1._RK, n)
        t = atan2(sqrt(1._RK-n**2),n)/3._RK
        l1 = sqrt( abs(m*cos(t) + IC/3._RK) )
        l2 = sqrt( abs(m*cos(t+2._RK/3._RK*pi) + IC/3._RK) )
        l3 = sqrt( abs(m*cos(t+4._RK/3._RK*pi) + IC/3._RK) )
      endif
! stretch and inverse
       !invariants
       IU = l1 + l2 + l3
       IIU = l1*l2 + l1*l3 + l2*l3
       IIIU = l1*l2*l3
       D = IU*IIU-IIIU

       U(:,:)  = (-CC(:,:) + (IU**2-IIU)*C(:,:) + IU*IIIU*I(:,:))/D
       Ui(:,:) = (C(:,:) - IU*U(:,:) + IIU*I(:,:))/IIIU
! rotation
      R(:,:) = matmul(F,Ui)
      end subroutine polarRU
!-----------------------------------------------------------------------
      SUBROUTINE trans_matrx_6(Q,R)
!-----------------------------------------------------------------------
! construct the 6x6 Voigt rotation matrix
!-----------------------------------------------------------------------
      IMPLICIT NONE
      integer,parameter :: RK=KIND(1.D0) ! real kind
! arguments
      real(RK), intent(OUT) :: Q(6,6)
      real(RK), intent(IN) :: R(3,3)
!-----------------------------------------------------------------------
      Q(1,1) = R(1,1)**2
      Q(2,1) = R(2,1)**2
      Q(3,1) = R(3,1)**2
      Q(4,1) = R(1,1)*R(2,1)
      Q(5,1) = R(2,1)*R(3,1)
      Q(6,1) = R(3,1)*R(1,1)

      Q(1,2) = R(1,2)**2
      Q(2,2) = R(2,2)**2
      Q(3,2) = R(3,2)**2
      Q(4,2) = R(1,2)*R(2,2)
      Q(5,2) = R(2,2)*R(3,2)
      Q(6,2) = R(3,2)*R(1,2)

      Q(1,3) = R(1,3)**2
      Q(2,3) = R(2,3)**2
      Q(3,3) = R(3,3)**2
      Q(4,3) = R(1,3)*R(2,3)
      Q(5,3) = R(2,3)*R(3,3)
      Q(6,3) = R(3,3)*R(1,3)

      Q(1,4) = R(1,1)*R(1,2)*2._RK
      Q(2,4) = R(2,1)*R(2,2)*2._RK
      Q(3,4) = R(3,1)*R(3,2)*2._RK
      Q(4,4) = R(1,1)*R(2,2)+R(1,2)*R(2,1)
      Q(5,4) = R(2,1)*R(3,2)+R(2,2)*R(3,1)
      Q(6,4) = R(3,1)*R(1,2)+R(3,2)*R(1,1)

      Q(1,5) = R(1,2)*R(1,3)*2._RK
      Q(2,5) = R(2,2)*R(2,3)*2._RK
      Q(3,5) = R(3,2)*R(3,3)*2._RK
      Q(4,5) = R(1,2)*R(2,3)+R(1,3)*R(2,2)
      Q(5,5) = R(2,2)*R(3,3)+R(2,3)*R(3,2)
      Q(6,5) = R(3,2)*R(1,3)+R(3,3)*R(1,2)

      Q(1,6) = R(1,1)*R(1,3)*2._RK
      Q(2,6) = R(2,1)*R(2,3)*2._RK
      Q(3,6) = R(3,1)*R(3,3)*2._RK
      Q(4,6) = R(1,1)*R(2,3)+R(1,3)*R(2,1)
      Q(5,6) = R(2,1)*R(3,3)+R(2,3)*R(3,1)
```

```
      Q(6,6) = R(3,1)*R(1,3)+R(3,3)*R(1,1)
      END SUBROUTINE trans_matrx_6
!-----------------------------------------------------------------------
```