Published in the International Journal for Engineering with Computers, vol. 15(1), pages 2-11, 1999.

Object Oriented Approach to Hyperelasticity

B. Jeremić [1] **K. Runesson** [2] **S. Sture** [3]

[1]Assistant Professor,
Department of Civil and Environmental Engineering,
Clarkson University, Potsdam, NY, 13699, U.S.A.
phone (315) 268-4435, fax (315) 268-7985
Email: `Jeremic@Polaris.Clarkson.edu`

[2]Professor,
Division of Solid Mechanics,
Chalmers University of Technology, S-41296 Göteborg, Sweden
Email: `KeRu@Solid.Chalmers.SE`

[3]Professor,
Department of Civil, Environmental, and Architectural Engineering,
University of Colorado, Boulder, CO 80309-0428, U.S.A.
Email: `Stein.Sture@Civil.Colorado.edu`

Abstract

This paper describes the application of an Object Oriented Paradigm (OOP) to the implementation of a hyperelastic constitutive driver. The C++ programming language used in our implementation leads to an efficient and readable program. It will be shown that Object Oriented implementation naturally follows from analytical developments in isotropic hyperelasticity. Examples of developed classes and results from a number of large deformation hyperelastic numerical test will be presented.

Key Words: Hyperelasticity, Object Oriented Programming, Tensor Analysis, Constitutive Driver, Finite Element Programming.

1 Introduction

A traditional approach to the development and implementation of an algorithm in computational mechanics is to use tensorial notations for the theoretical development and then transform developed formulae to matrix and vector notation for implementation in FORTRAN or C programming languages. Needless to say, considerable amount of time is often devoted to the actual implementation process. In this paper we show how OOP can help alleviate painful implementation issues and yet produce an efficient, self–explainable code. A set of class libraries called **nDarray** (c.f. [7]) are used to build an object oriented framework for computations in large deformation hyperelasticity.

For readers who wish to explore the subject of Object Oriented Programming and implementation strategies in greater depth we suggest the following references [2], [17], [3] [4], C++ draft standard [1] and journal articles [8], [19].

2 Hyperelasticity

In this section we present the underlying theory and derivation of isotropic hyperelastic material models. We first start with an overview of hyperelasticity, then focus our attention on isotropic behavior. Then, we describe the volumetric–deviatoric decomposition and present the Simo–Serrin formulation. Lagrangian and Eulerian stress measures and tangent stiffness operators are developed. Finally a number of commonly used isotropic hyperelastic material models are presented as a specializations of general derivations.

2.1 Background

A material is called *hyperelastic* or *Green elastic*, if there exists an *elastic potential function* W, also called the *strain energy function per unit volume of the undeformed configuration*, which represents a scalar function of strain or deformation tensors, whose derivatives with respect to a strain component determines the corresponding stress component. The most general form of the elastic potential function with restriction to pure mechanical theory, by using the *axiom of locality* and the *axiom of entropy production*[1] can be presented as:

$$W = W\left(X_K, F_{kK}\right) \tag{1}$$

where $F_{kK} = \partial x_k / \partial X_K = x_{k,K}$ is a deformation gradient. By using the *axiom of material frame indifference*, we conclude that W depends only on X_K and C_{IJ}, that is:

$$W = W\left(X_K, C_{IJ}\right) \quad \text{or:} \quad W = W\left(X_K, c_{ij}\right) \tag{2}$$

Left and right Cauchy–Green tensors are defined as:

$$C_{IJ} = (F_{kI})^t F_{kJ} \quad ; \quad (c^{-1})_{ij} = b_{ij} = F_{iK}(F_{jK})^t \tag{3}$$

By assuming hyperelastic response, the following are the constitutive equations for the material stress tensors:

- 2. Piola–Kirchhoff stress tensor $S_{IJ} = 2\partial W / \partial C_{IJ}$
- Mandel stress tensor $T_{IJ} = 2C_{IK} \partial W / \partial C_{KJ}$

[1]See Marsden and Hughes [9] pp. 190.

- 1. Piola–Kirchhoff stress tensor $P_{iJ} = 2\partial W/\partial C_{IJ}(F_{iI})^t$

and the spatial, Kirchhoff stress tensor is defined as:

- Kirchhoff stress tensor $\tau_{ij} = 2\ F_{iA}(F_{jB})^t \partial W/\partial C_{AB}$

The material tangent stiffness relation is defined as:

$$dS_{IJ} = \frac{1}{2}\,\mathcal{L}_{IJKL}\,dC_{KL} \quad \text{where} \quad \mathcal{L}_{IJKL} = 4\,\frac{\partial^2 W}{\partial C_{IJ}\,\partial C_{KL}} \tag{4}$$

The spatial tangent stiffness tensor $\mathcal{E}_{ijkl}$ is obtained by the following *push–forward* operation with the deformation gradient:

$$\mathcal{E}_{ijkl} = F_{iI}F_{jJ}(F_{kK})^t(F_{lL})^t\mathcal{L}_{IJKL} \tag{5}$$

2.2 Isotropic Hyperelasticity

In the case of material isotropy, the strain energy function $W\,(X_K, C_{IJ})$ belongs to the class of isotropic, invariant scalar functions. It satisfies the relation:

$$W\,(X_K, C_{KL}) = W\left(X_K, Q_{KI}C_{IJ}\,(Q_{JL})^t\right) \tag{6}$$

where Q_{KI} is the proper orthogonal transformation. The polar decomposition theorem permits the unique representation[2]:

$$F_{ij} = R_{ik}U_{kj} = v_{ik}R_{kj} \tag{7}$$

where U_{kj}, v_{ik} are positive definite symmetric tensors, called *right stretch tensors* and *left stretch tensors*, respectively, and R_{kj} is an *orthogonal tensor* such that:

$$R_{ik}R_{jk} = \delta_{ij} \qquad \text{and} \qquad R_{ki}R_{kj} = \delta_{ij} \tag{8}$$

[2]referring x_i and X_i to the same reference axes and using lower case indices for both.

If we choose $Q_{KI} = R_{KI}$, then:

$$W\left(X_K, C_{KL}\right) = W\left(X_K, U_{KL}\right) = W\left(X_K, v_{kl}\right) \tag{9}$$

The right and left stretch tensors, U_{KL}, v_{kl} have the same principal values[3] $\lambda_i \;\; ; \;\; i = \overline{1,3}$ so the strain energy function W can be represented in terms of principal stretches, or similarly in terms of principal invariants of the deformation tensor:

$$W = W\left(X_K, \lambda_1, \lambda_2, \lambda_3, \right) = W\left(X_K, I_1, I_2, I_3\right) \tag{10}$$

where:

$$\begin{aligned} I_1 &\overset{\text{def}}{=} C_{II} \\ I_2 &\overset{\text{def}}{=} \frac{1}{2}\left(I_1^2 - C_{IJ}C_{JI}\right) \\ I_3 &\overset{\text{def}}{=} \det\left(C_{IJ}\right) = \frac{1}{6} e_{IJK} e_{PQR} C_{IP} C_{JQ} C_{KR} = J^2 \end{aligned} \tag{11}$$

and e_{IJK} is the Levi–Civita permutation tensor.

The spectral[4] decomposition theorem for symmetric positive definite tensors[5] states that:

$$C_{IJ} = \lambda_A^2 \left(N_I^{(A)} N_J^{(A)}\right)_A \qquad \text{where} \qquad A = \overline{1,3} \tag{12}$$

and N_I are the eigenvectors (so that $\|N_I\| = 1$) of C_{IJ}. Values λ_A^2 are the roots of the characteristic polynomial

$$P(\lambda_A^2) \overset{\text{def}}{=} -\lambda_A^6 + I_1\, \lambda_A^6 - I_2\, \lambda_A^4 + I_3 = 0 \tag{13}$$

It should be noted that no summation is implied over indices in parenthesis. For example, in the present case $N_I^{(A)}$ is the Ath eigenvector with members $N_1^{(A)}$, $N_2^{(A)}$ and $N_3^{(A)}$, so that the actual equation $C_{IJ} = \lambda_A^2 \left(N_I^{(A)} N_J^{(A)}\right)_A$ can also be written as

[3]Principal stretches.

[4]We follow, to some extent, developments described by Simo and Taylor [16].

[5]Cauchy–Green tensor C_{IJ} for example.

$C_{IJ} = \sum_{A=1}^{A=3} \lambda_{(A)}^2 N_I^{(A)} N_J^{(A)}$. In order to follow the consistency of indicial notation in this work, we shall make an effort to represent all the tensorial equations in indicial form.

The mapping of the eigenvectors can be deduced from equation $dx_k = F_{kK} dX_K = \partial x_k / \partial X_K dX_K = x_{k,K} dX_K$ and is given by

$$\lambda_{(A)}\, n_i^{(A)} = F_{iJ}\, N_J^{(A)} \tag{14}$$

where $\|n_i^{(A)}\| \equiv 1$. The spectral decomposition of F_{iJ}, R_{iJ} and b_{ij} is then given by

$$F_{iJ} = \lambda_A \left(n_i^{(A)} N_J^{(A)} \right)_A \tag{15}$$

$$R_{iJ} = \sum_{A=1}^{3} n_i^{(A)} N_J^{(A)} \tag{16}$$

$$b_{ij} = \lambda_A^2 \left(n_i^{(A)} n_j^{(A)} \right)_A \tag{17}$$

Spectral decomposition from equation (12) is valid for the case of non–equal principal stretches, i.e. $\lambda_1 \neq \lambda_2 \neq \lambda_3$. If two or all three principal stretches are equal, we shall introduce a small perturbation to the numerical values for principal stretches in order to make them distinct. The case of two or all three values of principal stretches being equal is theoretically possible and may result for example from standard axisymmetric triaxial or isotropic compression experiments. However, we are never certain about equivalence of two real numbers, because of the finite precision arithmetics involved in calculation of these numbers. From a numerical point of view, two numbers are equal, if the difference between them is smaller than the machine precision (*macheps*) specific to the computer platform on which computations are performed. Our perturbation will be a function of the *macheps*.

Recently, Ting [18] and Morman [11] have used Serrin's (c.f. [14]) representation theorem in order to devise a useful representation for generalized strain tensors E_{IJ} and e_{ij} through C_{IJ}^m and b_{ij}^m. General strain tensors can be defined by considering a *scale function* (c.f. Hill [5]) for the stretch. A scale function is any smooth, monotonic function of stretch $f(\lambda)$ such that:

$$f(\lambda) \;\; ; \;\; \lambda \in [0, \infty) \;\; \text{subject to} \;\; f(1) = 0, f'(1) = 1 \tag{18}$$

The scale function is often taken in the form $(\lambda^{2m}-1)/2m$, where m may have any value. If we choose m to be an integer, the corresponding strain tensor is:

$$E_{IJ} = \frac{(U_{IJ}^{2m} - \delta_{IJ})}{2m} \qquad \text{and} \qquad e_{ij} = \frac{\left(\delta_{ij} - v_{ij}^{2m}\right)}{2m} \tag{19}$$

in Lagrangian and Eulerian settings, respectively. Morman [11] has shown that b_{ij}^{m} can be stated as

$$b_{ij}^{m} = \lambda_A^{2m} \left(\frac{(b^2)_{ij} - \left(\left(I_1 - \lambda_{(A)}^2\right) b_{ij} + I_3 \lambda_{(A)}^{-2} \delta_{ij} \right)}{2\lambda_{(A)}^4 - I_1 \lambda_{(A)}^2 + I_3 \lambda_{(A)}^{-2}} \right)_A \tag{20}$$

By comparing equations (20) and (17) it follows that the Eulerian eigen–triad $n_i^{(A)} n_j^{(A)}$ can be written as

$$n_i^{(A)} n_j^{(A)} = \frac{(b^2)_{ij} - \left(I_1 - \lambda_{(A)}^2\right) b_{ij} + I_3 \lambda_{(A)}^{-2} \delta_{ij}}{2\lambda_{(A)}^4 - I_1 \lambda_{(A)}^2 + I_3 \lambda_{(A)}^{-2}} \tag{21}$$

The Lagrangian eigendyad $N_I^{(A)} N_J^{(A)}$, from equation (12), can be derived, if one substitutes the mapping of the eigenvectors, (14), into equation (21) to get:

$$N_I^{(A)} N_J^{(A)} = \lambda_{(A)}^2 \frac{C_{IJ} - \left(I_1 - \lambda_{(A)}^2\right) \delta_{IJ} + I_3 \lambda_{(A)}^{-2} (C^{-1})_{IJ}}{2\lambda_{(A)}^4 - I_1 \lambda_{(A)}^2 + I_3 \lambda_{(A)}^{-2}} \tag{22}$$

where it was used that:

$$C_{IJ} = (F_{iI})^{-1} (b^2)_{ij} (F_{jJ})^{-t} \ \ ; \ \ \delta_{IJ} = (F_{iI})^{-1} b_{ij} (F_{jJ})^{-t} \ \ ; \ \ (C^{-1})_{IJ} = (F_{iI})^{-1} \delta_{ij} (F_{jJ})^{-t}$$

It should be noted that the denominator in equations (21) and (22) can be written as:

$$2\lambda_{(A)}^4 - I_1 \lambda_{(A)}^2 + I_3 \lambda_{(A)}^{-2} = \left(\lambda_{(A)}^2 - \lambda_{(B)}^2\right) \left(\lambda_{(A)}^2 - \lambda_{(C)}^2\right) \overset{\text{def}}{=} D_{(A)} \tag{23}$$

where indices A, B, C are cyclic permutations of $1, 2, 3$. It follows directly from the definition of $D_{(A)}$ in equation (23) that $\lambda_1 \neq \lambda_2 \neq \lambda_3 \Rightarrow D_{(A)} \neq 0$ for equations (21) and (22) to be valid.

2.3 Volumetric–Deviatoric Decomposition of Deformation

It proves useful to separate deformation in volumetric and deviatoric parts by a multiplicative split of a deformation gradient as

$$F_{iI} = \tilde{F}_{i\beta}\,{}^{vol}F_{\beta I} \qquad \text{where} \qquad \tilde{F}_{i\beta} = F_{i\beta}J^{-\frac{1}{3}} \quad ; \quad {}^{vol}F_{\beta I} = J^{\frac{1}{3}}\delta_{\beta I} \tag{24}$$

where x_β represents an intermediate configuration such that deformation $X_I \rightarrow x_\beta$ is purely volumetric and $x_\beta \rightarrow x_i$ is purely deviatoric. It also follows from equation (24) that $\tilde{F}_{\beta I}$ and F_{iI} have the same eigenvectors.

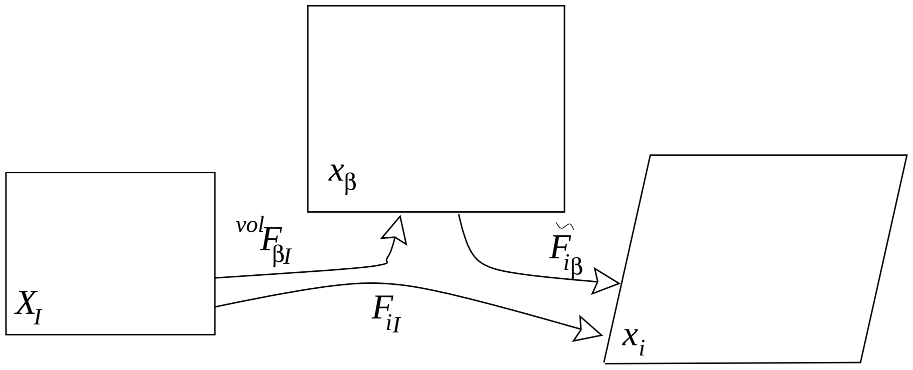

Figure 1: Volumetric deviatoric decomposition of deformation.

The deviatoric part of the Green deformation tensor C_{IJ}, defined in equation (12) can be defined as

$$\tilde{C}_{IJ} = J^{-\frac{2}{3}}C_{IJ} = \tilde{\lambda}_A^2\left(N_I^{(A)}N_J^{(A)}\right)_A \tag{25}$$

while the deviatoric part of the Finger deformation tensor b_{ij} can be defined similarly as

$$\tilde{b}_{ij} = J^{-\frac{2}{3}}b_{ij} = \tilde{\lambda}_A^2\left(n_i^{(A)}n_j^{(A)}\right)_A \tag{26}$$

where the deviatoric principal stretches are defined as

$$\tilde{\lambda}_A = J^{-\frac{1}{3}}\lambda_A = (\lambda_1\lambda_2\lambda_3)^{-\frac{1}{3}}\lambda_A \tag{27}$$

The free energy W is then decomposed additively as:

$$W\left(X_K,\lambda_{(A)}\right) = {}^{dev}W\left(X_K,\tilde{\lambda}_{(A)}\right) + {}^{vol}W\left(X_K,J\right) \tag{28}$$

2.4 Simo–Serrin's Formulation

In Sections (2.1) and (2.2) we have presented the most general form of the isotropic strain energy function W in terms of of principal stretches:

$$W = W\left(X_K,\lambda_1,\lambda_2,\lambda_3,\right) \tag{29}$$

It was also shown that it is necessary to calculate the gradient $\partial W/\partial C_{IJ}$ in order to obtain the 2. Piola–Kirchhoff stress tensor S_{IJ} and accordingly other stress measures. Likewise, it was shown that the material tangent stiffness tensor $\mathcal{L}_{IJKL}$ (as well as the spatial tangent stiffness tensor $\mathcal{E}_{ijkl}$) requires second order derivatives of the strain energy function $\partial^2 W/(\partial C_{IJ}\,\partial C_{KL})$. In order to obtain these quantities we introduce a second order tensor $M_{IJ}^{(A)}$

$$\begin{aligned}
M_{IJ}^{(A)} &\overset{\text{def}}{=} \lambda_{(A)}^{-2}\, N_I^{(A)}\, N_J^{(A)} \\
&= \left(F_{iI}\right)^{-1}\left(n_i^{(A)}n_j^{(A)}\right)\left(F_{jJ}\right)^{-t} \\
&= \frac{1}{D_{(A)}}\left(C_{IJ} - \left(I_1-\lambda_{(A)}^2\right)\delta_{IJ} + I_3\lambda_{(A)}^{-2}(C^{-1})_{IJ}\right) \quad \text{from eq. (22)}
\end{aligned} \tag{30}$$

where $D_{(A)}$ was defined by equation (23). It then follows from equation (12):

$$C_{IJ} = \lambda_A^4\left(M_{IJ}^{(A)}\right)_A \tag{31}$$

We are now in a position to define the *Simo–Serrin* fourth order tensor $\mathcal{M}_{IJKL}$ as (c.f. [16]):

$$\mathcal{M}_{IJKL}^{(A)} \overset{\text{def}}{=} \frac{\partial M_{IJ}^{(A)}}{\partial C_{KL}} =$$

$$\frac{1}{D_{(A)}} \left(I_{IJKL} - \delta_{KL}\delta_{IJ} + \lambda^2_{(A)} \left(\delta_{IJ}\, M^{(A)}_{KL} + M^{(A)}_{IJ}\, \delta_{KL}\right) + \right.$$
$$+ \quad I_3\lambda^{-2}_{(A)} \left((C^{-1})_{IJ}(C^{-1})_{KL} + \frac{1}{2}\left((C^{-1})_{IK}(C^{-1})_{JL} + (C^{-1})_{IL}(C^{-1})_{JK}\right)\right) -$$
$$\left. - \quad \lambda^{-2}_{(A)}\, I_3 \left((C^{-1})_{IJ}\, M^{(A)}_{KL} + M^{(A)}_{IJ}\, (C^{-1})_{KL}\right) - D'_{(A)}\, M^{(A)}_{IJ}\, M^{(A)}_{KL}\right) \tag{32}$$

For a detailed derivation of $\mathcal{M}_{IJKL}$ interested readers are refered to [6].

2.5 Stress Measures

In Section (2.1) we have defined various stress measures in terms of derivatives of the free energy function W. With the free energy function decomposition, as defined in equation (28) we can appropriately decompose all the previously defined stress measures:

- 2. Piola–Kirchhoff stress tensor $S_{IJ} = 2\partial^{dev}W/\partial C_{IJ} + 2\partial^{vol}W/\partial C_{IJ}$
- Mandel stress tensor $T_{IJ} = 2C_{IK}\partial^{dev}W/\partial C_{KJ} + 2C_{IK}\partial^{vol}W/\partial C_{KJ}$
- 1. Piola–Kirchhoff stress tensor $P_{iJ} = 2\partial^{dev}W/\partial C_{IJ}(F_{iI})^t + 2\partial^{vol}W/\partial C_{IJ}(F_{iI})^t$
- Kirchhoff stress tensor $\tau_{ab} = 2F_{aI}(F_{bJ})^t\partial^{dev}W/\partial C_{IJ} + 2F_{aI}(F_{bJ})^t\partial^{vol}W/\partial C_{IJ}$

The derivative of the volumetric part of the free energy function is

$$\frac{\partial^{vol}W(J)}{\partial C_{IJ}} = \frac{\partial^{vol}W(J)}{\partial J}\frac{\partial J}{\partial C_{IJ}} = \frac{1}{2}\,\frac{\partial^{vol}W(J)}{\partial J}\; J\,(C^{-1})_{IJ} \tag{33}$$

while the derivative of the deviatoric part of the free energy function yields

$$\frac{\partial^{dev}W(\tilde{\lambda}_{(A)})}{\partial C_{IJ}} = \frac{\partial^{dev}W(\tilde{\lambda}_{(A)})}{\partial \lambda_{(A)}}\,\frac{\partial \lambda_{(A)}}{\partial C_{IJ}} = \frac{1}{2}\frac{\partial^{dev}W(\lambda_{(A)})}{\partial \lambda_{(A)}}\lambda_{(A)}(M^{(A)}_{IJ})_A = \frac{1}{2}w_A(M^{(A)}_{IJ})_A \tag{34}$$

and w_A can be written as (for a detailed derivation see [6]):

$$w_A = \frac{\partial^{dev}W(\lambda_{(A)})}{\partial \tilde{\lambda}_B}\,\frac{\partial \tilde{\lambda}_B}{\partial \lambda_{(A)}}\;\tilde{\lambda}_{(A)} = -\frac{1}{3}\,\frac{\partial^{dev}W(\tilde{\lambda}_{(A)})}{\partial \tilde{\lambda}_B}\;\tilde{\lambda}_B\; + \frac{\partial^{dev}W(\tilde{\lambda}_{(A)})}{\partial \tilde{\lambda}_{(A)}}\;\tilde{\lambda}_{(A)} \tag{35}$$

The decomposed 2. Piola–Kirchhoff stress tensor is

$$S_{IJ} = {}^{vol}S_{IJ} + {}^{dev}S_{IJ} = \frac{\partial^{vol}W(J)}{\partial J}\; J\, (C^{-1})_{IJ} + w_A\, (M_{IJ}^{(A)})_A \tag{36}$$

It is obvious that the only material dependent parts are derivatives in the form $\partial^{vol}W/\partial J$ and w_A, while the rest is independent of the chosen hyperelastic material model.

2.6 Tangent Stiffness Operator

The free energy function decomposition (28) is used together with the appropriate definitions made in section (2.1) toward the tangent stiffness operator decomposition:

$$\mathcal{L}_{IJKL} = {}^{vol}\mathcal{L}_{IJKL} + {}^{dev}\mathcal{L}_{IJKL} = 4\, \frac{\partial^2 \left({}^{vol}W\right)}{\partial C_{IJ}\, \partial C_{KL}} + 4\, \frac{\partial^2 \left({}^{dev}W\right)}{\partial C_{IJ}\, \partial C_{KL}} \tag{37}$$

The volumetric part $\partial^2 \left({}^{vol}W\right)/(\partial C_{IJ}\, \partial C_{KL})$ can be written as:

$$\frac{\partial^{2\,vol}W}{\partial C_{IJ}\partial C_{KL}} = \frac{1}{4}\left(J^2 \frac{\partial^2 \left({}^{vol}W\right)}{\partial J \partial J} + J \frac{\partial \left({}^{vol}W\right)}{\partial J}\right)(C^{-1})_{KL}(C^{-1})_{IJ} + \frac{1}{2} J \frac{\partial \left({}^{vol}W\right)}{\partial J} I_{IJKL}^{(C^{-1})} \tag{38}$$

and the deviatoric part $\partial^2 \left({}^{dev}W\right)/(\partial C_{IJ}\, \partial C_{KL})$ can be written in the following form:

$$\frac{\partial^{2\,dev}W(\lambda_{(A)})}{\partial C_{IJ}\partial C_{KL}} = \frac{1}{4}\, Y_{AB}\, (M_{KL}^{(B)})_B\, (M_{IJ}^{(A)})_A + \frac{1}{2}\, w_A\, (\mathcal{M}_{IJKL}^{(A)})_A \tag{39}$$

Detailed derivation are given in [6]. Finally, one can write the volumetric and deviatoric parts of the tangent stiffness tensors as:

$$
\begin{aligned}
{}^{vol}\mathcal{L}_{IJKL} &= J^2 \frac{\partial^{2\,vol}W(J)}{\partial J \partial J}(C^{-1})_{KL}(C^{-1})_{IJ} + \frac{\partial^{vol}W(J)}{\partial J}\left(J(C^{-1})_{KL}(C^{-1})_{IJ} + 2JI^{(C^{-1})}_{IJKL}\right) \\
{}^{dev}\mathcal{L}_{IJKL} &= Y_{AB}\,(M^{(B)}_{KL})_B\,(M^{(A)}_{IJ})_A + 2\,w_A\,(\mathcal{M}^{(A)}_{IJKL})_A
\end{aligned}
$$

where Y_{AB} is given as:

$$
\begin{aligned}
Y_{AB} = \quad & \frac{\partial^{dev}W}{\partial \tilde{\lambda}_{(A)}}\,\delta_{(A)(B)}\,\tilde{\lambda}_{(B)} + \frac{\partial^{2\,dev}W}{\partial \tilde{\lambda}_{(A)}\partial \tilde{\lambda}_{(B)}}\,\tilde{\lambda}_{(A)}\,\tilde{\lambda}_{(B)} \\
- \; & \frac{1}{3}\left(\frac{\partial^{2\,dev}W}{\partial \tilde{\lambda}_C \partial \tilde{\lambda}_{(B)}}\,\tilde{\lambda}_C\,\tilde{\lambda}_{(B)} + \frac{\partial^{dev}W}{\partial \tilde{\lambda}_{(B)}}\,\tilde{\lambda}_{(B)} + \frac{\partial^{2\,dev}W}{\partial \tilde{\lambda}_{(A)}\partial \tilde{\lambda}_D}\,\tilde{\lambda}_{(A)}\tilde{\lambda}_D + \frac{\partial^{dev}W}{\partial \tilde{\lambda}_{(A)}}\,\tilde{\lambda}_{(A)}\right) \\
+ \; & \frac{1}{9}\left(\frac{\partial^{2\,dev}W}{\partial \tilde{\lambda}_C \partial \tilde{\lambda}_D}\,\tilde{\lambda}_C\,\tilde{\lambda}_D + \frac{\partial^{dev}W}{\partial \tilde{\lambda}_D}\,\tilde{\lambda}_D\right)
\end{aligned} \tag{40}
$$

In a similar manner to the stress definitions it is clear that the only material–model dependent parts are Y_{AB}, w_A, $\partial W/\partial J$ and $\partial^2 W/\partial J^2$. The remaining second and fourth order tensors $M^{(A)}_{IJ}$, $\mathcal{M}^{(A)}_{IJKL}$ and $I^{(C^{-1})}_{IJKL}$ are independent of the choice of the material model. This observation has a practical consequence in that it is possible to create a *template derivations* for various hyperelastic isotropic material models. Only the first and second derivatives of the strain energy function with respect to deviatoric principal stretches ($\tilde{\lambda}_A$) and Jacobian (J) are needed in addition to the independent tensors, for the determination of various stress and tangent stiffness tensors. Tangent stiffness tensor $\mathcal{L}_{IJKL}$ is used on the finite element level for building stiffness matrix.

2.7 Isotropic Hyperelastic Models

In what follows, we will briefly overview a number of commonly used strain energy functions for isotropic hyperelastic solids. Moreover, we provide derivatives of the strain energy function, which are used in the definitions of the stress measures and tangent stiffness operators.

2.7.1 Ogden Model

A very general set of hyperelastic models was defined by Ogden [12]. The deviatoric strain energy is expressed as a function of principal stretches as:

$$^{dev}W = \sum_{r=1}^{N\to\infty} \frac{c_r}{\mu_r} \left(\tilde{\lambda}_1^{\mu_r} + \tilde{\lambda}_2^{\mu_r} + \tilde{\lambda}_3^{\mu_r} - 1\right)$$

where c_r and μ_r are material constants and it was used that $\tilde{\lambda}_i = J^{-\frac{1}{3}}\lambda_i$.

Derivatives needed for building tensors w_A and Y_{AB} are given by the following formulae:

$$\frac{\partial^{dev}W}{\partial\tilde{\lambda}_A} = \sum_{r=1}^{N\to\infty} c_r \left(\tilde{\lambda}_A\right)^{\mu_r-1} \quad ; \quad \frac{\partial^2\left(^{dev}W\right)}{\partial\tilde{\lambda}_A^2} = \sum_{r=1}^{N\to\infty} c_r \; (\mu_r - 1)\left(\tilde{\lambda}_A\right)^{\mu_r-2} \quad ; \quad \frac{\partial^2\left(^{dev}W\right)}{\partial\tilde{\lambda}_A\partial\tilde{\lambda}_B} = 0$$

2.7.2 Neo–Hookean Model

The deviatoric part of the Neo–Hookean isotropic elastic model (c.f. [12]) can be written as:

$$^{dev}W = \frac{G}{2}\left(\tilde{\lambda}_1^2 + \tilde{\lambda}_2^2 + \tilde{\lambda}_3^2 - 3\right) \tag{41}$$

while the volumetric part is:

$$^{vol}W = \frac{K_b}{2}\left(J^2 - 1\right)^2 \tag{42}$$

where G and K_b are the shear and bulk moduli respectively. Derivatives needed for building tensors w_A and Y_{AB} are given by the following formulae:

$$\frac{\partial^{dev}W}{\partial\tilde{\lambda}_A} = G\,\tilde{\lambda}_A \quad ; \quad \frac{\partial^2\left(^{dev}W\right)}{\partial\tilde{\lambda}_A^2} = G \quad ; \quad \frac{\partial^2\left(^{dev}W\right)}{\partial\tilde{\lambda}_A\partial\tilde{\lambda}_B} = 0$$

$$\frac{\partial^{vol}W}{\partial J} = 2K_bJ(J^2 - 1) \quad ; \quad \frac{\partial^2\left(^{vol}W\right)}{\partial J^2} = 2K_b(3J^2 - 1)$$

2.7.3 Mooney–Rivlin Model

The strain energy function for the Mooney-Rivlin material model (deviatoric behavior) is of the form (c.f. [12]):

$$\begin{aligned} {}^{dev}W &= \left(C_1\left(\tilde{I}_1 - 3\right) + C_2\left(\tilde{I}_2 - 3\right)\right) \\ \tilde{I}_1 &= \tilde{\lambda}_1^2 + \tilde{\lambda}_2^2 + \tilde{\lambda}_3^2 \\ \tilde{I}_1 &= \tilde{\lambda}_1^{-2} + \tilde{\lambda}_2^{-2} + \tilde{\lambda}_3^{-2} \end{aligned}$$

Derivatives needed for building tensors w_A and Y_{AB} are given by the following formulae:

$$\frac{\partial^{dev}W}{\partial\tilde{\lambda}_A} = 2\ C_1\ \tilde{\lambda}_A - 2\ C_2\ \tilde{\lambda}_A^{-3}\ \ ;\ \ \frac{\partial^2\left({}^{dev}W\right)}{\partial\tilde{\lambda}_A^2} = 2\ C_1 + 6\ C_2\ \tilde{\lambda}_A^{-4}\ \ ;\ \ \frac{\partial^2\left({}^{dev}W\right)}{\partial\tilde{\lambda}_A\partial\tilde{\lambda}_B} = 0$$

2.7.4 Logarithmic Model

A simple deviatoric strain energy function for the logarithmic hyperelastic material model can be presented in the form (c.f. [10]):

$${}^{dev}W = G\left(\left(\ln\tilde{\lambda}_1\right)^2 + \left(\ln\tilde{\lambda}_2\right)^2 + \left(\ln\tilde{\lambda}_3\right)^2\right) \tag{43}$$

while the volumetric part is suggested in the form (c.f. [15]):

$${}^{vol}W = \frac{K_b}{2}\left(\ln J\right)^2 \tag{44}$$

Derivatives needed for building tensors w_A and Y_{AB} are given by the following formulae:

$$\frac{\partial^{dev}W}{\partial\tilde{\lambda}_A} = 2\ G\ \left(\tilde{\lambda}_A\right)^{-1}\ \ ;\ \ \frac{\partial^2\left({}^{dev}W\right)}{\partial\tilde{\lambda}_A^2} = -2\ G\ \left(\tilde{\lambda}_A\right)^{-2}\ \ ;\ \ \frac{\partial^2\left({}^{dev}W\right)}{\partial\tilde{\lambda}_A\partial\tilde{\lambda}_B} = 0$$

$$\frac{d\left({}^{vol}W\right)}{dJ} = K_b\ J^{-1}\ln J\ \ ;\ \ \frac{d^2\left({}^{vol}W\right)}{dJ^2} = K_b\ J^{-2} - K_b\ J^{-2}\ln J$$

3 Implementation Strategy

The formulation for the general isotropic hyperelastic material model presented in the previous section has been implemented in a constitutive driver. In developments described in the next few sections we make use of a previously developed **nDarray** set of classes (c.f. Jeremić and Sture [7]). In particular we make frequent use of a class `tensor` which can be understood as a new concrete data type. We first briefly describe **nDarray** and and basic hyperelastic classes and then proceed to more specific issues for different hyperelastic material models. Finally, we comment on the portability and efficiency of the implementation.

3.1 nDarray Classes

The main purpose of the **nDarray** set of classes is to facilitate algebraic manipulations with matrices, vectors and tensors that are often found in computer codes for solving engineering problems. **nDarray** class library consists of `nDarray_rep`, `nDarray`, `matrix`, `vector` and `tensor` classes. Particularly interesting for this work is intuitive programming of tensorial formulae. For example, tensorial formula: $C_{il} = (A_{ijk} + B_{ijk}) * D_{jkl}$ is programmed as `C=(A("ijk")+B("ijk"))*D("jkl")`. Index notation is used by the program for proper tensor multiplications. Detailed description of **nDarray** class libraries is given in [7]. For readers interested in actual implementation details, source code, examples and makefiles are available at `http://sokocalo.cee.edu/~jeremic/nDarray/`

3.2 Basic Hyperelastic Classes

The basic class `lde_model` is defined as a container for common functions and operators for derived models. We define a set of virtual functions for the strain energy density W, then we devise both the deviatoric and volumetric components of the strain energy density function and their respective derivatives with respect to the deformation tensor. Each of these functions will be overloaded in each material model class.

```
virtual double W(tensor & )     const ; // strain energy function (W)
virtual double Wdev(tensor & )  const ; // W_dev deviatoric W
virtual double Wvol(tensor & )  const ; // W_vol volumetric W

virtual tensor dWodC(tensor & F)     const; // \partial W / \partial C
virtual tensor dWdevodC(tensor & F)  const; // \partial W_dev / \partial C
virtual tensor dWvolodC(tensor & F)  const; // \partial W_vol / \partial C
virtual tensor d2WodC2(tensor & F)    const; // \partial^2 W / \partial C^2
virtual tensor d2WdevodC2(tensor & F) const; // \partial^2 W_dev / \partial C^2
virtual tensor d2WvolodC2(tensor & F) const; // \partial^2 W_vol / \partial C^2
```

We also define a set of virtual functions that will return the second Piola–Kirchhoff stress tensors S_{ij} and stiffness tensor $\mathcal{L}_{ijkl}$ for a given deformation tensor C_{ij}. Both deviatoric and volumetric as well as the full component of stress and stiffness tensors can be obtained.

```
virtual stresstensor LDdevSPKstress(tensor & C) const ;
virtual stresstensor LDvolSPKstress(tensor & C) const ;
virtual stresstensor LDSPKstress(tensor & C) const ;

virtual tensor LDdevStiffnessTensorE(tensor & C) const ;
virtual tensor LDvolStiffnessTensorE(tensor & C) const ;
virtual tensor LDStiffnessTensorE(tensor & C) const ;
```

The previously defined functions are then used in all derived material model classes for obtaining stress and stiffness tensors.

As described in the previous sections, for various isotropic hyperelastic material models we can use general developments and implement tensorial variables that are common to isotropic hyperelastic models once, and then use them through the inheritance paradigm. Definitions for such common tensors follows:

```
    tensor M1_ij(tensor & C) const;
    tensor M2_ij(tensor & C) const;
    tensor M3_ij(tensor & C) const;
//
    tensor M1_ijkl(tensor & C) const;
    tensor M2_ijkl(tensor & C) const;
    tensor M3_ijkl(tensor & C) const;
//
    tensor ICm1_ijkl(tensor & C) const;
```

The material model dependent classes are implemented as virtual functions. They are later overloaded in class definitions for every material model. The transparent use of `tensor` data type allows for an efficient and elegant development of the presented implementation:

```
    virtual tensor wa(double lambda1, double lambda2, double lambda3) const;
    virtual tensor Yab(double lambda1, double lambda2, double lambda3) const;
    virtual double dWodJ(tensor &) const;
    virtual double d2WodJ2(tensor &) const;
```

In the following we present as an example the implementation of tensorial formulae for for $I^{(C^{-1})}_{IJKL}$ where the closed form reads:

$$I^{(C^{-1})}_{IJKL} = \frac{1}{2}\left((C^{-1})_{IK}(C^{-1})_{JL} + (C^{-1})_{IL}(C^{-1})_{JK}\right)$$

and for a given deformation tensor C_{ij}, $I^{(C^{-1})}_{IJKL}$ is implemented as:

```
tensor Cinv = C.inverse();
tensor Icm1 =   ((Cinv("ij")*Cinv("kl")).transpose0110()
               +  (Cinv("ij")*Cinv("kl")).transpose0111()
               *0.5;
```

where `transpose0110()` and `transpose0111()` are functions that transpose the fourth order tensor such that:

`transpose0110()`	$ijkl \rightarrow ikjl$
`transpose0111()`	$ijkl \rightarrow iljk$

3.3 Derived Classes

The isotropic hyperelastic material models described above are implemented through the following classes:

```
class Logarit : public lde_model
class NeoHook : public lde_model
class Ogden   : public lde_model
class MoonRiv : public lde_model
```

The isotropic hyperelastic material model classes inherit functions from basic class `lde_model` (`lde` stands for large deformation elasticity) and overload number of material model dependent functions:

```
    virtual tensor wa(double lambda1, double lambda2, double lambda3) const;
    virtual tensor Yab(double lambda1, double lambda2, double lambda3) const;
    virtual double dWodJ(tensor &) const;
    virtual double d2WodJ2(tensor &) const;
```

With these definitions and theoretical developments from section 2 we can easily implement general tensors as well as specific, material–model dependent tensors. By combining them in tensorial formulae we implement tensorial functions for stresses and stiffness tensors.

As an example, for a given right deformation tensor C_{ij} and for a Neo–Hookean material model named `NeoHook` the following lines of code gives second Piola–Kirchhoff stress tensor S_{ij} and then calculates the corresponding Kirchhoff stress tensor τ_{ij}:

```
  stresstensor S2 =  NeoHook.LDSPKstress(C);
  stresstensor tau = F("iA") * S2("AB") *F("jB");
```

3.4 Portability and Efficiency

The hyperelastic driver has been implemented using standard C++ language. Portability has been tested by compiling and computing example problems on a number of hardware platforms and by using a wide variety of compilers.

As for the efficiency, it was believed until recently that C++ is inherently slower than FORTRAN or the C programming languages (c.f. [13]). Work described in this paper also suffers from performance lower than that of FORTRAN code. However, it is questionable if some of the implemented concepts and formulae could be implemented in FORTRAN at all. We believe that the performance of our code should be measured by having in mind both the raw execution speed as well as the development time.

Recently developed techniques for C++ (c.f. [20], [21]) called *Expressions Template* and *Template Metaprograms* are showing performance on par with a hand tuned FORTRAN code and in some cases even better. There is an effort underway to implement template expressions in the **nDarray** class libraries.

4 Hyperelastic Simulations

A simple shear example is tested using Neo–Hookean and Logarithmic hyperelastic material models. The assumed material properties are $E = 206GPa$ and $\nu = 0.33$. Figure 2 depicts the coordinate system and Kirchhoff stresses τ_{ij}. It should be noted that the Kirchhoff stresses (τ_{ij}) are in this case equal to the Cauchy stresses σ_{ij} since the behavior is purely deviatoric. It is important to note that the simple shear example involves significant rotations of Lagrangian and Eulerian triads.

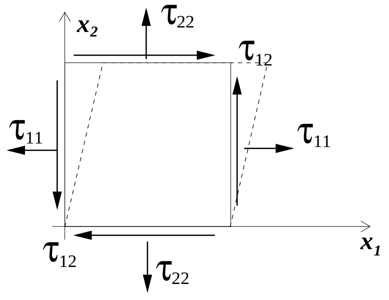

Figure 2: Coordinate system and Kirchhoff stresses on a simple shear specimen.

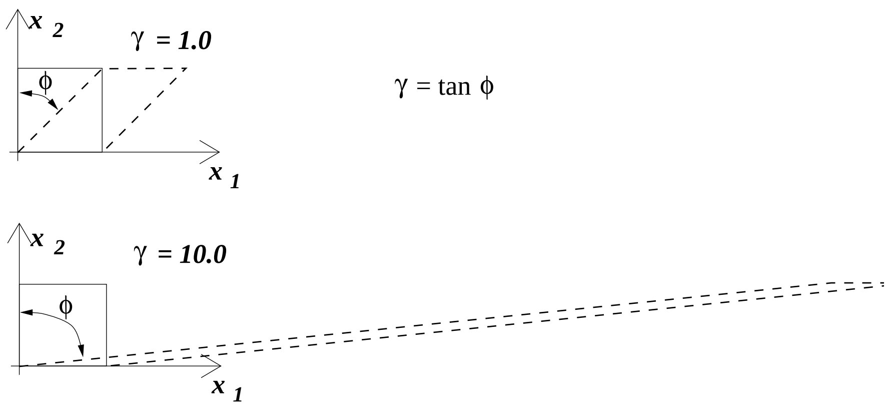

Figure 3: Modes of deformation for the simple shear specimen.

Figure 3 depicts a deformation mode for simple shear. Shear deformation angle of

45° yields $\gamma = 1$. Deformations larger than $\gamma = 1$ are usually not achieved[6] but are shown here for illustration purposes.

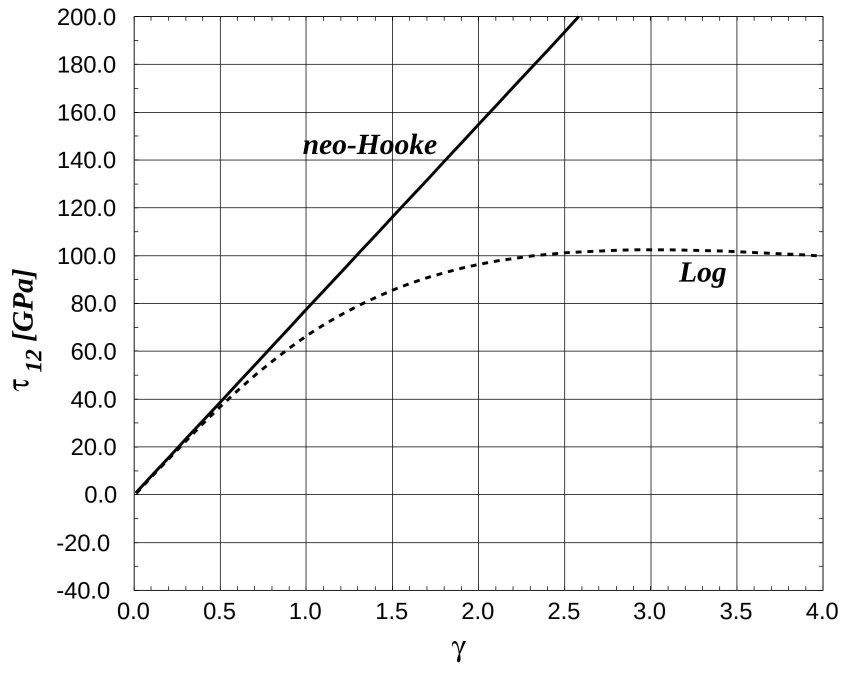

Figure 4: Neo–Hookean and Logarithmic stress results for simply sheared hyperelastic specimen. Shear deformation extends to $\gamma = 4$.

Figure 4 depicts the shear behavior of the hyperelastic specimen. The shear deformation is extending to $\gamma = 4$.

A number of interesting observations regarding shear stress results can be made. The solution using the Neo–Hookean model gives a linear response. The shear stress solution resulting from hyperelastic logarithmic model has a limit point, and apparent softening behavior for large shear deformation. Figure 5 shows an extended diagram of stress results, up to a deformation level of $\gamma = 10$.

Figures 6 and 7 are depicting combined hyperelastic stress results. It can be seen that for relatively small shear deformation the difference between Neo–Hookean and

[6]For example in the case of elastomeric bearings $\gamma < 1$ usually.

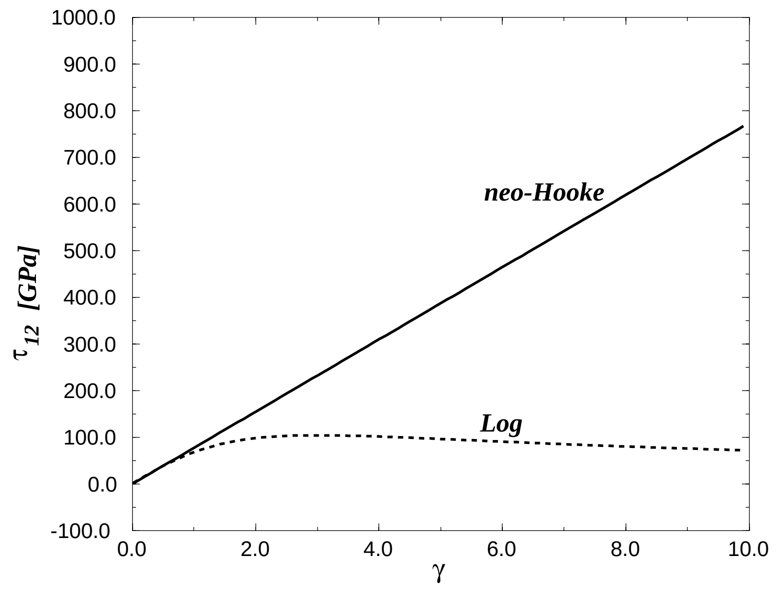

Figure 5: Extended simple shearing of a hyperelastic specimen. Neo–Hookean and Logarithmic stress results. Shear deformation extends to $\gamma = 10$.

Logarithmic models can be neglected. Normal stresses, τ_{11} and τ_{22} have a second order influence. With the increase of shear deformation, both Neo–Hookean and Logarithmic normal stress are increasing out of proportion, and they become much larger than the shear stresses.

5 Summary

In this paper we have shown how theoretical developments in isotropic hyperelasticity can lead to an object oriented implementation of the constitutive driver. After theoretical derivations for isotropic hyperelasticity in a general form, we have shown that many of the commonly used material models can be defined by changing a small set of material dependent scalar and tensorial functions. The implementation follows naturally from theoretical developments and is transparent, if one uses `tensor` concrete data types developed earlier by the authors (c.f. [7]). We have also shown numerical modeling

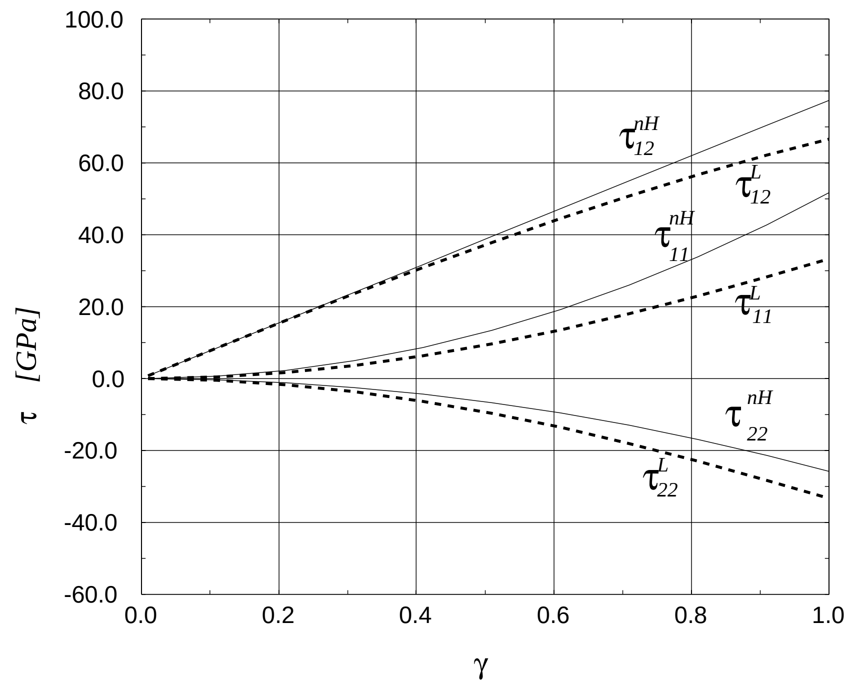

Figure 6: Combined hyperelastic stress results. Shear deformation up to $\gamma = 1$.

examples together with parts of the actual implementation. The described implementation is part of larger set of classes called **FEMtools**, in which we have included finite elements classes, solvers classes, solution procedures for non–linear finite element system of equations classes, elasto–plastic set of classes and other useful functions.

Acknowledgment

The authors gratefully acknowledge partial support by NASA Grant NAS8-38779 from Marshall Space Flight Center. The authors also wish to thank reviewers for helpful comments and suggestions.

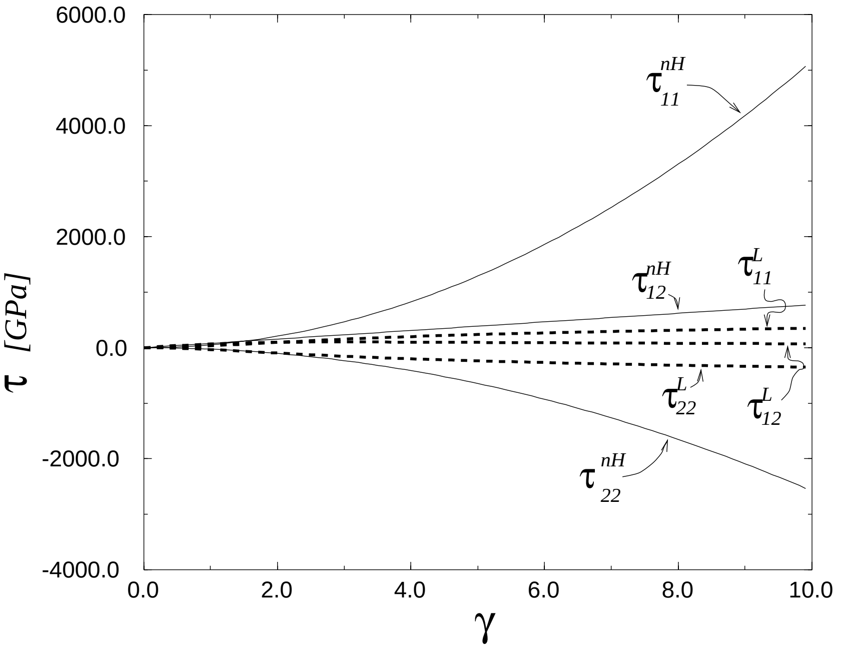

Figure 7: Combined hyperelastic stress results. Shear deformation up to $\gamma = 10$.

References

[1] ANSI/ISO. *Working Paper for Draft Proposed International Standard for Information Systems–Programming Language C++*. Washington DC, April 1995. Doc. No. ANSI X3J16/95-0087 ISO WG21/N0687.

[2] Booch, G. *Object Oriented Analysis and Design with Applications*, second ed. Series in Object–Oriented Software Engineering. Benjamin Cummings, 1994.

[3] Coplien, J. O. *Advanced C++, Programming Styles and Idioms.* Addison – Wesley Publishing Company, 1992.

[4] Eckel, B. *Using C++*. Osborne McGraw – Hill, 1989.

[5] Hill, R. Aspects of invariance in solid mechanics. In *Advances in Applied Mechanics*, C.-S. Yih, Ed., vol. 18. Academic Press, 1978, pp. 1–72.

[6] Jeremić, B. *Finite Deformation Hyperelasto–Plasticity of Geomaterials.* PhD thesis, University of Colorado at Boulder, July 1997.

[7] Jeremić, B., and Sture, S. Tensor data objects in finite element programming. *International Journal for Numerical Methods in Engineering 41* (1998), 113–126.

[8] Koenig, A. C++ columns. *Journal of Object Oriented Programming* (1989 - 1993).

[9] Marsden, J. E., and Hughes, T. J. R. *Mathematical Foundations of Elasticity.* Prentice Hall Inc,, 1983. local CM65; 4QA 931.M42 ; ISBN 0-13-561076-1.

[10] Miehe, C. Aspects of the formulation and finite element implementation of large strain isotropic elasticity. *International Journal for Numerical Methods in Engineering 37* (1994), 1981–2004.

[11] Morman, K. N. The generalized strain meassures with application to nonhomogeneous deformation in rubber–like solids. *Journal of Applied Mechanics 53* (1986), 726–728.

[12] OGDEN, R. W. *Non–Linear Elastic Deformations.* Series in mathematics and its applications. Ellis Horwood Limited, 1984.

[13] ROBISON, A. D. C++ gets faster for scientific computing. *Computers in Physics 10*, 5 (Sept/Oct 1996), 458–462.

[14] SERRIN, J. *Encyclopedia of Physics*, vol. VIII/1. Springer Verlag, 1959, ch. Mathematical Principles of Classical Fluid Mechanics.

[15] SIMO, J. C., AND ORTIZ, M. A unified approach to finite deformation elastoplastic analysis based on the use of hyperelastic constitutive equations. *Computer Methods in Applied Mechanics and Engineering 49* (1985), 221–245.

[16] SIMO, J. C., AND TAYLOR, R. L. Quasi–incompressible finite elasticity in principal stretches. continuum basis and numerical algorithms. *Computer Methods in Applied Mechanics and Engineering 85* (1991), 273–310.

[17] STROUSTRUP, B. *The Design and Evolution of C++.* Addison–Wesley Publishing Company, 1994.

[18] TING, T. C. T. Determination of $C^{1/2}$, $C^{-1/2}$ and more general isotropic tensor functions of C. *Journal of Elasticity 15* (1985), 319–323.

[19] VARIOUS AUTHORS. The C++ report: Columns on C++, 1991-.

[20] VELDHUIZEN, T. Expression templates. *C++ Report 7*, 5 (June 1995), 26–31.

[21] VELDHUIZEN, T. Rapid linear algebra in C++. *Dr. Dobb's Journal* (August 1996).