IN-39
167938
p-55

NASA Technical Memorandum 106124

Explicit Robust Schemes for Implementation of a Class of Principal Value-Based Constitutive Models: Symbolic and Numeric Implementation

S.M. Arnold
Lewis Research Center
Cleveland, Ohio

and

A.F. Saleeb, H.Q. Tan, and Y. Zhang
University of Akron
Akron, Ohio

May 1993

NASA

Explicit Robust Schemes for Implementation of a Class of Principal Value-Based Constitutive Models: Symbolic and Numeric Implementation

S. M. Arnold
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

A. F. Saleeb, H.Q. Tan, and Y. Zang
University of Akron
Akron, Ohio 44325

Abstract

The issue of developing effective and robust schemes to implement a class of the Ogden-type hyperelastic constitutive models is addressed. To this end, special purpose functions (running under MACSYMA) are developed for the symbolic derivation, evaluation, and automatic FORTRAN code generation of explicit expressions for the corresponding stress function and material tangent stiffness tensors. These explicit forms are valid over the entire deformation range, since the singularities resulting from repeated principal-stretch values have been theorectically removed. The required computational algorithms are outlined, and the resulting FORTRAN computer code is presented.

1 Introduction

To a great extent, constitutive models of the so-called generalized Rivlin-Mooney type [1,2] (i.e., with the stored strain energy density written as a polynomial function in terms of the deformation invariants) have dominated the phenomenological theory of isotropic hyperelasticity [1-6]. Such models dominate the related computational literature on finite-strain elasticity [7-9] as well. Recently though, alternative representations in terms of the principal stretches have become increasingly popular in nonlinear finite element analyses [6,8,10]. However, from the viewpoint of numerical implementation, the use of these models presents a number of unique and difficult problems, which do not arise in classical representations using the strain invariants. The main difficulty is that (in addition to being reasonably complicated functions of the strain components) taken separately, the main constituents of the deformation tensor (i.e., principal values and associated eigenvectors) are, in general, not uniquely defined and continuously differentiable functions. A careful consideration is thus called for in implementing constitutive models formulated in terms of these principal-strain measures; this was the main problem addressed by Saleeb and Arnold [11] . They bypassed the difficulty entirely by resorting to explicit derivations of appropriate forms of the material tangent-stiffness matrices, which are valid for the entire deformation range. The explicit expressions they developed [11] were for two specific forms of the Ogden-type, strain-energy functions, which actually encompass many of the popular representations currently in use for rubber materials. Results were obtained by simply applying a systematic limiting procedure for one type of tensor-valued function and its spectral representation.

Symbolic computation specializes in exact computation with numbers, formulas, vectors, matrices, equations and the like. Numerical computation, on the other hand, uses floating-point numbers to compute approximate solutions to problems of practical interest. The two approaches are complementary and, when combined into an integrated form, can be very powerful in engineering applications. In particular, application of symbolic manipulation can provide significant incentive for the development of new constitutive theories and their applications, for example, finite element. Recently, a problem-oriented, self-contained, symbolic expert system, named SDICE (see [12-13]), was developed; it is capable of efficiently deriving, in analytical form, potential based constitutive models whose representations are in terms of the *classical* invariant formulation [14-15]. In addition, the FORTRAN code associated with the resulting analytical expressions can be automatically generated.

The objective of the present paper is to discuss three special purpose functions (SDIFF, SDIFFEV, and TEMPLATE) running under DOE MACSYMA [16]. These three functions have been developed to allow the derivation and automatic FORTRAN code generation of *alternative* potential based constitutive models composed of principal values and their associated eigenvectors, as discussed in reference 11. All three functions are written at the MACSYMA command level. In the future, these functions will be integrated into the collection of special purpose functions known as SDICE. This paper begins by reviewing *highlights* of the previous theoretical development and discussing the associated computer algorithm for the derivation of the explicit expressions for the second Piola Kirchhoff stress tensor S_{ij} and the material moduli tensor D_{ijkl}. The paper concludes with the evaluation of a separable strain energy function, similar to that discussed in reference 11, and its associated FORTRAN source code generation.

2 Background

The theoretical development of *singularity-free* representations for principal value-based constitutive models has been discussed at length in reference 11. For brevity, we will confine our discussion, for illustrative purposes, to hyperelastic isotropic materials whose strain energy function W can be taken to have the following separable functional dependence:

$$W = W(\lambda_i) = \sum_{n=1}^{p} \overline{a}_n(\lambda_1^{\alpha_n} + \lambda_2^{\alpha_n} + \lambda_3^{\alpha_n}) \tag{1}$$

where λ_i represents the principal values of the right Cauchy-Green deformation tensor C_{ij}. Denoting n_i ($i = 1, 2,$ or 3) to be the associated eigenvectors of C_{ij}, we can define

$$C_{ij} = \sum_{l=1}^{3} \lambda_{(l)} N_{ij}^{(l)} \tag{2}$$

where $N_{ij}^{(l)}$ is defined as

$$N_{ij}^{(l)} = n_i^l n_j^l \tag{3}$$

and is often referred to as the (orthogonal) *eigenprojection* operator related to the associated eigenvectors of C_{ij}.

Equation (2) is valid for the case when all three eigenvalues, λ_i, are distinct. However, for the case when two eigenvalues are the same (i.e., double coalescence $\lambda_1 \neq \lambda_2 = \lambda_3 = \lambda$) we have

$$C_{ij} = (\lambda_1 - \lambda) N_{ij}^{(1)} + \lambda\delta_{ij} \tag{4}$$

And for the case of triple coalescence ($\lambda_1 = \lambda_2 = \lambda_3 = \lambda$), we have

$$C_{ij} = \lambda \sum_{l=1}^{3} N_{ij}^{(l)} = \lambda\delta_{ij}. \tag{5}$$

Similarly, by manipulating equations (2) and (4), we can obtain explicit expressions for $N_{ij}^{(r)}$ in terms of C_{ij}:

$$N_{ij}^{(r)} = \frac{1}{(\lambda_r - \lambda_s)(\lambda_s - \lambda_t)}[(C_{ij} - \lambda_s\delta_{ij})(C_{ij} - \lambda_t\delta_{ij})] \tag{6}$$

and

$$N_{ij}^{(r)} = \frac{1}{(\lambda_r - \lambda)}(C_{ij} - \lambda\delta_{ij}) \tag{7}$$

In the preceding equations, the r, s, and t are any cyclic permutation of (1,2, or 3). These definitions, equations (2) through (7), will prove very useful in obtaining the pertinent singularity-free directional derivatives of both the strain-energy potential function W and the stress function $S_{ij} = S_{ij}(C_{ij})$.

The explicit singularity-free expressions for the second Piola Kirchhoff stress tensor $S_{ij}(C_{ij})$ are defined as

$$S_{ij} = 2\frac{\partial W}{\partial C_{ij}} \equiv S_{ij}(C_{ij}) \tag{8}$$

and those for the material moduli tensor $D_{ijkl}(C_{ij})$ are obtained by applying the directional derivative formula to S_{ij}, that is

$$D_{ijkl} = 2\frac{\partial S_{ij}}{\partial C_{kl}} = 4\frac{\partial^2 W}{\partial C_{ij}\partial C_{kl}} \equiv D_{ijkl}(C_{ij}) \tag{9}$$

As a result, the explicit expressions of the functional dependence of tensors S_{ij} and D_{ijkl} on C_{ij} can be obtained directly for the following three cases: 1) all three eigenvalues are distinct; 2) a single singularity ($\lambda_1 \neq \lambda_2 = \lambda_3 = \lambda$, i.e., double coalescence) is present; or 3) a double singularity ($\lambda_1 \neq \lambda_2 = \lambda_3 = \lambda$, i.e., triple coalescence) is present.

3 Computer Algorithm

The objective of the present study was to construct three special purpose functions (SDIFF, SDIFFEV, and TEMPLATE) written at the MACSYMA command level that can, respectively,

(1) Derive explicit expressions for the stress tensor S_{ij} (eqs. (8)) and material tensor D_{ijkl} (eqs. (9))) given three, one, or no distinct eigenvalues

(2) Evaluate symbolically the expressions generated by SDIFF for a given strain-energy function W

(3) Evaluate the expressions generated by SDIFF and use the built-in MACSYMA function **gentran** to automatically generate the associated FORTRAN code needed to evaluate the expressions numerically for a given function, W.

These special purpose functions contain a list of built-in MACSYMA instructions (**factor**, **expand**, **ev**, **ratsubst**, **diff**, **limit**, and **for-loops**, to name a few) arranged in a specific algorithmic order. Each function, then, can be thought of as a macro command.

3.1 SDIFF(case)

Issuing the command SDIFF invokes the following algorithm (consisting of 15 steps) for automatic derivation of S_{ij} and D_{ijkl}. In this context, case $\equiv 1$ indicates that all three eigenvalues are distinct; case $\equiv 2$ indicates that only one is distinct; and case $\equiv 3$, that none are distinct.
To obtain S_{ij},

(1) Differentiate W with respect to C_{ij} (see eq. (8))

$$S_{ij} = \sum_{l=1}^{3} 2 \frac{\partial W}{\partial \lambda_{(l)}} \frac{\partial \lambda_{(l)}}{\partial C_{ij}} \tag{10}$$

(2) Apply the special directional derivative rules obtained from equation (2), that is,

$$N_{ij}^{(l)} = \frac{\partial \lambda_{(l)}}{\partial C_{ij}} \tag{11}$$

whose value is given in equation (6).

(3) Obtain typical scalar derivatives by using the built-in **diff** command:

$$s(\lambda_{(l)}) = 2 \frac{\partial W}{\partial \lambda_{(l)}} \tag{12}$$

(4) Multiply the results $s(\lambda_{(l)})$ and $N_{ij}^{(l)}$, then sum and factor out coefficients of like terms (i.e., $C_{ik}C_{kj}$, C_{ij}, and δ_{ij}), thereby obtaining the functional dependence of S_{ij} on C_{ij}. In the case of three distinct eigenvalues,

$$S_{ij} = aC_{ik}C_{kj} + bC_{ij} + c\delta_{ij} \tag{13}$$

where δ_{ij} is the second order identity tensor and

$$a = -m[s(\lambda_1)(\lambda_2 - \lambda_3) + s(\lambda_2)(\lambda_3 - \lambda_1) + s(\lambda_3)(\lambda_1 - \lambda_2)] \tag{14}$$

$$b = m[s(\lambda_1)(\lambda_2^2 - \lambda_3^2) + s(\lambda_2)(\lambda_3^2 - \lambda_1^2) + s(\lambda_3)(\lambda_1^2 - \lambda_2^2)] \tag{15}$$

$$c = -m[s(\lambda_1)\lambda_2\lambda_3(\lambda_2 - \lambda_3) + s(\lambda_2)\lambda_3\lambda_1(\lambda_3 - \lambda_1) + s(\lambda_3)\lambda_1\lambda_2(\lambda_1 - \lambda_2)] \tag{16}$$

and

$$m = \frac{1}{(\lambda_1 - \lambda_2)(\lambda_2 - \lambda_3)(\lambda_3 - \lambda_1)} \tag{17}$$

To obtain D_{ijkl} :

(5) Differentiate S_{ij} with respect to C_{kl} (see eq. (9)):

$$D_{ijkl} = 2\{a[\frac{1}{2}(\delta_{ik}\delta_{ml} + \delta_{il}\delta_{mk})C_{mj} + \frac{1}{2}C_{im}(\delta_{jk}\delta_{ml} + \delta_{jl}\delta_{mk})] + \sum_{r=1}^{3} \frac{\partial a}{\partial \lambda_r}\frac{\partial \lambda_r}{\partial C_{kl}} C_{im}C_{mj} + b[\frac{1}{2}(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk})] + \sum_{r=1}^{3} \frac{\partial b}{\partial \lambda_r}\frac{\partial \lambda_r}{\partial C_{kl}} C_{ij} + \sum_{r=1}^{3} \frac{\partial c}{\partial \lambda_r}\frac{\partial \lambda_r}{\partial C_{kl}} \delta_{ij}\} \tag{18}$$

(6) Apply the special directional derivative rule

$$N_{ij}^{(l)} = \frac{\partial \lambda_{(l)}}{\partial C_{ij}} \tag{19}$$

(7) Obtain the nine scalar derivatives,

$$\frac{\partial a}{\partial \lambda_r}, \frac{\partial b}{\partial \lambda_r}, \frac{\partial c}{\partial \lambda_r} \tag{20}$$

of equations (14) to (16) for $r = 1, 2$, and 3.

(8) Substitute the preceding expressions and group-like terms, thus giving

$$D_{ijkl} = 2a_1 C_{kl}^2 C_{ij}^2 + 2a_2(C_{kl}C_{ij}^2 + C_{kl}^2 C_{ij}) + 2a_3(\delta_{kl}C_{ij}^2 + C_{kl}^2\delta_{ij}) + 2a_4(C_{kl}C_{ij}) + 2a_5(\delta_{ik}C_{lj} + C_{ik}\delta_{jl} + \delta_{kl}C_{ij} + C_{kl}\delta_{ij}) + 2a_6(\delta_{ik}\delta_{jl} + \delta_{kl}\delta_{ij}) \tag{21}$$

(9) For comparison of equation (21) to the forms described in reference 11, section 4, we make use of the symmetry properties of C_{ij} and δ_{ij}, and define two second order symmetric tensors, P and Q,

$$P_{ijkl}(G, H) = G_{ik}H_{jl} + G_{il}H_{jk} \tag{22}$$

$$Q_{ijkl}(G, H) = G_{ik}H_{jl} + G_{ij}H_{jk} + G_{jl}H_{ik} + G_{jk}H_{il} \tag{23}$$

such that upon substitution we obtain

$$D_{ijkl} = a_1 P(C_{kl}^2, C_{ij}^2) + a_2[P(C_{kl}^2, C_{ij}) + P(C_{kl}, C_{ij}^2)] + a_3[Q(C_{kl}^2 \delta_{ij}) + P(\delta_{kl}, C_{ij}^2)] + a_4 P(C_{kl}, C_{ij})$$

$$+ a_5[Q(C_{kl}, \delta_{ij}) + Q(\delta_{kl}, C_{ij})] + 2a_6 I_{ijkl} \quad (24)$$

where $a1, a2, ...a6$ are as defined in reference 11 and the preceding equation (eq. (24)) is directly comparable to equations 4.6a in reference 11. Note that

$$I_{ijkl} = \frac{1}{2}[\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}] \quad (25)$$

$$C^2_{ij} = C_{im}C_{mj} \quad (26)$$

in the foregoing expressions.

Next, given the case of nondistinct eigenvalues, for example, case II when ($\lambda_1 \neq \lambda_2 = \lambda_3 = \lambda$), or case III when ($\lambda_1 = \lambda_2 = \lambda_3$), we must

(10) Remove the singularity (case II) or singularities (case III) by defining an appropriate "path" for taking the limit of a,b,c and $C_{ik}C_{kj}$ in equations (13); that is,

- For case II
  $\lambda_1, \lambda_2 = \lambda + \Delta, \lambda_3 = \lambda - \Delta$
- For case III
  $\lambda_1 = \lambda, \lambda_2 = \lambda + \Delta, \lambda_3 = \lambda - \Delta$

(11) Substitute the preceding eigenvalues into the expressions for a,b,and c in equations (13), and take the limit of the numerator and denominator of a,b, and c as $\Delta \rightarrow 0$.

(12) If both limits are zero, apply l'Hospital's rule recursively to the now equivalent one dimensional problem. For example, given case II, we obtain

$$lim_{\Delta \rightarrow 0} a(\Delta) = \frac{1}{(\lambda_1 - \lambda)^2}[s(\lambda_1) - s(\lambda) - (\lambda_1 - \lambda)s'(\lambda)] \quad (27)$$

$$lim_{\Delta \rightarrow 0} b(\Delta) = \frac{1}{(\lambda_1 - \lambda)^2}[-2\lambda[s(\lambda_1) - s(\lambda)] + (\lambda_1^2 - \lambda^2)s'(\lambda)] \quad (28)$$

$$lim_{\Delta \rightarrow 0} c(\Delta) = \frac{1}{(\lambda_1 - \lambda)^2}[\lambda^2 s(\lambda_1) + \lambda_1(\lambda_1 - 2\lambda)s(\lambda) - \lambda_1\lambda(\lambda_1 - \lambda)s'(\lambda)] \quad (29)$$

where

$$s'(\lambda_r) = \frac{\partial s(\lambda_r)}{\partial \lambda_{(r)}} = \frac{2\partial^2 W}{\partial \lambda_{(r)} \partial \lambda_{(r)}} \quad (30)$$

(13) Simplify $C_{ik}C_{kj}$ by using the definition of C_{ij} and N_{ij}, that is,

$$C_{ij}^2 = \lambda_1 N_{ij}^{(1)} + \lambda_2 N_{ij}^{(2)} + \lambda_3 N_{ij}^{(3)} \tag{31}$$

In addition, by using $\delta_{ij} = N_{ij}^{(1)} + N_{ij}^{(2)} + N_{ij}^{(3)}$ and equation (4), for case II we obtain

$$C_{ik}C_{kj} = \frac{1}{(\lambda_1 - \lambda)}[(\lambda_1^2 - \lambda^2)C_{ij} + (\lambda^2\lambda_1 - \lambda\lambda_1^2)\delta_{ij}] \tag{32}$$

and with equation (5) for case III we have

$$C_{ik}C_{kj} = \lambda\delta_{ij}. \tag{33}$$

(14) Substitute the limiting values of a,b,c and $C_{ik}C_{kj}$ into equations (13) and group like terms to obtain the modified stress function, S_{ij}, and the $\overline{a}$ and $\overline{b}$ values for case II:

$$S_{ij} = \overline{a}C_{ij} + \overline{b}\delta_{ij} \tag{34}$$

where

$$\overline{a} = \frac{s(\lambda_1) - s(\lambda)}{(\lambda_1 - \lambda)} \tag{35}$$

and

$$\overline{b} = -\frac{[s(\lambda_1)\lambda - s(\lambda)\lambda_1]}{(\lambda_1 - \lambda)} \tag{36}$$

For case III,

$$S_{ij} = s(\lambda)\delta_{ij} \tag{37}$$

(15) Repeat steps 5 through 10, but now use the appropriate modified stress function. For case II, this results in,

$$D_{ijkl} = 2(\frac{\partial\overline{a}}{\partial\lambda_1}N_{ij}^{(1)} + \frac{\partial\overline{a}}{\partial\lambda}N_{ij}^{(2)})C_{kl} + 2\overline{a}\delta_{ijkl} + 2(\frac{\partial\overline{b}}{\partial\lambda_1}N_{ij}^{(1)} + \frac{\partial\overline{b}}{\partial\lambda}N_{ij}^{(2)})\delta_{kl} \tag{38}$$

And for case III,

$$D_{ijkl} = 2\frac{\partial\overline{\overline{a}}}{\partial\lambda_p}\frac{\partial\lambda_p}{\partial C_{kl}}\delta_{ij} \tag{39}$$

where the special derivative rule of equation (7) is now used.

The value in automating the foregoing procedure is evident: not only does this special purpose function relieve the user of the tedious manual derivation process but it also ensures analytical accuracy. This was illustrated prior to the publication of reference 11 in that a number of errors in the hand derivation were detected, verified and corrected. Furthermore, as will be discussed in a sequel paper [17], this automated derivation procedure facilitated the generalization of the preceding expressions to the *general nonseparable* case, which to the author's knowledge, has eluded researchers to date . Also, it should be apparent that this derivation process needs to be executed only once. However, with each new definition of W evaluation of $s(\lambda_{(l)})$ and $s'(\lambda_{(l)})$ is required in order to specialize the needed coefficients; for example, a,b and c, and $a1, a2, ...a6$. As a consequence, this motivated the development of SDIFFEV, as described in the next section.

3.2 SDIFFEV(case, W)

The function SDIFFEV symbolically evaluates the explicit expressions for the stress function S_{ij} and material moduli tensor D_{ijkl}, which were generated by SDIFF and stored in a LISP [18] level disk file. Only the coefficients of these expressions need be changed when a different strain-energy function is specified. The evaluation algorithm is illustrated here in pseudo code:

SDIFFEV(case, W)
IF (**diff**(W,λ_1, λ_2),**diff**(W,λ_2, λ_3), **diff**(W,λ_3, λ_1))=0 THEN
Display message: W is separable.
SEP = 1
ELSE Display message: W is non separable. SEP = 2
ENDIF
IF case=1 THEN .
Call Subroutine A
ELSE IF case = 2 THEN
. Call Subroutine B
ELSE IF case = 3 THEN
. Call Subroutine C
END IF
End

Subroutine A
IF SEP = 2 THEN
Do loop i = 1, 6
a[i] = ea[i] (ea[i] are the coefficients of tensor D stored on the disk file produced by **SDIFF(1)**)
End loop
ELSE IF SEP = 1 THEN

s[2,1] = s[3,1] = s[3,2] = 0
ENDIF
Do loop i = 1, 6
a[i] = **ev**(ea[i])
End loop
Do loop i = 1, 3
s[i] = 2***diff**(W,λ_i,1)
s[i,i] = 2***diff**(W,λ_i,2)
IF SEP = 2 THEN
Do loop j = 1, 3
s[i,j] = **diff**(W,λ_i, λ_j,2)
End loop
ENDIF
End loop
Call **OPTION**
Return End

Subroutine B

W = **ev**(W,$\lambda_3 = \lambda_2$)
IF SEP = 2 THEN
Do loop i = 1, 3
b[i] = eb[i] (eb[i] are the coefficients of tensor D stored on the disk file produced by **SDIFF (2)**)
End loop
ELSE IF SEP = 1 THEN
s[2,1] = 0
Do loop i = 1, 6
b[i] = **ev**(eb[i])
End loop
Do loop i = 1, 2
s[i] = 2***diff**(W,λ_i,1)
s[i,i] = 2***diff**(W,λ_i,2)
IF SEP = 2 THEN
Do loop j = 1, 2
s[i,j] = **diff**(W,λ_i, λ_j,2)
End loop
ENDIF
End loop
Call **OPTION**
Return

Subroutine C

W = **ev**(W,$\lambda_3 = \lambda_2 = \lambda_1$)
s[1]=2***diff**(W,λ_1,1)
s[1,1]=2***diff**(W,λ_1,2)
Call **OPTION**
Return

Subroutine OPTION

Display the formulae S[i,j] and D[i,j,k,l]. Then, ask if user wants to see the symbolic form for the given function W, the intermediate step evaluations, and the derivatives of W.
READ(type **y**, or **n** to the question)
DISPLAY the options user may choose
Return

3.3 TEMPLATE ()

The function TEMPLATE is similar to the function SDIFFEV in that both will evaluate the explicit expressions obtained from SDIFF. As a result neither can be employed unless preceded by an invocation of SDIFF. TEMPLATE, however, will automatically generate the associated FORTRAN source code needed to evaluate the expressions numerically for a given potential function W. Code generation is accomplished by utilizing the built-in MACSYMA function **gentran**, and a number of template files. The template files can be thought of as a framework for the generation of four basic FORTRAN subroutines (i.e., the main driving routine COMPSD and the three subroutines – one each for case I , case II, and case III) and numerous functions. Appendix A contains the template file for the main driving routine COMPSD. This subroutine is constructed for easy implementation into a finite element code; the input requirements are the strain tensor C_m (denoted as cmu) and its associated eigenvalues (i.e., $\lambda_1, \lambda_2, \lambda_3$ denoted by gl1, gl2, and gl3 respectively), and the outputs are the stress tensor S_n (denoted as s), and the material moduli tensor D_{nm} (denoted as d). Here, n and m run from 1 to 6. The only automated code generation required is that for the subroutines COMPSD1, COMPSD2, and COMPSD3. These codes are generated by issuing the command <**gentranin**> . The subroutines COMPSD1, COMPSD2, and COMPSD3 are associated with case I ($\lambda_1 \neq \lambda_2 \neq \lambda_3$), case II ($\lambda_1, \lambda = \lambda_2 = \lambda_3$), and case III ($\lambda = \lambda_1 = \lambda_2 = \lambda_3$), described in Section 2.0. The template files corresponding to these three cases are shown, respectively in appendixes B,C, and D. Note that in these routines, most of the FORTRAN code is automatically generated, since it pertains to the definition of coefficients a,b,c; $a1, a2, ..., a6$, and the first ($s1, s2, s3$, see eq. (12)) and second scalar ($s11, s22, s33$, see eq. (30)) derivatives of the strain energy function W. The

gentran commands are enclosed by double inequality signs, that is, $\ll \gg$. Finally, all functions that are associated with a given case have been included in the corresponding appendix.

4 Example

As an example, consider the case in which the strain energy function W of equation (1) consists of only two terms; that is,

$$W = x1(gl1^{y1} + gl2^{y1} + gl3^{y1}) + x2(gl1^{y2} + gl2^{y2} + gl3^{y2}) \tag{40}$$

where x1, x2, y1, and y2 are material coefficients and $gl1 = \lambda_1$, $gl2 = \lambda_2$, and $gl3 = \lambda_3$. After defining W, we can symbolically obtain the analytical expressions for S_{ij} and D_{ijkl} (given the case of three distinct eigenvalues) by merely issuing the command

$$\mathbf{sdiffev}(1, W);$$

at the MACSYMA command level. Case II or III can just as easily be obtained by substituting a 2 or 3 in place of the 1 in this command. The resulting output is shown in appendix E where the expressions for the coefficients a,b,c and a1,a2,...a6 could be further simplified and manipulated, if desired, by using other MACSYMA built-in functions. Typically, however, the analyst will ultimately desire a FORTRAN code for the resulting expressions in order to solve a given structural problem using the foregoing constitutive model. This code, described in the previous section, can easily be obtained by issuing the command

template();

at the MACSYMA command level. The generated FORTRAN code will then be stored in a file named temp.f. The automatically generated FORTRAN code for the above example is shown in appendix F.

5 Summary of Results

Taken separately, the main constituents of the deformation tensor (i.e., principal values and associated eigenvectors) are, in general, not uniquely defined and continuously differentiable functions. Careful consideration is thus called for in implementing constitutive models formulated in terms of these principal-strain measures. This difficulty can be entirely bypassed by resorting to explicit symbolic derivations of appropriate forms of the material tangent-stiffness matrices, which are valid over the entire deformation range. Furthermore, to enhance effective utilization and implementation of the present results, automatic FORTRAN generation of these explicit expressions has been pursued and achieved. As a result, three special purpose functions(SDIFF, SDIFFEV and TEMPLATE), running under MACSYMA, have been developed and verified.

References

[1] Green, A.E. and Adkins, J.E.: Large Elastic Deformations. 2nd Edition. Oxford Univ. Press, Clarendon, 1970.

[2] Rivlin, R.S.: Large Elastic Deformations of Isotropic Materials; Part III: Experiments on the Deformation of Rubber. Philo. Trans. Roy. Soc., London, Ser. A, Vol. 243, pp.251-288, 1951.

[3] Rivlin, R.S.: "Forty Years of Nonlinear Continuum Mechanics", Proc. IX Int., Congress on Rheology, Mexico, 1984.

[4] Treloar, L.R.G.: The Physics of Rubber Elasticity, 3rd Ed., Oxford Univ. Press., U.K., 1975.

[5] Ogden, R.W., "Recent Advances in the Phenomenological Theory of Rubber Elasticity," J.Rubber Chem. Tech., Vol. 59, pp. 361-383, 1986.

[6] Finney, R.H., and Kumar, A.: "Development of Material Constants for Non-linear Finite Element Analysis," J. Rubber Chem. Tech., Vol.. 61, pp. 879-891, 1988.

[7] Oden, J.T.: Finite Elements of Nonlinear Continua, McGraw Hill, New York, 1972.

[8] Sussman, T.; and Bathe, K.J.: A Finite Element Formulation for Nonlinear Incompressible Elastic and Inelastic Analysis, Comp. Struct. , Vol. 26, pp. 357-409, 1987.

[9] Simo, J.C.; Taylor, R.L; and Pister, K.S.: "Variational and Projection Methods for the Volume Constraint in Finite Deformation Elastoplasticity", Comp. Meth. Appl. Mech. Engng., Vol. 51, pp. 177-208, 1985.

[10] Chang, T.Y., Saleeb, A.F., and Li, G.: "Large Strain Analysis of Rubber-Like Materials by a Mixed Finite Element", Computational Mech., Vol. 8, No. 4, pp. 221-233, 1991.

[11] Saleeb.A.F.; and Arnold, S.M.: Explicit Robust Schemes for Implementation of a Class Of Principle Value-Based Constitutive Models: Theoretical Development. NASA TM 105345, 1991

[12] Arnold, S.M.; and Tan, H.Q.: "Symbolic Derivation of Potential Based Constitutive Equations", Computational Mech., Vol. 6, pp. 237-246, 1990.

[13] Arnold, S.M.; Tan, H.Q.; and Dong, X.: Application of Symbolic Computations to The Constitutive Modeling of Structural Materials. Symbolic Computations and Their Impact on Mechanics, Noor, A.K., Elishakoff, I. and Hulbert, G., eds., PVP-Vol. 205, ASME, pp.215-229.,1990.

[14] Malvern, L.E.: Introduction to the Mechanics of a Continuous Medium. Prentice-Hall, 1969.

[15] Chen, W.F.; and Saleeb, A.F.: Constitutive Equations For Engineering Materials Volume 1: Elasticity and Modeling, John Wiley & Sons, 1982.

[16] MATHLAB Group: MACSYMA Reference Manual. Version 10. Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1984.

[17] Arnold, S.M.; et al.: Explicit Robust Schemes for Implementation of General Principal Value-Based Constitutive Models, submitted to Comp. and Struct., 1993.

[18] Wilensky, R.: LISPcraft. W.W. Norton Co, 1984.

APPENDIX A: Template File Associated With COMPSD
The Main Driver Routine

```
c *****************************************************
c       This is the template subroutine to calculate
c       tensor S and D. inputs are eigenvalues gl1,gl2,gl3,
c       and cmu(6). cmu is assumed to be engineering strain(e),
c       e.g. the Cauchy-green deformation tensor cm(3,3) is related
c       to cmu(6) in the following fashion:
c       cm(1,1)=cmu(1), cm(2,2) = cmu(2), cm(3,3) =cmu(3),
c       cmu(4)=2*cm(1,2), cm(5)=2*cm(2,3),cmu(6)=2*cm(1,3).
c       The outputs are the second order tensor S(6)
c       and forth order tensor D(6,6) are related in the
c       following way:
c       S=D*C
c       S(1,1) = S(1)
c       S(2,2) = S(2)
c       S(3,3) = S(3)
c       S(1,2) = S(4)
c       S(2,3) = S(5)
c       S(3,1) = S(6)
c       C(1,1) = C(1)
c       C(2,2) = C(2)
c       C(3,3) = C(3)
c       C(1,2) = C(4)
c       C(2,3) = C(5)
c       C(3,1) = C(6)
c
        subroutine compsd(gl1,gl2,gl3,cmu,s,d)
        real*8 gl1,gl2,gl3,ts(3,3),td(3,3,3,3)
        real*8 delt(3,3),delt4(3,3,3,3),s(6),d(6,6)
        real*8 cmu(6),cm(3,3)
c
c       converts cmu(6) to matrix cm(3,3) in a way that
c       cm(1,2)=cm(2,1)=cmu(4),cm(2,3)=cm(3,2)=cmu(5),
c       cm(1,3)=cm(3,1)=cmu(6).
c
```

```
      do 5 i=1,3
         do 5 j=1,3
           if (i.eq.j) then
             iq=i
             cm(i,j)=cmu(iq)
           else if (i.ne.j) then
              if((i+j).eq.3) iq=4
              if((i+j).eq.4) iq=6
              if((i+j).eq.5) iq=5
              cm(i,j)=cmu(iq)/2
           end if
          continue
 5 continue
c
c       Initiates the second identity tensor delt(3,3) which
c       is a 2X2 identity matrix.
c
        do 6 i=1,3
           delt(i,i)=1.0
 6      continue
c
c       Computes the forth order identity tensor delt4(3,3,3,3)
c       by definition.
c
        do 7 i=1,3
         do 7 j=1,3
           delt4(i,j,i,j)=delt(i,i)*delt(j,j)+delt(i,j)*delt(j,i)
           delt4(i,j,j,i)=delt4(i,j,i,j)
 7     continue
c
c*********************************************************************
c       For different eigenvalues gl1,gl2,gl3 the computation
c       is different. case1 is gl1#gl2#gl3 call subroutine comsd1.
c       case2 is gl3=gl2#gl1 or gl1=gl3#gl2 or gl1=gl2#gl3 then
c       call subroutine compsd2. case3 is gl1=gl2=gl3 call subroutine
c       compsd3.
c*********************************************************************
        if ((gl1.ne.gl2).and.(gl2.ne.gl3).and.(gl1.ne.gl3)) then
        call compsd1(gl1,gl2,gl3,delt,delt4,cm,ts,td)
        else if((gl2.eq.gl3).and.(gl1.ne.gl3)) then
```

```
          call compsd2(gl1,gl2,delt,delt4,cm,ts,td)
     else if((gl1.eq.gl2).and.(gl3.ne.gl2)) then
          gl1=gl3
          call compsd2(gl1,gl2,delt,delt4,cm,ts,td)
     else if((gl1.eq.gl3).and.(gl2.ne.gl3)) then
          gl1=gl2
          gl2=gl3
          call compsd2(gl1,gl2,delt,delt4,cm,ts,td)
     else
     call compsd3(gl1,delt,delt4,ts,td)
     end if
c
c    Rewrite the tensor ts(i,j) td(i,j,k,l)to S(i) and D(i,j)
c    respectively by using the symetric property.
c    converts ts(3,3) s(6) and td(3,3,3,3) to D(6,6)
c
     do 8 i=1,3
       do 8 j=i,3
            if (i.eq.j) iq=i
    if (i.eq.1.and.j.eq.2) iq=4
    if (i.eq.2.and.j.eq.3) iq=5
    if (i.eq.1.and.j.eq.3) iq=6
            s(iq)=ts(i,j)
       continue
 8   continue
     do 9 i=1,3
       do 9 j=i,3
         d(i,j)=td(i,i,j,j)
       continue
 9   continue
     do  10 i=1,3
        d(i,4)=td(i,i,1,2)+td(i,i,2,1)
        d(i,5)=td(i,i,2,3)+td(i,i,3,2)
        d(i,6)=td(i,i,3,1)+td(i,i,1,3)
 10   continue
     d(4,4)=(td(1,2,1,2)+td(1,2,2,1)+td(2,1,1,2)+td(2,1,2,1))/2.
     d(4,5)=(td(1,2,2,3)+td(1,2,3,2)+td(2,1,2,3)+td(2,1,3,2))/2.
     d(4,6)=(td(1,2,1,3)+td(1,2,3,1)+td(2,1,1,3)+td(2,1,3,1))/2.
     d(5,5)=(td(2,3,2,3)+td(2,3,3,2)+td(3,2,2,3)+td(3,2,3,2))/2.
     d(5,6)=(td(2,3,1,3)+td(2,3,3,1)+td(3,2,1,3)+td(3,2,3,1))/2.
```

```
      d(6,6)=(td(3,1,1,3)+td(3,1,3,1)+td(1,3,1,3)+td(1,3,3,1))/2.
      do  11 i = 1,6
        do 11 j = 1,6
           d(i,j) = d(j,i)
 11   continue
c
c     prints out the inputs gl1,gl2,gl3,cmu(6) and outputs S and D
c
      print*, 'gl1=', gl1
      print*, 'gl2=', gl2
      print*, 'gl3=', gl3
      print*, 'Input tensor C(6):'
      print*, (cmu(i), i = 1,6)
      print*,"second order tensor S(6):"
      print*, (s(i), i=1,6)
      print*, "The forth order tensor D(6,6):"
      do 101 i=1,6
          print*,(d(i,j),j=1,6)
 101  continue
      return
      end
c
      subroutine compsd1(gl1,gl2,gl3,delt,delt4,cm,ts,td)
<<
      gentranin("case11.tem")$
>>
      subroutine compsd2(gl1,gl2,delt,delt4,cm,ts,td)
<<
      gentranin("case22.tem")$
>>
      subroutine compsd3(gl1,delt,delt4,ts,td)
<<
      gentranin("case3.tem")$

>>
```

```
c
c       This subroutine computes P and Q forth order tensors
c       which we define in tensor D.
c
        subroutine pqcom(cm1,cm2,p,q)
        real*8 cm1(3,3),cm2(3,3), p(3,3,3,3),q(3,3,3,3)
        do 100 i=1,3
         do 100 j=1,3
          do 100 k=1,3
           do 100 l=1,3
            p(i,j,k,l)=cm1(i,k)*cm2(j,l)+cm1(i,l)*cm2(j,k)
            q(i,j,k,l)=p(i,j,k,l)+cm1(j,l)*cm2(i,k)+cm1(j,k)*cm2(i,l)
 100    continue
        return
        end
c
c       This subroutine computes matrix product cmXcm.
c
        subroutine product(mat1,cmm)
        real*8 mat1(3,3),cmm(3,3),sum
        do 25 i=1,3
        do 25 j=1,3
          sum=0.0
          do 26 k=1,3
           sum=sum+mat1(i,k)*mat1(k,j)
 26       continue
          cmm(i,j)=sum
 25     continue
        return
        end
```

APPENDIX B: Template File Associated With COMPSD1 Valid For Three Distinct Eigenvalues

```
      real*8 gl1,gl2,gl3,ts(3,3),td(3,3,3,3)
      real*8 cm(3,3),delt(3,3),delt4(3,3,3,3),p(3,3,3,3)
      real*8 q(3,3,3,3),cmm(3,3),p1(3,3,3,3),p21(3,3,3,3)
      real*8 p31(3,3,3,3),q11(3,3,3,3),q12(3,3,3,3),p22(3,3,3,3)
      real*8 q21(3,3,3,3),q22(3,3,3,3),a,b,c,a1,a2,a3,a4,a5,a6
c
c     Obtains cmm(3,3)=cm(3,3)*cm(3,3) from subroutine product
      call product(cm,cmm)
c     Uses the formula we derived in code to compute second order
c     tensor ts(3,3).
c
<<
      gentran(for i:1 thru 3 do
        (for j:1 thru 3 do
         (ts[i,j]:a(gl1,gl2,gl3)*cmm[i,j]+b(gl1,gl2,gl3)
                 *cm[i,j]+c(gl1,gl2,gl3)*delt[i,j])))$
>>
c
c     Call subroutine to compute all the functions we defined
c     when we derived forth order tenosor td, namely P(i,j,k,l)
c     and Q(i,j,k,l) which are the functions of cm(3,3) and
c     the matrix product cmm(3,3).
c
      call pqcom(cmm,cmm,p1,q)
      call pqcom(cmm,cm,p21,q)
      call pqcom(cm,cmm,p22,q)
      call pqcom(cm,cm,p31,q)
      call pqcom(cmm,delt,p,q11)
      call pqcom(delt,cmm,p,q12)
      call pqcom(cm,delt,p,q21)
      call pqcom(delt,cm,p,q22)
```

```
c
c       Computes forth order tensor td(i,j,k,l)
c
<<
        gentran(for i:1 thru 3 do
         (for j:1 thru 3 do
          (for k:1 thru 3 do
           (for l:1 thru 3 do
            (td[i,j,k,l]:a1(gl1,gl2,gl3)*p1[i,j,k,l]+a2(gl1,gl2,gl3)
           *(p21[i,j,k,l]+p22[i,j,k,l])+a4(gl1,gl2,gl3)*p31[i,j,k,l]
            +a3(gl1,gl2,gl3)*(q11[i,j,k,l]+q12[i,j,k,l])+
             a5(gl1,gl2,gl3)*(q21[i,j,k,l]+q22[i,j,k,l])+
             a6(gl1,gl2,gl3)*delt4[i,j,k,l])))))$
>>
        return
        end
c
c       a,b,c,a1-a6 are the coefficients we derived in code.
c
<<
        gentran(a(gl1,gl2,gl3):=block(type(function,a),
                                      type("real*8",gl1,gl2,gl3),
                                      type("real*8",a,s1,s2,s3),
                                      a:eval(ta)))$
>>
<<
        gentran(b(gl1,gl2,gl3):=block(type(function,b),
                                      type("real*8",b,gl1,gl2,gl3),
                                      type("real*8",s1,s2,s3),
                                      b:eval(tb)))$
>>
<<

        gentran(c(gl1,gl2,gl3):=block(type(function,c),
                                      type("real*8",c,gl1,gl2,gl3),
                                      type("real*8",s1,s2,s3),
                                      c:eval(tc)))$
>>
```

```
<<
        gentran(a1(gl1,gl2,gl3):=block(type(function,a1),
                                  type("real*8",a1,gl1,gl2,gl3),
                       type("real*8",s1,s2,s3,s11,s22,s33),
                        a1:eval(ta1)))$
>>
<<
        gentran(a2(gl1,gl2,gl3):=block(type(function,a2),
                       type("real*8",a2,gl1,gl2,gl3),
                       type("real*8",s1,s2,s3,s11,s22,s33),
                       a2:eval(ta2)))$
>>
<<
        gentran(a3(gl1,gl2,gl3):=block(type(function,a3),
                       type("real*8",a3,gl1,gl2,gl3),
                       type("real*8",s1,s2,s3,s11,s22,s33),
                       a3:eval(ta3)))$
>>
<<

        gentran(a4(gl1,gl2,gl3):=block(type(function,a4),
                       type("real*8",a4,gl1,gl2,gl3),
                       type("real*8",s1,s2,s3,s11,s22,s33),
                       a4:eval(ta4)))$
>>
<<
        gentran(a5(gl1,gl2,gl3):=block(type(function,a5),
                       type("real*8",a5,gl1,gl2,gl3),
                       type("real*8",s1,s2,s3,s11,s22,s33),
                       a5:eval(ta5)))$
>>
<<
        gentran(a6(gl1,gl2,gl3):=block(type(function,a6),
                       type("real*8",a6,gl1,gl2,gl3),
                       type("real*8",s1,s2,s3,s11,s22,s33),
                       a6:eval(ta6)))$
>>
```

```
c
c       The s1,s2,s3,s11,s22,s33 are derivatives of W
c
        function s1(gl1,gl2,gl3)
        <<cut(var);>>
<<
        gentran(type(  "real*8",s1,gl1,gl2,gl3),
                      s1:2*eval(diff(w,'gl1,1)))$
>>
        return
        end
c
        function s2(gl1,gl2,gl3)
        <<cut(var);>>
<<
        gentran(type(   "real*8",s2,gl1,gl2,gl3),
                       s2:2*eval(diff(w,'gl2,1)))$
>>
        return
        end
c
        function s3(gl1,gl2,gl3)
        <<cut(var);>>
<<
        gentran(type(  "real*8",s3,gl1,gl2,gl3),
                       s3:2*eval(diff(w,'gl3,1)))$
>>
        return
        end
c
        function s11(gl1,gl2,gl3)
        <<cut(var);>>
<<
        gentran(type(  "real*8",s11,gl1,gl2,gl3),
                       s11:2*eval(diff(w,'gl1,2)))$
>>
        return
        end
c
```

```
        function s22(gl1,gl2,gl3)
        <<cut(var);>>
<<
        gentran(type("real*8",s22,gl1,gl2,gl3),
                s22:2*eval(diff(w,'gl2,2)))$
>>
        return
        end
c
        function s33(gl1,gl2,gl3)
        <<cut(var);>>
<<
        gentran(type(  "real*8",s33,gl1,gl2,gl3),
                     s33:2*eval(diff(w,'gl3,2)))$
>>
        return
        end
```

APPENDIX C: Template File Associated With COMPSD2 Valid For Double Coalesence Case

```
c
c
        real*8 gl1,gl2,ts(3,3),td(3,3,3,3)
        real*8 cm(3,3),delt(3,3),delt4(3,3,3,3),p1(3,3,3,3)
        real*8 q2(3,3,3,3),q1(3,3,3,3),p(3,3,3,3),q(3,3,3,3)
        real*8 b1,b2,b3, abar,bbar
c
c       Computes second order tensor ts(i,j) based on the formula
c       derived in code.
c
<<
        gentran(for i:1 thru 3 do
         (for j:1 thru 3 do
         (ts[i,j]:abar(gl1,gl2)*cm[i,j]+bbar(gl1,gl2)*delt[i,j])))$
>>
c
c       Call subroutine to get P, Q which are defined in code.
c
        call pqcom(cm,cm,p1,q)
        call pqcom(cm,delt,p,q1)
        call pqcom(delt,cm,p,q2)
c
c       Computes tensor td(i,j,k,l).
c
<<
        gentran(for i:1 thru 3 do
        (for j:1 thru 3 do
         (for k:1 thru 3 do
          (for l:1 thru 3 do
           (td[i,j,k,l]:b1(gl1,gl2)*p1[i,j,k,l]+b2(gl1,gl2)*
           (q1[i,j,k,l]+q2[i,j,k,l])+b3(gl1,gl2)*
            delt4[i,j,k,l])))))$
>>
        return
        end
```

```
c
c       abar,bbar are b1, b2, b3 functions derived in code.
c
<<
        gentran(abar(gl1,gl2):=block(type(function,abar),
                                     type("real*8",abar,gl1,gl2),
                                     type("real*8", ss1,ss2),
                                     abar:eval(abar)))$
>>
<<
        gentran(bbar(gl1,gl2):=block(type(function,bbar),
                                      type("real*8",bbar,gl1,gl2),
                                      type("real*8", ss1,ss2),
                                      bbar:eval(bbar)))$
>>
<<
        gentran(b1(gl1,gl2):=block(type(function,b1),
                                     type("real*8",b1,gl1,gl2),
                                     type("real*8", ss1,ss2,ss11,ss22),
                                     b1:eval(tb1)))$
>>
<<
        gentran(b2(gl1,gl2):=block(type(function,b2),
                                     type("real*8",b2,gl1,gl2),
                                     type("real*8", ss1,ss2,ss11,ss22),
                                     b2:eval(tb2)))$
>>
<<
        gentran(b3(gl1,gl2):=block(type(function,b3),
                                     type("real*8",b3,gl1,gl2),
                                     type("real*8", ss1,ss2,ss11,ss22),
                                     b3:eval(tb3)))$
>>
<<
        neww:subst(['gl3='gl2],w)$
>>
```

```
c
c       ss1,ss2,ss11,ss22 are derivatives of W.
c
        function ss1(gl1,gl2)
        <<cut(var);>>
<<
        gentran(type("real*8",ss1,gl1,gl2),
               ss1:2*eval(diff(neww,'gl1,1)))$
>>
        return
        end
c
        function ss2(gl1,gl2)
        <<cut(var);>>
<<
        gentran(type("real*8",ss2,gl1,gl2),
        ss2:2*eval(diff(neww,'gl2,1)))$
>>
        return
        end
c
        function ss11(gl1,gl2)
        <<cut(var);>>
<<
        gentran(type("real*8",ss11,gl1,gl2),
                     ss11:2*eval(diff(neww,'gl1,2)))$
>>
        return
        end
c
        function ss22(gl1,gl2)
        <<cut(var);>>
<<
        gentran(type("real*8",ss22,gl1,gl2),
               ss22:2*eval(diff(neww,'gl2,2)))$
>>
        return
        end
```

APPENDIX D: Template File Associated With COMPSD3 Valid For The Triple Coalesence Case

```
c
        real*8 gl1,ts(3,3),td(3,3,3,3),delt(3,3),delt4(3,3,3,3)
        real*8 cc1,abbar
c
<<
        gentran(for i:1 thru 3 do
          (for j:1 thru 3 do
           (ts[i,j]:abbar(gl1)*delt[i,j])))$
>>
<<
        gentran(for i:1 thru 3 do
         (for j:1 thru 3 do
          (for k:1 thru 3 do
           (for l:1 thru 3 do
           (td[i,j,k,l]:cc1(gl1)*delt4[i,j,k,l])))))$
>>
        return
        end
<<
        gentran(abbar(gl1):=block(type(function,abbar),
                            type("real*8", abbar,gl1),
                            abbar:eval(abbar)))$
>>
<<
        gentran(cc1(gl1):=block(type(function,cc1),
                          type("real*8", cc1,gl1),
                          cc1:eval(cc1)))$
>>
<<
        www:subst(['gl3='gl1, 'gl2='gl1],w)$
>>
```

```
c
      function sss1(gl1)
      <<cut(var);>>
<<
      gentran(type("real*8",sss1,gl1),
        sss1:2*eval(diff(www,'gl1,1)))$
>>
      return
      end
c
      function sss11(gl1)
      <<cut(var);>>
<<
      gentran(type("real*8",sss11,gl1),
              sss11:2*eval(diff(www,'gl1,2)))$
>>
      return
      end
```

APPENDIX E: Listing of MACSYMA Session Resulting From Issuing The SDIFFEV Command

```
c6) sdiffev(1,w);

w is a separable function.

                y2     y2     y2              y1     y1     y1
       w = (gl3 + gl2 + gl1   ) x2 + (gl3 + gl2 + gl1   ) x1

This is case 1 with distinct  eigenvalues gl1#gl2#gl3.

Please type y if your answer is yes, otherwise type n to skip it.

Do you want to display the second order tensor s[i,j]?
y;

                s     = a c     c     + c delt     + b c
                 i, j      i, k  k, j          i, j       i, j

Do you want to display c[i,j] and delta[i,j]?
y;

                                   [ 1 0 0 ]
                                   [       ]
                      delt       = [ 0 1 0 ]
                          i, j     [       ]
                                   [ 0 0 1 ]

                    c     = gl3 n3   n3   + gl2 n2   n2   + gl1 n1   n1
                     i, j           i    j            i    j            i    j

n1, n2, n3 are eigenvectors associatedd with eigenvalues gl1, gl2, gl3.

If c[i,j]) is given then the eigenvectors can be computed
```

Do you want to display a,b,c in s[i,j] form?
y;

$$a = - \frac{s3}{t31\ (t31 - t21)} + \frac{s2}{t21\ (t31 - t21)} - \frac{s1}{t21\ t31}$$

$$b = - \frac{s3\ (2\ t31 - t21 - 2\ gl3)}{t31\ (t31 - t21)} + \frac{s1\ (t31 + t21 + 2\ gl1)}{t21\ t31} - \frac{s2\ (t31 - t21 + gl2 + gl1)}{t21\ (t31 - t21)}$$

$$c = - \frac{s3\ (t31 - gl3)\ (t31 - t21 - gl3)}{t31\ (t31 - t21)} - \frac{s2\ (t21 - gl2)\ (t31 + gl1)}{t21\ (t31 - t21)} - \frac{s1\ (t21 + gl1)\ (t31 + gl1)}{t21\ t31}$$

Do you want to display t21,t31, s1,s2,s3?
y;

$$t21 = gl2 - gl1$$

$$t31 = gl3 - gl1$$

s1,s2,s3 are the first derivatives of W with respect to gl1,gl2,gl3.

$$s1 = gl1^{y2 - 1}\ x2\ y2 + gl1^{y1 - 1}\ x1\ y1$$

$$s2 = g12^{y2 - 1} x2\, y2 + g12^{y1 - 1} x1\, y1$$

$$s3 = g13^{y2 - 1} x2\, y2 + g13^{y1 - 1} x1\, y1$$

Do you want to display the forth order tensor d[i,j,k,l]?
y;

$$d_{i, j, k, l} = [a_6\, delt4 + a_3\, (q(delt, c^2) + q(c^2, delt)) + a_5\, (q(delt, c) + q(c, delt)) + a_1\, p(c^2, c^2) + a_2\, (p(c^2, c) + p(c, c^2)) + a_4\, p(c, c)]$$

Do you want to display the functions p,q and delt4?
y;

$$p(g, h) = g_{i, k}\, h_{j, l} + g_{i, l}\, h_{j, k}$$

$$q(g, h) = g_{i, k}\, h_{j, l} + h_{i, k}\, g_{j, l} + 2\, g_{i, l}\, h_{j, k}$$

$$delt4 = delt_{i, k}\, delt_{j, l} + delt_{i, l}\, delt_{j, k}$$

Do you want to continue displaying a_1

y;

$$a_1 = \frac{2\ s3\ (2\ t31 - t21)}{t31^3\ (t31 - t21)^3} - \frac{2\ s1\ (t31 + t21)}{t21^3\ t31^3} - \frac{s33}{t31^2\ (t31 - t21)^2} - \frac{s22}{t21^2\ (t31 - t21)^2} + \frac{2\ s2\ (t31 - 2\ t21)}{t21^3\ (t31 - t21)^3} - \frac{s11}{t21^2\ t31^2}$$

Do you want to continue displaying a_2 ?

y;

$$a_2 = \frac{s1\ (2\ t31^2 + 3\ t21\ t31 + 4\ g11\ t31 + 2\ t21^2 + 4\ g11\ t21)}{t21^3\ t31^3}$$

$$- \frac{s2\ (2\ t31^2 - 3\ t21\ t31 + 4\ g11\ t31 - t21^2 - 8\ g11\ t21)}{t21^3\ (t31 - t21)^3}$$

$$- \frac{s3\ (t31^2 + 3\ t21\ t31 + 8\ g11\ t31 - 2\ t21^2 - 4\ g11\ t21)}{t31^3\ (t31 - t21)^3}$$

$$- \frac{s33\ (2\ t31 - t21 - 2\ g13)}{t31^2\ (t31 - t21)^2} + \frac{s11\ (t31 + t21 + 2\ g11)}{t21^2\ t31^2}$$

```
     s22 (t31 - t21 + gl2 + gl1)
+ ---------------------------
             2              2
          t21   (t31 - t21)

Do  you want to continue displaying a ?
                                      3
y;
               3          2          2      2                                  2
a  = - s3 (t31  - 2 t21 t31  - gl1 t31  + t21  t31 - 3 gl1 t21 t31 -4 gl1 t31
 3

             2         2          3           3
 + 2 gl1 t21  + 2 gl1  t21)/(t31  (t31 - t21) )

               2           2       2                                2
 - s1 (t21 t31  + 2 gl1 t31  + t21  t31 + 3 gl1 t21 t31 + 2 gl1  t31

             2         2          3    3
 + 2 gl1 t21  + 2 gl1  t21)/(t21  t31 )

               2           2         2                                 2           3
 + s2 (t21 t31  + 2 gl1 t31  - 2 t21  t31 - 3 gl1 t21 t31 + 2 gl1  t31 + t21

          2         2          3           3           3
 - gl1 t21  - 4 gl1  t21)/(t21  (t31 - t21) )
   s33 (t31 - gl3) (t31 - t21 - gl3)    s22 (t21 - gl2) (t31 + gl1)
 - --------------------------------- + ---------------------------
              2             2                    2             2
           t31  (t31 - t21)                   t21  (t31 - t21)

   s11  (t21 + gl1)  (t31 + gl1)
 - -----------------------------
                  2      2
               t21   t31
```

Do you want to continue displaying a_4 ?

y;

$$a_4 = \frac{2\ s3\ (t21 + 2\ gl1)\ (t31^2 + t21\ t31 + 4\ gl1\ t31 - t21^2 - 2\ gl1\ t21)}{t31^3\ (t31 - t21)^3}$$

$$- \frac{2\ s1\ (t31 + t21 + 2\ gl1)\ (t31^2 + t21\ t31 + 2\ gl1\ t31 + t21^2 + 2\ gl1\ t21)}{t21^3\ t31^3}$$

$$+ \frac{2\ s2\ (t31 + 2\ gl1)\ (t31^2 - t21\ t31 + 2\ gl1\ t31 - t21^2 - 4\ gl1\ t21)}{t21^3\ (t31 - t21)^3}$$

$$- \frac{s33\ (2\ t31 - t21 - 2\ gl3)^2}{t31^2\ (t31 - t21)^2} - \frac{s11\ (t31 + t21 + 2\ gl1)^2}{t21^2\ t31^2}$$

$$- \frac{s22\ (t31 - t21 + gl2 + gl1)^2}{t21^2\ (t31 - t21)^2}$$

Do you want to continue displaying a_5 ?

y;

$$
\begin{aligned}
a_5 = {} & s1\,(t21\, t31^3 + 2\, gl1\, t31^3 + t21^2\, t31^2 + 6\, gl1\, t21\, t31^2 + 6\, gl1^2\, t31^2 \\
& + t21^3\, t31 + 6\, gl1\, t21^2\, t31 + 9\, gl1^2\, t21\, t31 + 4\, gl1^3\, t31 + 2\, gl1\, t21^3 \\
& + 6\, gl1^2\, t21^2 + 4\, gl1^3\, t21)/(t21^3\, t31^3) \\
& + s3\,(t21\, t31^3 + 2\, gl1\, t31^3 - 2\, t21^2\, t31^2 - 6\, gl1\, t21\, t31^2 - 3\, gl1^2\, t31^2 \\
& + t21^3\, t31 - 9\, gl1^2\, t21\, t31 - 8\, gl1^3\, t31 + 2\, gl1\, t21^3 + 6\, gl1^2\, t21^2 \\
& + 4\, gl1^3\, t21)/(t31^3\, (t31 - t21)^3) - s2 \\
& (t21\, t31^3 + 2\, gl1\, t31^3 - 2\, t21^2\, t31^2 + 6\, gl1^2\, t31^2 + t21^2\, t31^3 - \\
& 6\, gl1\, t21^2\, t31 - 9\, gl1^2\, t21\, t31 + 4\, gl1^3\, t31 + 2\, gl1\, t21^3 - \\
& 3\, gl1^2\, t21^2 - 8\, gl1^3\, t21)/(t21^3\, (t31 - t21)^3) \\
& - \frac{s33\,(t31 - gl3)\,(t31 - t21 - gl3)\,(2\, t31 - t21 - 2\, gl3)}{t31^2\,(t31 - t21)^2} \\
& + \frac{s11\,(t21 + gl1)\,(t31 + gl1)\,(t31 + t21 + 2\, gl1)}{t21^2\, t31^2}
\end{aligned}
$$

```
   s22 (t21 - gl2) (t31 + gl1) (t31 - t21 + gl2 + gl1)
 - ----------------------------------------------------
                          2            2
                       t21  (t31 - t21)

Do you want to continue displaying a ?
                                     6
y;

a   =
 6

                                                 2                            2
  2 gl1 s1 (t21 + gl1) (t31 + gl1) (t31  + t21 t31 + gl1 t31 + t21  + gl1 t21)
- -----------------------------------------------------------------------------
                                               3    3
                                            t21  t31


                                                  2                             2
+ 2 gl1 s2 (t21 + gl1) (t31 + gl1) (t31  - 2 t21 t31 + gl1 t31 + t21

                  3            3
- 2 gl1 t21)/(t21  (t31 - t21) ) - 2 gl1 s3 (t21 + gl1) (t31 + gl1)


    2                               2                  3            3
(t31  - 2 t21 t31 - 2 gl1 t31 + t21  + gl1 t21)/(t31  (t31 - t21) )

                  2                   2                    2           2
  s33 (t31 - gl13)  (t31 - t21 - gl13)    s22 (t21 - gl2)  (t31 + gl1)
- ----------------------------------- - ------------------------------
                2            2                       2            2
             t31  (t31 - t21)                     t21  (t31 - t21)

                  2           2
  s11 (t21 + gl1)  (t31 + gl1)
- -----------------------------
                 2    2
              t21  t31
```

```
Do you want to display s11?
y;

                     y2 - 2                           y1 - 2
           s11 = g11        x2 (y2 - 1) y2 + g11            x1 (y1 - 1) y1

Do you want to display s22?
y;

                     y2 - 2                           y1 - 2
           s22 = g12        x2 (y2 - 1) y2 + g12            x1 (y1 - 1) y1

Do you want to display s33?
y;

                     y2 - 2                           y1 - 2
           s33 = g13        x2 (y2 - 1) y2 + g13            x1 (y1 - 1) y1

(d6) done
```

APPENDIX F: Listing of Automatically Generated FORTRAN Code

```
c       This is the template subroutine to calculate tensor S and D.
c       inputs are eigenvalues gl1,gl2,gl3, and cmu(6). cmu is assumed to be
c       engineering strain(e), e.g. the Cauchy-green deformation tensor
c       cm(3,3) is related to cmu(6) in the follcwing fashion:
c
c       cm(1,1)=cmu(1), cm(2,2)=cmu(2), cm(3,3)=cmu(3), cmu(4)=2*cm(1,2),
c       cm(5)=2*cm(2,3),cmu(6)=2*cm(1,3).
c
c       The outputs are the second order tensor S(6) and forth order
c       tensor D(6,6) are related in the following way:
c
c       S=D*C
c
c       S(1,1) = S(1)
c       S(2,2) = S(2)
c       S(3,3) = S(3)
c       S(1,2) = S(4)
c       S(2,3) = S(5)
c       S(3,1) = S(6)
c
c       C(1,1) = C(1)
c       C(2,2) = C(2)
c       C(3,3) = C(3)
c       C(1,2) = C(4)
c       C(2,3) = C(5)
c       C(3,1) = C(6)
c
        subroutine compsd(gl1,gl2,gl3,cmu,s,d)
        real*8 gl1,gl2,gl3,ts(3,3),td(3,3,3,3)
        real*8 delt(3,3),delt4(3,3,3,3),s(6),d(6,6)
        real*8 cmu(6),cm(3,3)
c
c       converts cmu(6) to matrix cm(3,3) in a way that
c       cm(1,2)=cm(2,1)=cmu(4),cm(2,3)=cm(3,2)=cmu(5),
c       cm(1,3)=cm(3,1)=cmu(6)
```

```
c
      do 5 i=1,3
        do 5 j=1,3
          if (i.eq.j) then
            iq=i
            cm(i,j)=cmu(iq)
          else if (i.ne.j) then
            if((i+j).eq.3) iq=4
            if((i+j).eq.4) iq=6
            if((i+j).eq.5) iq=5
            cm(i,j)=cmu(iq)/2
          end if
 5    continue
c
c     Initiates the second identity tensor delt(3,3) which
c     is a 2X2 identity matrix
c
      do 6 i=1,3
        delt(i,i)=1.0
 6    continue
c
c     Computes the forth order identity tensor delt4(3,3,3,3)
c     by definition.
c
      do 7 i=1,3
        do 7 j=1,3
          delt4(i,j,i,j)=delt(i,i)*delt(j,j)+delt(i,j)*delt(j,i)
          delt4(i,j,j,i)=delt4(i,j,i,j)
 7    continue
c
c*****************************************************************
c     For different eigenvalues gl1,gl2,gl3 the computation is
c     different.
c     case1 is gl1#gl2#gl3 call subroutine comsd1.
c     case2 is gl3=gl2#gl1 or gl1=gl3#gl2 or gl1=gl2#gl3 then
c     call subroutine compsd2.
c     case3 is gl1=gl2=gl3 call subroutine ccmpsd3.
c*****************************************************************
c
```

```
        if ((gl1.ne.gl2).and.(gl2.ne.gl3).and.(gl1.ne.gl3)) then
          call compsdl(gl1,gl2,gl3,ae t,delt4,cm,ts,td)
        else if((gl2.eq.gl3).and.(gl1.ne.gl3)) then
          call compsd2(gl1,gl2,delt,delt4,cm,ts,td)
        else if((gl1.eq.gl2).and.(gl3.ne.gl2)) then
          gl1=gl3
          call compsd2(gl1,gl2,delt,delt4,cm,ts,td)
        else if((gl1.eq.gl3).and.(gl2.ne.gl3)) then
          gl1=gl2
          gl2=gl3
          call compsd2(gl1,gl2,delt,delt4,cm,ts,td)
        else
          call compsd3(gl1,delt,delt4,ts,td)
        end if
c
c       Rewrite the tensor ts(l,j) td(i,j,k,l) to S(i) and D(i,j)
c       respectively by uslng the symetric property.
c       converts ts(3,3) s(6) and td(3,3,3,3) to D(6,6)
c
        do 8 i=1,3
          do 8 j=i,3
            if (i.eq.j) iq=i
            if (i.eq.l.and.j.eq.2) iq=4
            if (i.eq.2.and.j.eq.3) iq=5
            if (i.eq.l.and.j.eq.3) iq=6
    s(iq)=ts(i,j)
 8      continue
c
        do 9 i=1,3
          do 9 j=i,3
            d(i,j)=td(i,i,j,j)
 9      continue
c
        do 10 i=1,3
          d(i,4)=td(i,i,1,2)+td(i,i,2,1)
          d(i,5)=td(i,i,2,3)+td(i,i,3,2)
  d(i,6)=td(i,i,3,1)+td(i,i,1,3)
 10     continue
c
```

```
        d(4,4)= (td(1,2,1,2)+td(1,2,2,1)+td(2,1,1,2)+td(2,1,2,1))/2.
        d(4,5)= (td(1,2,2,3)+td(1,2,3,2)+td(2,1,2,3)+td(2,1,3,2))/2.
        d(4,6)= (td(1,2,1,3)+td(1,2,3,1)+td(2,1,1,3)+td(2,1,3,1))/2.
c
        d(5,5)=(td(2,3,2,3)+td(2,3,3,2)+td(3,2,2,3)+td(3,2,3,2))/2.
        d(5,6)=(td(2,3,1,3)+td(2,3,3,1)+td(3,2,1,3)+td(3,2,3,1))/2.
        d(6,6)=(td(3,1,1,3)+td(3,1,3,1)+td(1,3,1,3)+td(1,3,3,1))/2.
c
        do 11 i=1,6
          do 11 j=1,6
            d(i,j) = d(j,i)
 11     continue
c
c       prints out the inputs gl1,cl2,gl3,cmu(6)
c       and outputs S and D
c
        print*, ' gl1=', gl1
        print*, ' gl2=', gl2
        print*, ' gl3=', gl3
        print*, 'Input tensor C(6):'
        print*, (cmu(i), i=1,6)
        print*, "second order tensor S(6):"
        print*, (s(i), i=1,6)
        print*, "The forth order tensor D(6,6):"
        do 101 i=1,6
  print*, (d(i,j),j=1,6
 101    continue
        return
        end
c
subroutine compsd1(gl1,c312,gl3,delt,delt4,cm,ts,td)
c
        real*8 gl1,gl2,gl3,ts(3,3),td(3,3,3,3)
        real*8 cm(3,3),delt(3,3),delt4(3,3,3,3),p(3,3,3,3)
        real*8 q(3,3,3,3),cmm(3,3),p1(3,3,3,3),p21(3,3,3,3)
        real*8 p31(3,3,3,3),q11(3,3,3,3),q12(3,3,3,3),p22(3,3,3,3)
        real*8 q21(3,3,3,3),q22(3,3,3,3),a,b,c,a1,a2,a3,a4,a5,a6
```

```
c
c Obtains cmm(3,3)=cm(3,3)*cm(3,3) from subroutine product
c
        call product(cm,cmm)
c
c Uses the formula we derived in code to compute second order
c tensor ts(3,3).
c
        do 25037 i=1,3
          do 25038 j=1,3
            ts(i,j)=a(gl1,gl2,gl3)*cmm(i,j)+b(gl1,gl2,gl3)*cm(i,j)
     .      c(gl1,gl2,gl3)*delt(i,j)
25038     continue
25037   continue
c
c       call subroutine to compute the functions we defined
c       when we derived forth order tensor td, namely P(i,j,k,l)
c       and Q(l,j,k,l) which are the functions of cm(3,3) and
c       the matrix product cmm(3,3).
c
        call pqcom(cmm,cmm,p1,q)
        call pqcom(cmm,cm,p21,q)
        call pqcom(cm,cmm,p22,q)
        call pqcom(cm,cm,p31,q)
        call pqcom(cmm,delt,p,q11)
        call pqcom(delt,cmm,p,q12)
        call pqcom(cm,delt,p,q21)
        call pqcom(delt,cm,P,q22)
c
c       computes forth order tensor td(i,j,k,l)
c
        do 25039 i=1,3
          do 25040 j=1,3
            do 25041 k=1,3
              do 25042 l=1,3
                td(i,j,k,l)=a1(gl1,gl2,gl3)*p1(i,j,k,l)+
     .           a2(gl1,gl2,gl3)*(p21(i,j,k,l)+p22(i,j,k,l))+
     .           a4(gl1,gl2,gl3)*p31(i,j,k,l)+a3(gl1,gl2,gl3)*
     .           (q11(i,j,k,l)+q12(i,j,k,l))+a5(gl1,gl2,gl3)*
     .           (q21(i,j,k,l)+q22(i,j,k,l))+a6(gl1,gl2,gl3)
```

```
     .          *delt4(i,j,k,l)
25042        continue
25041      continue
25040    continue
25039   continue
c
        return
        end
c
c       a,b,c,a1-a6 are the coefficients we derived in code.
c
        real*8 function a(gl1,gl2,gl3)
        real*8 gl1,gl2,gl3,s1,s2,s3
        a=-s1(gl1,gl2,gl3)/(-gl1+gl2)/(-gl1+gl3)+s2(gl1,gl2,gl3)/
     .  (-gl1+gl2)/(-gl2+gl3)-s3(gl1,gl2,gl3)/(-gl1+gl3)/(-gl2+gl3)
        return
        end
c
        real*8 function b(gl1,gl2,gl3)
        real*8 gl1,gl2,gl3,s1,s2,s3
        b=s3(gl1,gl2,gl3)*(gl1+gl2)/(-gl1+gl3)/(-gl2+gl3)-s2(gl1,gl2,
     .  gl3)*(gl1+gl3)/(-gl1+gl2)/(-gl2+gl3)+s1(gl1,gl2,gl3)*(gl2+gl3)
     .  /(-gl1+gl2)/(-gl1+gl3)
        return
        end
c
        real*8 function c(gl1,gl2,gl3)
        real*8 gl1,gl2,gl3,s1,s2,s3
        c=-s1(gl1,gl2,gl3)*gl2*gl3/(-gl1+gl2)/(-gl1+gl3)+gl1*s2(gl1,
     .  gl2,gl3)*gl3/(-gl1+gl2)/(-gl2+gl3)-gl1*s3(gl1,gl2,gl3)*gl2/
     .  (-gl1+gl3)/(-gl2+gl3)
        return
        end
c
```

```
      real*8 function a1(gl1,gl2,gl3)
      real*8 gl1,gl2,gl3,s1,s2,s3,s11,s22,s33
c
      a1=- s11(gl1,gl2,gl3)/(-gl1+gl2)**2/(-gl1+gl3)**2+2.0*
     .      s2(gl1,gl2,gl3)*(gl1-2.0*gl2+gl3)/(-gl1+gl2)**3/
     .     (-gl2+gl3)**3-s22(gl1,gl2,gl3)/(-gl1+gl2)**2/(-gl2+gl3)**2
     .     -s33(gl1,gl2,gl3)/(-gl1 +gl3)**2/(-gl2+gl3)**2-2.0*
     .     s1(gl1,gl2,gl3)*(-2.0*gl1+gl2+gl3)/(-gl1+gl2)**3/
     .     (-gl1+gl3)**3+2.0*s3(gl1,gl2,gl3)*(-gl1-gl2+2.0*gl3)/
     .     (-gl1+gl3)**3/(-gl2+gl3)**3
      return
      end
c
      real*8 function a2(gl1,gl2,gl3)
      real*8 gl1,gl2,gl3,s1,s2,s3,s11,s22,s33
      a2=s33(gl1,gl2,gl3)*(gl1+gl2)/(-gl1+gl3)**2/(-gl2+gl3)**2+
     . s22(gl1,gl2,gl3)*(gl1+gl3)/(-gl1+gl2)**2/(-gl2+gl3)**2+
     . s11(gl1,gl2,gl3)*(gl2+gl3)/(-gl1+gl2)**2/(-gl1+gl3)**2
     . -s3(gl1,gl2,gl3)*(-2.0*gl1**2-3.0*gl1*gl2-2.0*gl2**2+
     . 3.0*gl1*gl3+3.0*gl2*gl3+gl3**2)/(-gl1+gl3)**3/(-gl2+gl3)**3
     . -s2(gl1,gl2,gl3)*(2.0*gl1**2-3.0*gl1*gl2-gl2**2+3.0*gl1*
     . gl3-3.0*gl2*gl3+2.0*gl3**2)/(-gl1+gl2)**3/(-gl2+gl3)**3+
     . s1(gl1,gl2,gl3)*(-gl1**2-3.0*gl1*gl2+2.0*gl2**2-3.0*gl1*
     . gl3+3.0*gl2*gl3+2.0*gl3**2)/(-gl1+gl2)**3/(-gl1+gl3)**3
      return
      end
c
      real*8 function a3(gl1,gl2,gl3)
      real*8 gl1,gl2,gl3,s1,s2,s3,s11,s22,s33
      a3=-s11(gl1,gl2,gl3)*gl2*gl3/(-gl1+gl2)**2/(-gl1+gl3)**2-gl1*
     . s22(gl1,gl2,gl3)*gl3/(-gl1+gl2)**2/(-gl2+gl3)**2-gl1*
     . s33(gl1,gl2,gl3)*gl2/(-gl1+gl3)**2/(-gl2+gl3)**2+
     . s2(gl1,gl2,gl3)*(gl1**2*gl2-2.0*gl1*gl2**2+gl2**3+gl1**2
     . *gl3-gl1*gl2*gl3-2.0*gl2**2*gl3+gl1*gl3**2+gl2*gl3**2)/
     . (-gl1+gl2)**3/(-gl2+gl3)**3-s1(gl1,gl2,gl3)*(gl1**3-2.0
     . *gl1**2*gl2+gl1*gl2**2-2.0*gl1**2*gl3-gl1*gl2*gl3+gl2**2*gl3
     . +gl1*gl3**2+gl2*gl3**2)/(-gl1+gl2)**3/(-gl1+gl3)**3-s3(gl1,
     . gl2,gl3)*(gl1**2*gl2+gl1*gl2**2+gl1**2*gl3-gl1*gl2*gl3+gl2**2*
     . gl3-2.0*gl1*gl3**2-2.0*gl2*gl3**2+gl3**3)/(-gl1+gl3)**3
     . /(-gl2+gl3)**3
```

```
c
      real*8 function a4(gl1,gl2,gl3)
      real*8 gl1,gl2,gl3,s1,s2,s3,s11,s22,s33
      a4=-s33(gl1,gl2,gl3)*(gl1+gl2)**2/(-gl1+gl3)**2/(-gl2+gl3)**2-
     . s22(gl1,gl2,gl3)*(gl1+gl3)**2/(-gl1+gl2)**2/(-gl2+gl3)**2
     . -s11(gl1,gl2,gl3)*(gl2+gl3)**2/
     . (-gl1+gl2)**2/(-gl1+gl3)**2+2.0*s2(gl1,gl2,gl3)*(gl1+gl3)*
     . (gl1**2-gl1*gl2-gl2**2+gl1*gl3-gl2*gl3+gl3**2)/(-gl1+
     . gl2)**3/(-gl2+gl3)**3-2.0*s1(gl1,gl2,gl3)*(gl2+gl3)*(-gl1**2-
     . gl1*gl2+gl2**2-gl1*gl3+gl2+gl3+gl3**2)/(-gl1+gl2)**3/(-gl1+gl3
     . )**3+2.0*s3(gl1,gl2,gl3)*(gl1+gl2)*(-gl1**2-gl1*gl2-gl2**2+gl1*
     . gl3+gl2*gl3+gl3**2)/(-gl1+gl3)**3/(-gl2+gl3)**3
      return
      end
c
      real*8 function a5(gl1,gl2,gl3)
      real*8 gl1,gl2,gl3,s1,s2,s3,s11,s22,s33
      a5=gl1*s33(gl1,gl2,gl3)*gl2*(gl1+gl2)/(-gl1+gl3)**2/(-gl2+gl3)
     . **2+gl1*s22(gl1,gl2,gl3)*gl3*(gl1+gl3)/(-gl1+gl2)**2/(-gl2+gl3)
     . **2+s11(gl1,gl2,gl3)*gl2*gl3*(gl2+gl3)/(-gl1+gl2)**2/(-gl1+gl3)
     . **2+s3(gl1,gl2,gl3)*(gl1**3*gl2+gl1**2*gl2**2+gl1*gl2**3+gl1**
     . 3*gl3+gl1**2*gl2*gl3+gl1*gl2**2*gl3+gl2**3*gl3-2.0*gl1**2*gl3**
     . 2-5.0*gl1*gl2*gl3**2-2.0*gl2**2*gl3**2+gl1*gl3**3+gl2*gl3**3)/
     . (-gl1+gl3)**3/(-gl2+gl3)**3-s2(gl1,gl2,gl3)*(gl1**3*gl2-2.0*gl1
     . **2*gl2**2+gl1*gl2**3+gl1**3*gl3+gl1**2*gl2*gl3-5.0*gl1*gl2**2*
     . gl3+gl2**3*gl3+gl1**2*gl3**2+gl1*gl2*gl3**2-2.0*gl2**2*gl3**2+
     . gl1*gl3**3+gl2*gl3**3)/(-gl1+gl2)**3/(-gl2+gl3)**3+s1(gl1,gl2,
     . gl3)*(gl1**3*gl2-2.0*gll**2*gl2**2+gll*gl2**3+gll**3*gl3-5.0*
     . gll**2*gl2*gl3+gll*gl2**2*gl3+gl2**3*gl3-2.0*gll**2*gl3**2+gll
     . +gl2*gl3**2+gl2**2*gl3**2+gll*gl3**3+gl2*gl3**3)/(-gll+gl2)**3/
     . (-gll+gl3)**3
      return
      end
c
```

```
      real*8 function a6(gl1,gl2,gl3)
      real*8 gl1,gl2,gl3,s1,s2,s3,s11,s22,s33
      a6=-s11(gl1,gl2,gl3)*gl2**2*gl3**2/(-gl1+gl2)**2/(-gl1+gl3)**2
     . -gl1**2*s22(gl1,gl2,gl3)*gl3**2/(-gl1+gl2)**2/(-gl2+gl3)**2-gl1
     . **2*s33(gl1,gl2,gl3)*gl2**2/(-gl1+gl3)**2/(-gl2+gl3)**2-2.0*gl1
     . *s3(gl1,gl2,gl3)*gl2*gl3*(gl1**2+gl1*gl2+gl2**2-2.0*gl1*gl3-2.0*
     . gl2*gl3+gl3**2)/(-gl1+gl3)**3/(-gl2+gl3)**3+2.0*gl1*s2(gl1,gl2,
     . gl3)*gl2*gl3*(gl1**2-2.0*gl1*gl2+gl2**2+gl1*gl3-2.0*gl2*gl3+
     . gl3**2)/(-gl1+gl2)**3/(-gl2+gl3)**3-2.0*gl1*s1(gl1,gl2,gl3)*gl2*
     . gl3*(gl1**2-2.0*gl1*gl2+gl2**2-2.0*gl1*gl3+gl2*gl3+gl3**2)/
     . (-gl1+gl2)**3/(-gl1+gl3)**3
      return
      end
c
c     The s1,s2,s3,s11,s22,s33 are derivatives of W
c
      function s1(gl1,gl2,gl3)
      common /param/ x1,x2,y1,y2
      real*8 x1,x2,y1,y2
      real *8 s1,gl1,gl2,gl3
      s1=2.0*(gl1**(-1+y1)*x1*y1+gl1**(-1+y2)*x2*y2)
      return
      end
c
      function s2(gl1,gl2,gl3)
      common /param/ x1,x2,y1,y2
      real*8 x1,x2,y1,y2
      real*8 s2,gl1,gl2,gl3
      s2=2.0*(gl2**(-1+y1)*x1*y1+gl2**(-1+y2)*x2*y2)
      return
      end
c
      function s3(gl1,gl2,gl3)
      common /param/ x1,x2,y1,y2
      real*8 x1,x2,y1,y2
      real*8 s3,gl1,gl2,gl3
      s3=2.0*(gl3**(-1+y1)*x1*y1+gl3**(-1+y2)*x2*y2)
      return
      end
c
```

```
      function s11(gl1,gl2,gl3)
      common /param/ x1,x2,y1,y2
      real*8 x1,x2,y1,y2
      real*8 s11,gl1,gl2,gl3
      s11=2.0*(gl1**(-2+y1)*x1*(-1.0+y1)*y1+gl1**(-2+y2)*x2*
     . (-1.0+y2)*y2)
      return
      end
c
      function s22 (gl1,gl2,gl3)
      common /param/ x1,x2,y1,y2
      real*8 x1,x2,y1,y2
      real*8 s22, gl1,gl2,gl3
      s22=2.0*(gl2**(-2+y1)*x1*(-1.0+y1)*y1+gl2**(-2+y2)*x2*
     . (-1.0+y2)*y2
      return
      end
c
      function s33(gl1,gl2,cl3)
      common /param/ x1,x2,y1,y2
      real*8 x1,x2,y1,y2
      real*8 s33,gl1,gl2,gl3
      s33=2.0*(gl3**(-2+y1)*x1*(-1.0+y1)*y1+gl3**(-2+y2)*x2*
     . (-1.0+y2)*y2
      return
      end
c
      subroutine compsd2(gl1,gl2,delt,delt4,cm,ts,td)
      real*8 gl1,gl2,ts(3,3),td(3,3,3,3)
      real*8 cm(3,3),delt(3,3),delt4(3,3,3,3),pl(3,3,3,3)
      real*8 q2(3,3,3,3),q1(3,3,3,3),p(3,3,3,3),q(3,3,3,3)
      real*8 b1,b2,b3, abar,bbar
```

```
c
c       Computes second order tensor ts(i,j) based on the formula
c       derived in code.
c
        do 25043 i=1,3
          do 25044 j=1,3
            ts(i,j)=abar(gl1,gl2)*cm(i,j)+bbar(gl1,gl2)*delt (i,j)
25044     continue
25043   continue
c
c       Call subroutine to get P, Q which are defined in code.
c
        call pqcom(cm,cm,p1,q)
        call pqcom(cm,delt,p,q1)
        call pqcom(delt,cm,p,q2)
c
c       Computes tensor td(i,j,k,l).
c
        do 25045 l=1,3
          do 25046 j=1,3
            do 25047 k=1,3
              do 25048 1=1,3
                td(i,j,k,l)=b1(gl1,gl2)*p1(i,j,k,l)+b2(gl1,gl2)*
     .          (q1(i,j,k,l)+q2(i,j,k,l))+b3(gl1,gl2)*delt4(i,j,k,l)
25048         continue
25047       continue
25046     continue
25045   continue
        return
        end
c
c       abar,bbar are b1, b2, b3 functions derived in code.
c
        real*8 function abar(gl1,gl2)
        real8 gl1,gl2,ss1,ss2
        abar=(-ss1(gl1,gl2)+ss2(gl1,gl2))/(-gl1+gl2)
        return
        end
c
```

```
      real*8 function bbar(gl1,gl2)
      real*8 gl1,gl2,ss1,ss2
      bbar=(-gl1*ss2(gl1,gl2)+ss1(gl1,gl2)*gl2)/(-gl1+gl2)
      return
      end
c
      real*8 function b1(gl1,gl2)
      real*8 gl1,gl2,ss1,ss2,ss11,ss22
      b1=2.0*ss1(gl1,gl2)/(-gl1+gl2)**3-2.0*ss2(gl1,gl2)/(-gl1+gl2)
     . **3+ss11(gl1,gl2)/(-gl1+gl2)**2+ss22(gl1,gl2)/(-gl1+gl2)**2
      return
      end
c
      real*8 function b2(gl1,gl2)
      real*8 gl1,gl2,ss1,ss2,ss11,ss22
      b2=-gl1*ss22(gl1,gl2)/(-gl1+gl2)**2-ss11(gl1,gl2)*gl2/
     . (-gl1+gl2)**2-ss1(gl1,gl2)*(gl1+gl2)/(-gl1+gl2)**3+ss2(gl1,
     . gl2)*(gl1+gl2)/(-gl1+gl2)**3
      return
      end
c
      real*8 function b3(gl1,gl2)
      real*8 gl1,gl2,ss1,ss2,ss11,ss22
      b3=2.0*gl1*ss1(gl1,gl2)*gl2/(-gl1+gl2)**3-2.0*gl1*ss2(gl1,gl2)
     . *gl2/(-gl1+gl2)**3+gl1**2*ss22(gl1,gl2)/(-gl1+gl2)**2+ss11(gl1,
     . gl2)*gl2**2/(-gl1+gl2)**2
      return
      end
c
c     ss1,ss2,ss11,ss22 are derivatives of W.
c
      function ss1(gl1,gl2)
      common /param/ x1,x2,y1,y2
      real*8 x1,x2,y1,y2
      real*8 ss1,gl1,gl2
      ss1=2.0*(gl1**(-1+y1)*x1*y1+gl1**(-1+y2)*x2*y2)
      return
      end
c
```

```
      function ss2(gl1,gl2)
      common /param/ x1,x2,y1,y2
      real*8 x1,x2,y1,y2
      real ss2,gl1,gl2
      ss2=2.0*(2.0*gl2**(-1+y1)*x1*y1+2.0*gl2**(-1+y2)*x2*y2)
      return
      end
c
      function ss11(gl1,gl2)
      common /param/ x1,x2,y1,y2
      real*8 x1,x2,y1,y2
      real*8 ss11,gl1,gl2
      ss11=2.0*(gl1**(-2+y1)*x1*(-1.0+y1)*y1+gl1**(-2+y2)*x2*
     .(-1.0+y2)
      return
      end
c
      function ss22(gl1,gl2)
      common /param/ x1,x2,y1,y2
      real*8 x1,x2,y1,y2
      real*8 ss22,gl1,gl2
      ss22=2.0*(2.0*gl2**(-2+y1)*x1*(-1.0+y1)*y1+2.0*gl2**
     .(-2+y2)*x2*(-1.0+y2)*y2)
      return
      end
c
      subroutine compsd3(gl1,delt,delt4,ts,td)
      real*8 gl1,ts(3,3),td(3,3,3,3),delt(3,3),delt4(3,3,3,3)
      real*8 cc1,abbar
c
      do 25049 i=1,3
        do 25050 j=1,3
          ts(i,j)=abbar (gl1)*delt(i,j)
25050   continue
25049 continue
```

```
      do 25051 i=1,3
        do 25052 j=1,3
          do 25053 k=1,3
             do 25054 l=1,3
               td(i,j,k,l)=cc1(gl1)*delt4(i,j,k,l)
25054        continue
25053     continue
25052   continue
25051 continue
c
      return
      end
c
      real*8 function abbar(gl1)
      real*8 gl1
      abbar=sss1(gl1)
      return
      end
c
      real*8 function cc1(gl1)
      real*8 gl1
      ccl=sss11(gl1)
      return
      end
c
      function sss1(gl1)
      common /param/ x1,x2,y1,y2
      real*8 x1,x2,y1,y2
      real*8 sss1,gl1
      sss1=2.0*(3.0*gl1**(-1+y1)*x1*y1+3.0*gl1**(-1+y2)*x2*y2)
      return
      end
c
```

```
      function sss11(gl1)
      common /param/ x1,x2,y1,y2
      real*8 x1,x2,y1,y2
      real*8 sss11,gl1
      sss11=2.0*(3.0*gl1**(-2+y1)*x1*(-1.0+y1)*y1+3.0*gl1**
     . (-2+y2)*x2*(-1.0+y2)*y2)
      return
      end
c
c     This subroutine computes P and Q forth order tensors
c     which we define in tensor D
c
      subroutine pqcom(cm1,cm2,p,q)
      real*8 cm1(3,3),cm2(3,3),p(3,3,3,3),q(3,3,3,3)
      do 100 i=1,3
        do 100 j=1,3
          do 100 k=1,3
            do 100 l=1,3
              p(i,j,k,l)=cm1(i,k)*cm2(j,l)+cm1(i,l)*cm2(j,k)
     .        q(i,j,k,l)=p(i,j,k,l)+cm1(j,l)*cm2(i,k)+cm1(j,k)
     .        *cm2(i,l)
 100  continue
      return
      end
c
c     This subroutine computes matrix product cmXcm.
c
      subroutine product(mat1,cmm)
      real*8 mat1(3,3),cmm(3,3),sum
      do 25 i=1,3
        do 25 j=1,3
          sum=0.0
          do 26 k=1,3
            sum=sum+mat1(i,k)*mat1(k,j)
          continue
          cmm(i,j)=sum
 25   continue
      return
      end
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
May 1993

3. REPORT TYPE AND DATES COVERED
Technical Memorandum

4. TITLE AND SUBTITLE
Explicit Robust Schemes for Implementation of a Class of Principal Value-Based Constitutive Models: Symbolic and Numeric Implementation

5. FUNDING NUMBERS
WU-510-01-50

6. AUTHOR(S)
S.M. Arnold, A.F. Saleeb, H.Q. Tan, and Y. Zhang

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135–3191

8. PERFORMING ORGANIZATION REPORT NUMBER
E–7788

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, D.C. 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA TM–106124

11. SUPPLEMENTARY NOTES
S.M. Arnold, NASA Lewis Research Center; A.F. Saleeb, University of Akron, Department of Civil Engineering, Akron, Ohio 44325; H.Q. Tan and Y. Zhang, University of Akron, Department of Mechanical Sciences, Akron, Ohio 44325. Responsible person, S.M. Arnold, (216) 433–3334.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category ~~49~~ 59

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

The issue of developing effective and robust schemes to implement a class of the Ogden-type hyperelastic constitutive models is addressed. To this end, special purpose functions (running under MACSYMA) are developed for the symbolic derivation, evaluation, and automatic FORTRAN code generation of explicit expressions for the corresponding stress function and material tangent stiffness tensors. These explicit forms are valid over the entire deformation range, since the singularities resulting from repeated principal-stretch values have been theoretically removed. The required computational algorithms are outlined, and the resulting FORTRAN computer code is presented.

14. SUBJECT TERMS
Hyperelastic; Constitutive models; Symbolic computation; Principal value

15. NUMBER OF PAGES
55

16. PRICE CODE
A04

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102