UC Berkeley
SEMM Reports Series

Title
A three dimensional finite deformation viscoelastic model accounting for damage effects

Permalink
https://escholarship.org/uc/item/4cj3f6vj

Authors
Simo, Juan
Taylor, Robert

Publication Date
1985-03-01

eScholarship.org Powered by the California Digital Library
University of California

500
C23
85-02b

REPORT NO.
UCB/SESM-85/02b

STRUCTURAL ENGINEERING AND
STRUCTURAL MECHANICS

A THREE DIMENSIONAL FINITE DEFORMATION VISCOELASTIC MODEL ACCOUNTING FOR DAMAGE EFFECTS.

by

JUAN C. SIMO

and

ROBERT L. TAYLOR

EARTHQUAKE ENG. RES. CTR. LIBRARY
Univ. of Calif. - 453 R.F.S.
1301 So. 46th St.
Richmond, CA 94804-4698 USA
(510) 231-9403

MARCH 1985

DEPARTMENT OF CIVIL ENGINEERING
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA

A THREE-DIMENSIONAL FINITE DEFORMATION VISCOELASTIC MODEL ACCOUNTING FOR DAMAGE EFFECTS

J. C. Simo and R. L. Taylor

Department of Civil Engineering
University of California, Berkeley

March, 1985

EARTHQUAKE ENG. RES. CTR. LIBRARY
Univ. of Calif. - 453 R.F.S.
1301 So. 46th St.
Richmond, CA 94804-4698 USA
(510) 231-9403

CONTENTS

ABSTRACT.....2

INTRODUCTION.....3

NONLINEAR VISCOELASTIC RESPONSE. BASIC FORMULATION.....3

DEVELOPMENT OF A NONLINEAR DAMAGE MODEL.....5

Evolution of Damage.....7

INTEGRATION ALGORITHMS.....9

Algorithm for Viscoelastic Model Without Damage.....10

Consistent Tangent Moduli.....13

Extension of the Algorithm for Damage.....14

NUMERICAL EXPERIMENTS.....15

REFERENCES.....18

APPENDIX: Computer Listings.....25

A THREE-DIMENSIONAL FINITE DEFORMATION VISCOELASTIC MODEL ACCOUNTING FOR DAMAGE EFFECTS

ABSTRACT

In this paper, a fully three-dimensional finite deformation isotropic viscoelastic model is developed with the following characteristics.

(a) Uncoupled bulk response over any range of deformations.

(b) Generalization of the classical standard solid of the linearized theory.

(c) Both for relaxation times either extremely small or very large, the model exactly reduces to the classical Mooney-Rivlin model of rubber elasticity.

In addition, a damage mechanism is incorporated into the model of the type previously developed by the authors in the context of the linearized theory. This is a three-dimensional isotropic damage mechanism based on the maximum value attained by a scalar function of certain deviatoric Lagrangian strain measure.

The resulting 5-parameter finite deformation damage model has been implemented in the finite element computer program FEAP. The coding architecture also is compatible with the NIKE2D codes currently in use at LLNL.

INTRODUCTION

A simple three-dimensional fully nonlinear viscoelastic model is developed with the distinct characteristic of exactly reducing to the classical Neo-Hookean model of rubber elasticity in the elastic limits of very large or very small relaxations times. That is, for very fast and very low processes. In addition, volumetric and deviatoric response are completely uncoupled throughout the entire range of deformations. In a homogeneous simple shear test, for instance, the present model will predict zero hydrostatic pressure for arbitrary amounts of shearing. This in contrast with "second order" viscoelastic extensions of the Neo-Hookean model, such as the formulation proposed in [4].

A damage mechanism is incorporated into this nonlinear viscoelastic model, by extending to the fully nonlinear theory a damage mechanism previously proposed by the authors in the context of linearized kinematics [1]. This damage mechanism involves a von Mises type damage surface in strain space. The surface initially reduces to a point, and expands in an isotropic fashion according to a damage variable defined as the maximum attained by a positive scalar function of the volume preserving Lagrangian strain. The motivation for this choice of damage variable lies in the behavior of some highly filled polymers. It has been frequently noted that in a cyclic shear test the storage and loss moduli experience a degradation which, to a large extent, is independent of the number of cycles, but depends crucially on the maximum shear excursion the specimen is subject to. We refer to [1] for a background motivation.

The resulting five parameter nonlinear viscoelastic damage model proposed here is ideally suited for large scale computations. To enhance the computational applicability of the model, simple and efficient integration algorithms also are developed.

Although time constraints and lack of experimental data have prevented the authors from performing significant comparative analyses, it is hoped that further experimental work at LLNL in conjunction with extensive exercise of the present model will provide the basis for an extensive evaluation and shed light on future lines of research.

1. NONLINEAR VISCOELASTIC RESPONSE. BASIC FORMULATION.

In this section we develop the basic structure of the 3-dimensional viscoelastic finite deformation model advocated in the present work. A damage mechanism will be added to this basic structure in Section 2.

First, to introduce the conditions of uncoupled pressure/deviator response and elastic response for volumetric deformations we proceed as follows.

Let $\underset{\sim}{x}=\phi(\underset{\sim}{X},t)$ be the motion and $\underset{\sim}{F}(\underset{\sim}{X},t)$ the deformation gradient. We define a volume preserving right Cauchy-Green tensor by the expression

$$\overline{\underset{\sim}{C}} =: J^{-2/3}\underset{\sim}{C} \; ; \; \det \overline{\underset{\sim}{C}} = 1 \tag{1.1}$$

where

$$\underset{\sim}{C} = \underset{\sim}{F}^t\underset{\sim}{F} \quad ; \quad J = \det \underset{\sim}{F} \tag{1.2}$$

Further, we introduce a relaxation function $\mu(s) : [0,\infty) \rightarrow \mathbb{R}^+$ defined by

$$\mu(s) = G_\infty + [G_0 - G_\infty]e^{-s/\nu} \tag{1.3}$$

Clearly, $\mu(s)$ is the relaxation function corresponding to the standard solid model. More general relaxation functions may be considered by introducing the standard series representation in terms of exponentials of the form (1.3).

Next, consider the polar decomposition of the deformation grandient at current time:

$$\underset{\sim}{F} = \underset{\sim}{R}\,\underset{\sim}{U} \; ; \; \underset{\sim}{R}^t\underset{\sim}{R} = \underset{\sim}{1} \tag{1.5}$$

Denoting by $\underset{\sim}{\tau}$ the Kirchhoff stress tensor, we define the rotated Kirchhoff stress tensor as:

$$\underset{\sim}{\Sigma} =: \underset{\sim}{R}^t\,\underset{\sim\sim}{\tau}\underset{\sim}{R} \tag{1.6}$$

Our basic constitutive model may then be expressed as a convolution integral

of the form.

$$\underset{\sim}{\Sigma}_t = K \ln J \, \underset{\sim}{G} + \int_0^t \mu(t-s) \operatorname{dev} \dot{\overline{\underset{\sim}{C}}}(s) ds \tag{1.7}$$

where dev $\overline{\underset{\sim}{C}}(s)$ is defined as:

$$\operatorname{dev} \overline{\underset{\sim}{C}}(s) : \overline{\underset{\sim}{C}}(s) - \frac{1}{3} [\operatorname{tr} \overline{\underset{\sim}{C}}(s)] \underset{\sim}{G} \tag{1.8}$$

and $\underset{\sim}{G}$ is the metric tensor in the reference configuration.

Remark 1.1: Consider the limiting cases of (1.7) for which the relaxation time tends to either zero or ∞ in (1.3). We have the limiting values:

(a) $\nu \to \infty \Rightarrow \mu(s) \to G_0$

(b) $\nu \to 0 \Rightarrow \mu(s) \to G_\infty$

Thus, in both cases, $\mu(s) \equiv$ constant. Constitutive equation (1.7) then reduces to:

$$\underset{\sim}{\Sigma} = K \ln J \, \underset{\sim}{G} + \mu \operatorname{dev} \overline{\underset{\sim}{C}} \tag{1.9}$$

Consider now the case in which $K \to \infty$ (incompressibility). Then $J \to 1$ and $K \ln J \to P$ (hydrostatic pressure). Equation (1.9) then reduces to:

$$\underset{\sim}{\Sigma} = P \, \underset{\sim}{G} + \mu \, \underset{\sim}{C} \tag{1.10a}$$

or, since $\underset{\sim}{\tau} = \underset{\sim}{R} \, \underset{\sim}{\Sigma} \, \underset{\sim}{R}^t$ and $\underset{\sim}{b} \equiv \underset{\sim\sim}{F} \underset{\sim\sim}{F}^t \equiv \underset{\sim\sim\sim}{R} \underset{\sim\sim\sim}{C} \underset{\sim\sim\sim}{R}^t$, an equivalent statement of Eq. (1.10a) is

$$\underset{\sim}{\tau} = \tilde{p} \underset{\sim}{g} + \mu \underset{\sim}{b} \quad \text{(Neo-Hookean)} \tag{1.10b}$$

where $\underset{\sim}{g}$ is the metric tensor in the current configuration.

It follows from this discussion that the elastic limit of model (1.7) for very slow or very fast processes is the Neo-Hookean model. □

Remark 1.2: Model (1.7) should be compared with that proposed by Key [2]. The latter involves the rotated rate of deformation tensor instead of dev $\overline{C}(s)$ as proposed here. Regarding this choice it should be noted that:

(a) Key's model does not reduce to the Neo-Hookean model in the elastic limit, as opposed to (1.7).

(b) Computationally, Key's model requires an integration procedure base on the mid-point configuration (Hughes & Winget [5]). The one advocated here, on the other hand, does not require such an integration procedure. Thus, it is computationally more convenient. □

2. DEVELOPMENT OF A NONLINEAR DAMAGE MODEL

As in the linear kinematics version of the damage model developed by the authors, the proposed damage model extended to the fully nonlinear range relies upon the following physical assumptions:

1. The material is assumed to be isotropic in its virgin and in any of its damaged states.

2. The volumetric response of the material is assumed to be elastic, i.e., viscoelastic effects in bulk response are not considered.

3. The amount of damage the material is subjected to at any state is independent of the history of hydrostatic pressures; i.e. no damage in the material takes place as a result of hydrostatic loading.

4. The amount of damage is characterized by a scalar variable which depends on the maximum value attained by a scalar function of the volume preserving Lagrangian strain $\overline{C}$, defined by (1.1), during the loading history.

To define the scalar damage variable, we introduce the following standard inner product and associated norm for rank 2 tensors:

$$\underset{\sim}{A} : \underset{\sim}{B} \equiv \text{tr}\,(\underset{\sim\sim}{A}\underset{\sim}{B}^{T}) = A^{IJ}\,B_{JI}\ ;\ ||\underset{\sim}{A}|| = (\underset{\sim}{A} : \underset{\sim}{A})^{1/2} \tag{2.1}$$

We now consider the history of volume preserving Lagrangian strains $\overline{C}(s)$, where $\overline{\underset{\sim}{C}}$ is defined by Eqs. (1.1) and (1.2). In particular, we focus on the deviatoric part, i.e.

$$H_D = \{S \to \text{dev}\ \overline{\underset{\sim}{C}}(s)\ |\ \text{where}\ s\varepsilon(-\infty,t]\} \tag{2.2}$$

Here, dev $\overline{\underset{\sim}{C}}(s)$ is defined according to Eq. (1.8). In accordance with assumption (4) we define the scalar damage variable ψ_t as the function of the strain history H_D given by

$$\psi_t = \max_{s\varepsilon(-\infty,t]} ||\text{dev}\ \overline{\underset{\sim}{C}}(s)|| \tag{2.3}$$

In physical terms, ψ_t is simply the maximum value attained by the norm of dev $\overline{\underset{\sim}{C}}(s)$ during the past history. We shall denote by D_{ψ_t} the set of elements in deviatoric strain space for which the maximum in Eq. (2.3) is attained:

$$D_{\psi_t} \equiv \{\ \text{dev}\ \overline{\underset{\sim M}{C}} = \text{dev}\ \overline{\underset{\sim}{C}}(t_M)\ \Big|\ ||\text{dev}\ \overline{\underset{\sim M}{C}}|| = \psi_t\ \} \tag{2.4}$$

According to assumption (2) the volumetric response of the material is elastic. Hence, denoting by $\underset{\sim}{\sum}(t)$ the rotated stress tensor at time t, defined by Eq. (1.6), we introduce the following decomposition

$$\underset{\sim}{\sum}(t) = \frac{1}{3}\,(\text{tr}\ \underset{\sim}{\sum}(t))\ \underset{\sim}{G} + \text{dev}\ \underset{\sim}{\sum}(t) \tag{2.5}$$

where viscoelastic effects and, as a result of assumption (3), damage effects are included only in the response function associated with the stress deviator dev $\underset{\sim}{\sum}(t)$. To account for these effects, we follow the basic idea introduced

in our previous work [1], and modify the basic model (1.7) by considering the following convolution representation

$$\underset{\sim}{\Sigma}(t) = K \ln J \, \underset{\sim}{G} + \int_0^t \mu(t-s) \, \dot{\underset{\sim}{\pi}}(s) \, ds \tag{2.6}$$

where $\underset{\sim}{\pi}(s)$ is a deviatoric "effective" Lagrangian strain defined as follows.

$$\underset{\sim}{\pi}(s) = \overline{f}(\psi_s) \, \text{dev} \, \underset{\sim}{\overline{C}}(s) \tag{2.7}$$

Here, $\overline{f}(\psi_s)$ is a given function of the damage variable ψ_s defined by Eq. (2.3). In order to specify the evolution of $\dot{\underset{\sim}{\Pi}}(s)$ entering Eq. (3.6), the evolution of the damage variable ψ_t needs to be obtained.

2.1 Evolution of Damage

To specify the evolution of $\partial\psi/ds$ we introduce the following damage criterion formulated in strain space as:

$$\phi \equiv || \, \text{dev} \, \underset{\sim}{\overline{C}}(s) \, || - \psi_s \leq 0 \tag{2.8}$$

Next, we note that the following possibilities may occur during the loading process. Introducing the notation:

$$\underset{\sim}{N} \equiv \text{dev} \, \underset{\sim}{\overline{C}}(s) \, / \, || \, \text{dev} \, \underset{\sim}{\overline{C}}(s) \, || \tag{2.9}$$

we may have

$$\begin{cases} \phi < 0 & \text{(2.10a)} \\ \phi = 0 \text{ and } \text{dev} \, \dot{\underset{\sim}{C}} : \underset{\sim}{N} \begin{cases} < 0 & \text{(2.10b)} \\ = 0 & \text{(2.10c)} \\ > 0 & \text{(2.10d)} \end{cases} \end{cases}$$

Borrowing a terminology typically found in plasticity, we say that (b), (c), and (d) correspond to unloading, neutral loading, and loading, respectively. During loading, enforcement of the consistency condition yields

$$\dot{\phi} \equiv \underset{\sim}{N} : \text{dev} \, \dot{\underset{\sim}{C}}(s) - \dot{\psi}_s = 0 \tag{2.11}$$

Therefore, one has the following evolution equation for $\dot{\psi}_s$:

$$\frac{d\psi}{ds} = \begin{cases} \underset{\sim}{N} : \text{dev } \dot{\underset{\sim}{C}} & \text{if } \phi = 0 \text{ and } \underset{\sim}{N} : \text{dev } \dot{\underset{\sim}{C}} \geq 0 \\ \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

It should be noted that this (irreversible) evolution equation is consistent with definition (2.3) of the damage variable ψ_s. Indeed (2.3) is simply the integral curve of (2.11) subjected to the initial condition $\psi_s|_{s\to-\infty} = 0$.

It now follows from (2.7) and (2.12) that the evolution equation for $\underset{\sim}{\pi}(s)$ is given by:

$$\frac{d}{ds}\underset{\sim}{\Pi}(s) = \begin{cases} [\bar{f}\,\underset{\sim}{1} + \dfrac{f'}{||\text{dev}\underset{\sim}{C}||}\,\underset{\sim}{N}\otimes\underset{\sim}{N}] : \text{dev } \dot{\underset{\sim}{C}} & \text{if } \begin{cases} \phi = 0 \\ \text{and} \\ \underset{\sim}{N} : \text{dev } \underset{\sim}{C} > 0 \end{cases} \\ \\ \bar{f}\,\underset{\sim}{1} : \text{dev } \dot{\underset{\sim}{C}} & \text{otherwise} \end{cases} \tag{2.13}$$

This completes the basic formulation of our nonlinear viscoelastic damage model. It remains to address the selection of the function $\bar{f}(\psi_t)$ entering in definition (2.7) of the effective strain. Such a determination should be made on the basis of available experimental data. In what follows, we shall adopt the simple choice proposed in our previous work [1] in the context of linear kinematics. Accordingly, we set:

$$\bar{f}(x) \equiv \beta + (1-\beta)\,\frac{1-e^{-x/\alpha}}{x/\alpha} \tag{2.14}$$

where

$$\beta\varepsilon[0,1] \quad \text{and} \quad \alpha\varepsilon[0,\infty) \tag{2.15}$$

are given parameters. The resulting model is thus a five parameter model. The physical motivation for the choice expressed by (2.14) and (2.15) is discussed in [1].

Remark 2.1 Obviously, alternative definitions of damage variable and damage criterion are possible. As an example, consider a characterization of the damage mechanism based upon selecting the following damage criterion:

$$\phi \equiv \Psi_D - \psi_t \leq 0, \tag{2.16}$$

where Ψ_D is defined as follows:

$$\Psi_D \equiv \frac{1}{2} J^{-2/3} \underset{\sim}{C} : \underset{\sim}{G} \tag{2.17}$$

Enforcement of the consistency condition yields:

$$\dot{\psi}_t = \frac{1}{\Psi_t} \operatorname{dev} \overline{\underset{\sim}{C}} : \underset{\sim}{D} \qquad (\phi = 0 \text{ and } \dot{\phi} = 0) \tag{2.18}$$

where $\underset{\sim}{D} \equiv \underset{\sim}{R}^t \underset{\sim}{d} \underset{\sim}{R}$ is the rotated rate of deformation tensor. Then, one has (during loading)

$$\dot{\underset{\sim}{\Pi}}_t = \overline{f}(\psi_t) \operatorname{dev} \dot{\overline{\underset{\sim}{C}}} + \overline{f}' (\operatorname{dev} \overline{\underset{\sim}{C}} \otimes \operatorname{dev} \overline{\underset{\sim}{C}}) : \underset{\sim}{D} \tag{2.19}$$

Notice that in the present situation, ψ_t is defined as:

$$\psi_t := \max_{s \in (-\infty, t]} \{\frac{1}{2} \operatorname{tr} \overline{\underset{\sim}{C}}(s)\} \tag{2.20}$$

It should also be noted that for very slow or very fast processes, ψ_D defined by (2.17) becomes the distortional part of the elastic stored energy function, since

$$\mu \underset{\sim}{F} \frac{\partial \psi_D}{\partial \underset{\sim}{C}} \underset{\sim}{F}^T \equiv \mu \operatorname{dev} \overline{\underset{\sim}{b}} = \operatorname{dev} \underset{\sim}{\tau} \tag{2.21}$$

where $\underset{\sim}{\tau}$ is the Kirchhoff stress tensor, and $\mu > 0$ the shear modulus. Hence, the damage criterion (2.16) becomes a condition on the maximum of the distortional part of the stored energy function. □

3. INTEGRATION ALGORITHMS

In this section we consider the development of algorithms for the numerical integration of the proposed constitutive models in the context of the finite element method. First, we consider the algorithmic treatment of the viscoelastic model formulated in Section 1. The algorithmic treatment of the damage mechanism will be discussed subsequently.

3.1 Algorithm for Viscoelastic Model Without Damage

The algorithm treatment discussed below is inspired in the formulation advocated in [2]. Define the following history variable

$$\underset{\sim}{H}_t = \int_{-\infty}^{t} [G_0 - G_\infty] e^{-(t-s)/\nu} \text{dev}[\dot{\overline{\underset{\sim}{C}}}(s)] ds \tag{3.1}$$

We now introduce the following configuration update procedure. Let $\underset{\sim}{x} = \phi_n(\underset{\sim}{x},t)$ be the configuration at $t=t_n$, and $\overline{\underset{\sim}{u}}(\underset{\sim}{X})$ and incremental displacement field. We set

$$\begin{aligned} \phi_{n+1} &= \phi_n + \underset{\sim}{U} \equiv \phi_n + \underset{\sim}{u} \circ \phi_n \\ \underset{\sim}{F}_{n+1} &= F_n + \text{GRAD}\, \underset{\sim}{U} \equiv [1 + \nabla_n \underset{\sim}{U}]\, \underset{\sim}{F}_n \end{aligned} \tag{3.2}$$

where $\underset{\sim}{F}_n$ and $\underset{\sim}{F}_{n+1}$ are the deformation gradients at t_n and t_{n+1}, and we have set $\underset{\sim}{U}(\underset{\sim}{X},t) = \underset{\sim}{u}(\phi_n(\underset{\sim}{X},t))$. Since $\underset{\sim}{C}_{n+1} = \underset{\sim}{F}^t_{n+1}\, \underset{\sim}{F}_{n+1}$, making use of (2.1) and (2.2), it follows that

$$\begin{aligned} \underset{\sim}{\sum}_{n+1} &= K \ln J_{n+1}\, \underset{\sim}{G} + \text{dev}\, [\underset{\sim}{\sum}_{n+1}] \\ &\quad + [e^{-\Delta t/\nu}\, 1]\, \underset{\sim}{H}_n \\ &\quad + \overline{\mu}(\Delta t/\nu)\, \text{dev}\, [\Delta \overline{\underset{\sim}{C}}] \end{aligned} \tag{3.3}$$

Where:

$$\overline{\mu}(\Delta t/\nu) \equiv \left[G_\infty + (G_0 - G_\infty) \frac{1 - e^{-\Delta t/\nu}}{\Delta t/\nu} \right] \overline{f} \tag{3.4}$$

$$\Delta \overline{\underset{\sim}{C}} \equiv \overline{\underset{\sim}{C}}_{n+1} - \overline{\underset{\sim}{C}}_n$$

Thus, we may proceed as indicated in the table below.

TABLE 1

Kinematics

$$\phi_{n+1} = \phi_n + \underset{\sim}{U}_n$$

$$\underset{\sim}{F}_{n+1} = \underset{\sim}{F}_n + \underset{\sim}{GRAD}\ U_n$$

$$\underline{C}_{n+1} = \underline{F}_{n+1}^T \underline{F}_{n+1}$$

$$\underset{\sim}{R}_{n+1} = \underset{\sim}{F}_{n+1}\ \underline{U}_{n+1}^{-1} \quad \text{(Polar decomposition)}$$

$$J_{n+1} = (\det \underline{F}_{n+1})\ \sqrt{\det\, g/\det\, \underline{G}}$$

$$\underline{C}_{n+1} = J_{n+1}^{-2/3}\ \underline{C}_{n+1}$$

Stress Calculation (No Damage Mechanism)

$$\overline{P}_{n+1} = K \ln J_{n+1}$$

$$\underset{\sim}{H}_{n+1} = e^{-\Delta t/t}\ \underset{\sim}{H}_n + [\mu(\Delta t/\nu) - G_\infty]\ \mathrm{dev}[\tilde{\underline{\Delta C}}]$$

$$\mathrm{dev}\ \underset{\sim}{\sum}_{n+1} = \mathrm{dev}\ \underset{\sim}{\sum}_n + [\underset{\sim}{H}_{n+1} - \underset{\sim}{H}_n] + G_\infty\ \mathrm{dev}\ [\tilde{\underline{\Delta C}}]$$

$$\underset{\sim}{\tau}_{n+1} = \underset{\sim}{R}_{n+1}(\mathrm{dev}\ \underset{\sim}{\sum}_{n+1})\ \underset{\sim}{R}_{n+1}^T + \overline{P}_{n+1}\ \underset{\sim}{g}$$

Remark 3.1 Notice that the stress calculation is performed in the rotated configuration. □

Remark 3.2 The symbols $\underset{\sim}{g}$ and $\underset{\sim}{G}$ stand for the metric tensor in the current and reference configurations. Notice that the Jacobian J is given by the formula

$$J = \det \underset{\sim}{F}\ \sqrt{\det\, \underset{\sim}{g}}\ /\ \sqrt{\det\, \underset{\sim}{G}}$$

As an example, for axisymmetric problems one has

$$\underset{\sim}{F} = \begin{bmatrix} \frac{\partial \upsilon_1}{\partial R} & \frac{\partial r}{\partial Z} & 0 \\ \frac{\partial z}{\partial R} & \frac{\partial z}{\partial Z} & 0 \\ 0 & 0 & 1 \end{bmatrix} , \tag{3.5}$$

and Jacobian is defined as

$$J = (\det \underset{\sim}{F})\ \upsilon/R \qquad \square \tag{3.6}$$

3.2 Consistent Tangent Moduli:

It remains to compute the tangent moduli consistent with the above algorithm. As a result of the stress calculation we have:

$$\underset{\sim}{\sum}_{n+1} = K \ln J_{n+1} \underline{G} + \bar{\mu}(\Delta t/\nu)\,\mathrm{dev}\ \underset{\sim}{\bar{C}}_{n+1} + \underset{\sim}{\sum}{}^{*}_{n} \tag{3.7}$$

where $\sum^{*}_{n}$ denotes the effect of the stresses at time t_n (see (3.3)). Following Simo and Marsden [3] we have:

$$J\underset{\sim}{A}_{n+1} \equiv 4\rho_0 \frac{\partial^2 \psi}{\partial G^2} = 2\frac{\partial \underset{\sim}{\sum}_{n+1}}{\partial \underset{\sim}{G}} \tag{3.8}$$

If we recall that:

$$\frac{\partial J}{\partial \underset{\sim}{G}} = \frac{1}{2} J\, \underline{G}^{-1} \ ; \ \frac{\partial \underset{\sim}{C}}{\partial \underset{\sim}{G}} = \underline{0} \tag{3.9}$$

We obtain the following expression for the rotated moduli:

$$\begin{aligned} J\underset{\sim}{A} = {} & K\, \underset{\sim}{G}^{-1} \otimes \underset{\sim}{G}^{-1} + 2\bar{\mu}\,(\Delta t/\nu)\,[\underline{1} - \tfrac{1}{3}\, \underset{\sim}{G}^{-1} \otimes \underset{\sim}{G}^{-1}] - 2K \ln J\, \underline{1} \\ & - \frac{4}{3}\,\mu(\Delta t/\nu)\,[(\mathrm{dev}\ \underset{\sim}{\bar{C}}) \otimes \underset{\sim}{G}^{-1}]^{S} \end{aligned} \tag{3.10}$$

where $(\ )^S$ is the symmetric part of the argument. Transforming the rotated moduli (2.8) to the current configuration we obtain:

$$\underline{\underline{Jc}} \equiv K\, \underline{\underline{g}} \otimes \underline{\underline{g}} + 2\mu\, (\Delta t/t)\, \frac{\overline{I}}{3} \left(\underline{\underline{1}} - \frac{1}{3}\, \underline{\underline{g}} \otimes \underline{\underline{g}}\right) - 2K \ln J\, \underline{\underline{1}}$$

$$-\frac{4}{3}\, \overline{\mu}(\Delta t/\nu)\, [(\mathrm{dev}\ \underline{\underline{b}}) \otimes \underline{\underline{g}}]^S \tag{3.11}$$

where it is understood that both (3.10) and (3.11) are evaluated at time $t=t_{n+1}$. In expression (2.a) the following notation has been employed:

$$\overline{\underline{\underline{b}}} = \overline{J}^{2/3}\, \underline{\underline{F}}\, \underline{\underline{F}}^T \qquad \text{and} \qquad \overline{I} = \mathrm{tr}\, \overline{\underline{\underline{b}}} \tag{3.12}$$

In Eqs. (3.10) and (3.11), $\underline{\underline{1}}$ denotes the fourth order symmetric tensor.

Remark 3.3. The effect of the stress $\sum_n^*$ in the calculation of the tangent stiffness has been neglected. Thus, (3.11) furnishes only an approximation. Next, we consider the damage model developed in Section 2.

3.3 Extension of the Algorithm for Damage

The extension of the algorithm developed in Section 3.1 to the damage model discussed in Section 2 is summarized below.

Algorithm:

Let:

$$\underline{\underline{H}}_n =: \int_0^{t_n} e^{-(t_n - s)/\nu}\, \dot{\overline{\mathrm{dev}[\Pi](s)}}\, ds$$

$$\underline{\underline{\sum}}_n =: G_\infty \int_0^{tn} \dot{\overline{\mathrm{dev}[\underline{\underline{\Pi}}]}}\, ds + [G_0 - G_\infty]\underline{\underline{H}}_n \tag{3.13}$$

Then:

$$\underline{\underline{H}}_{n+1} = e^{\Delta t/t}\, \underline{\underline{H}}_n + \frac{1 - e^{-\Delta t/\nu}}{\Delta t/\nu}\, \mathrm{dev}[\underline{\underline{\Pi}}_{n+1} - \underline{\underline{\Pi}}_n]$$

$$\underline{\underline{\sum}}_{n+1} = \underline{\underline{\sum}}_n + (G_0 - G_\infty)[\underline{\underline{H}}_{n+1} - \underline{\underline{H}}_n]$$

$$+ G_\infty\, \mathrm{dev}\, [\underline{\underline{\Pi}}_{n+1} - \underline{\underline{\Pi}}_n] \tag{3.14}$$

In these equations, the effective strain $\underset{\sim}{\Pi}_{n+1}$ and $\underset{\sim}{\Pi}_n$ are computed as

$$\underset{\sim}{\Pi}_{n+1} = \overline{f}_{n+1}\, \underset{\sim}{\overline{C}}_{n+1} \quad ; \quad \underset{\sim}{\Pi}_n = f_n\, \underset{\sim}{\overline{C}}_n \tag{3.15}$$

where

$$\psi_{n+1} = \max \{ \, ||\text{dev}\, \underset{\sim}{\overline{C}}_{n+1}|| \, , \psi_n \} \quad ; \quad \overline{f}_{n+1} \equiv \overline{f}(\psi_{n+1}) \tag{3.16}$$

This completes our discussion of the algorithmic treatment of the nonlinear viscoelastic constitutive models developed in Section 1 and 2.

Remark 3.4. Identical implementations holds for the alternative characterization of damage discussed in Remark 2.1. This is the model which is implemented in the computer program included in the Appendix. □

4. NUMERICAL EXPERIMENTS

Simple numerical tests are presented for the purpose of exhibiting the basic behavior of the viscoelastic model and damage mechanism. In addition, the numerical solution of a boundary value problem that employs the present model is also considered.

It is emphasized that lack of experimental data prevents us from undertaking a comparison between the analytical models developed here and observed material response. It is expected that further experimental research and extensive exercise of the present model will enable such a comparison in the future.

Example 1. In this example, we illustrate the ability of the exponential damage law (2.14-2.15) of modeling the response of highly filled rubbers. For this purpose, we consider first experimental data obtained at LLNL and plotted in Fig. 4.1 (a). The figure clearly exhibits a degradation of the loss and storage moduli with increasing amplitude. Figure 4.1 (b) shows the normalized curves obtained from Fig. 4.1 (a). It is interesting to note that the curves for G' and G'' in Fig. 4.1 (a) lead to the same normalized curve in Fig. 4.1 (b). This figure also shows the excellent fit obtained with all exponential law.

Figure 4.2 shows the experimental data reported in [6], obtained by subjecting the specimen to a pure shear test. Again an excellent fit is obtained by means of the exponential damage law (2.14-2.15). □

Example 2. In this example, we consider the numerical simulation of a rubber block undergoing pure shear deformation. The data for this simulation corresponds to Fig. 4.2.

The specimen is subjected to a cyclic shear strain history, in the form of triangular pulse, and with increasing amplitude A. The progressive damage of the storage and loss moduli with increasing amplitude is clearly exhibited.

These results are in agreement with the experimental test reported in [6]. It should be noted that the discrepancy in the shape of the loop between Fig. 4.3 and the results reported in [6] is due to the different shape of the cyclic pulse employed in [6]. Analogous numerical simulations carried out with sinosoidal pulses revealed that the shape of the hysteretic loop is highly sensitive to the shape of the strain loading pulse. □

Example 3. The purpose of this final example is to demonstrate the effectiveness of the numerical algorithms and computer implementation discussed in Section 3. For this purpose, the extension of a (plane strain) strip clamped on both ends is considered. The data for this simulation is contained in Fig. 4.3 with α now given as $\alpha = .333$. The mesh consists of 50 bilinear 4-noded isoparametric elements with constant Jacobian and constant pressure on the elements in its current configuration. We refer to [7] for the details of the finite element formulations. The deformed mesh and the stress contours of the principal stresses are shown in Fig. 4.4 a,b and c for extension ratios of 50% and 100%.

Here we give special attention to the convergence behavior observed during the numerical simulation. The solution strategy employed is a standard Newton-Raphson incremental iterative procedure. The values of the maximum euclidean norm of the out-of-balance force during the iteration process corresponding to steps 2, 3 and 4 are displayed in Table 2. The final extension of 100% was achieved in 6 loading steps.

It should be noted that this norm is far more demanding than the usual energy norm used in the NIKE programs. The results in Table 2 clearly

demonstrate that the asymptotic rate of convergence is quadratic. The reason for this is that the consistently linearized tangent stiffness was employed. The details pertaining to this linearization process are nontrivial and will be considered in detail in a forthcoming report.

ACKNOWLEDGEMENTS

This work was performed for the Lawrence Livermore National Laboratory under Purchase Order Number 4627805 as part of a program supported by the U.S. Army Tank-Automotive Command, Warren, Michigan. The continued interest and support of Drs. Roger Werne, Don Lesuer, Alfred Goldberg, and Gerald Goudreau are gratifully acknowledged. Jacob Patt was the engineer for this study for TACOM. We also are indebted to Prof. J. Lubliner for helpful discussions.

REFERENCES

1. Simo, J. C. and R. L. Taylor, "A simple three dimensional viscoelastic model accounting for damage effects," Report No. UCB/SESM/83-10, SESM, U.C. Berkeley (1983).

2. Key, S. W. and R. D. Krieg "On the numerical implementations of inelastic time dependent and time independent, finite strains constitutive equations in structural mechanics," Comp. Meth. Appl. Mech. Eng., Vol. 33, 439-452 (1982).

3. Simo, J. C. and J. E. Marsden, "On the rotated stress tensor and the material version of the Doyle-Ericksen formula," Arch. Rat. Mech. Anal., Vol. 80, 3, 213-231 (1984).

4. Christensen, R. M., A nonlinear theory of viscoelasticity for application to elastomers, J. Appl. Mech., 47, 762-768 (1980).

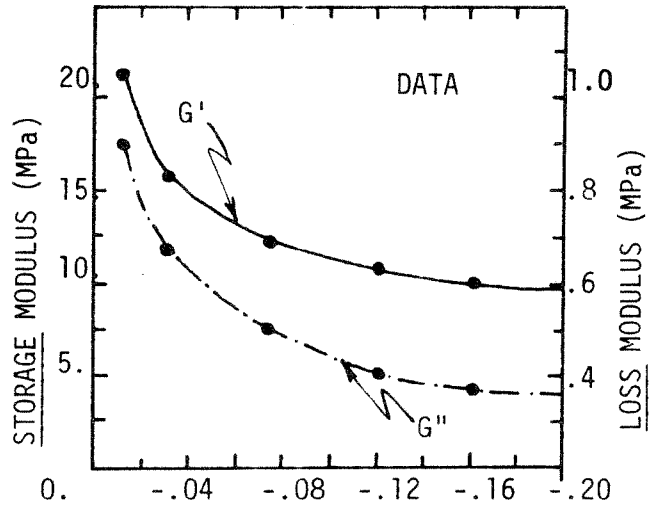
5. Hughes, T.J.R. and J. Winget, "Finite rotation effects in numerical integration of rate constitutive equations arising in large deformation analysis," Int. J. Num. Meth. Engng., Vol. 15, No. 12 (1980).

6. Kelly, J.M. and M. Celebi, "Verification testing of prototype bearings for a based isolated building," Report No. UCB/SESM-84101, Department of Civil Engineering, U.C. Berkeley (1984).

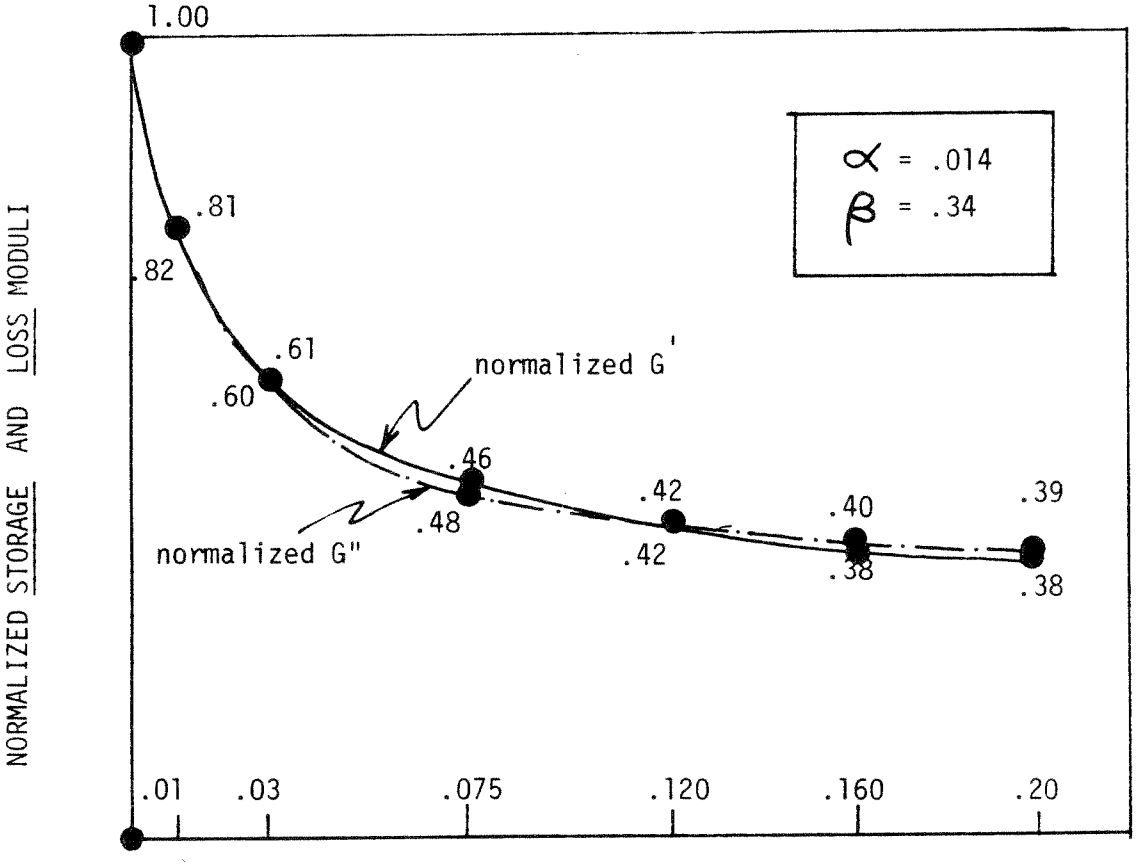
7. Simo, J.C. R.L. Taylor and K.S. Pister, "Variational and projection methods for the volume constraint in finite deformations elastoplasticity," Comp. Meth. Appl. Mech. Engng.., (to appear).

Fig.4.1. EXPONENTIAL FIT to L.L.N,L EXPERIMENTAL DATA.

(a)



(b)



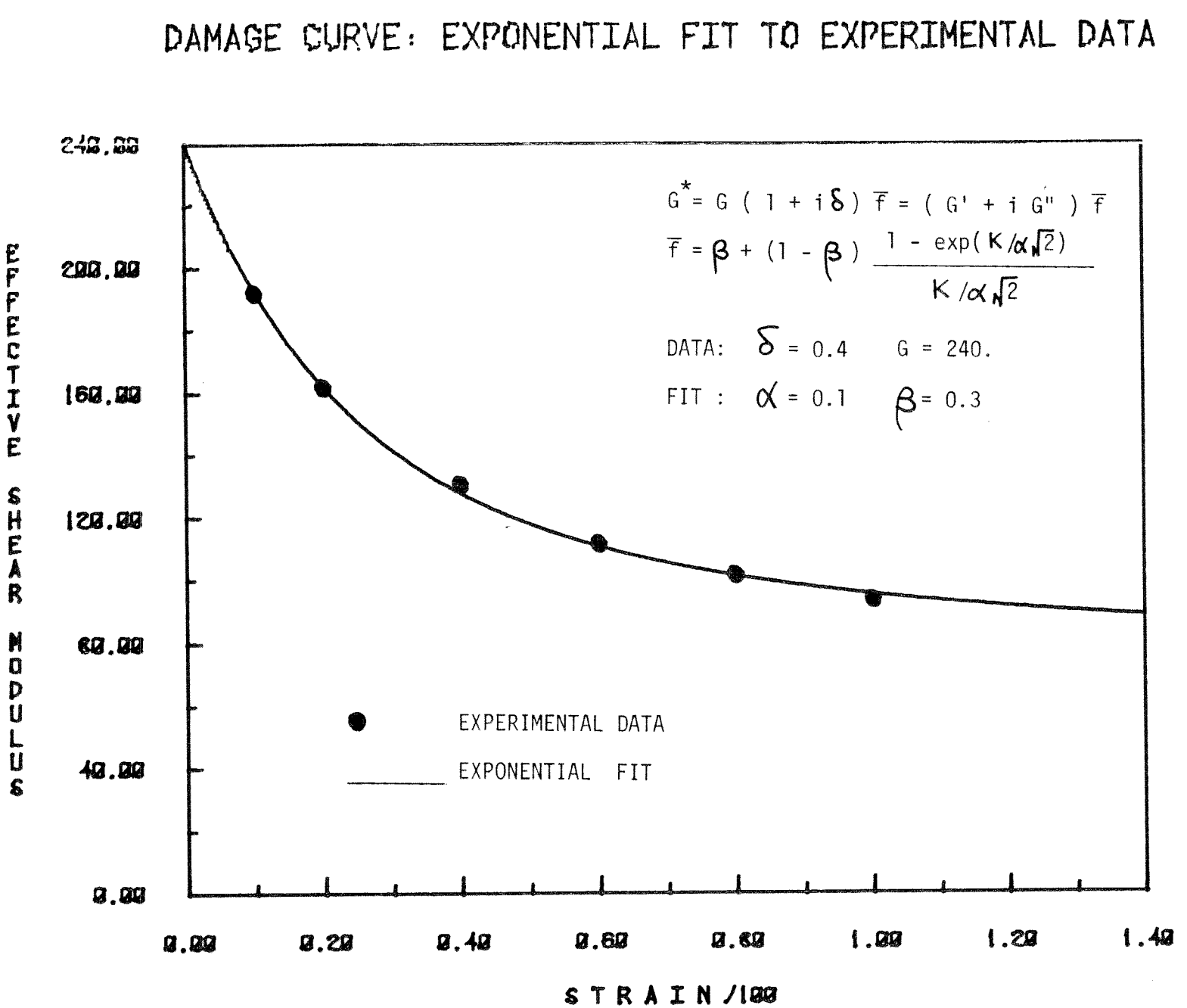


Fig.4.2. Exponential fit to experimental data reported in [6]. The elastomer is a highly fiiled rubber.

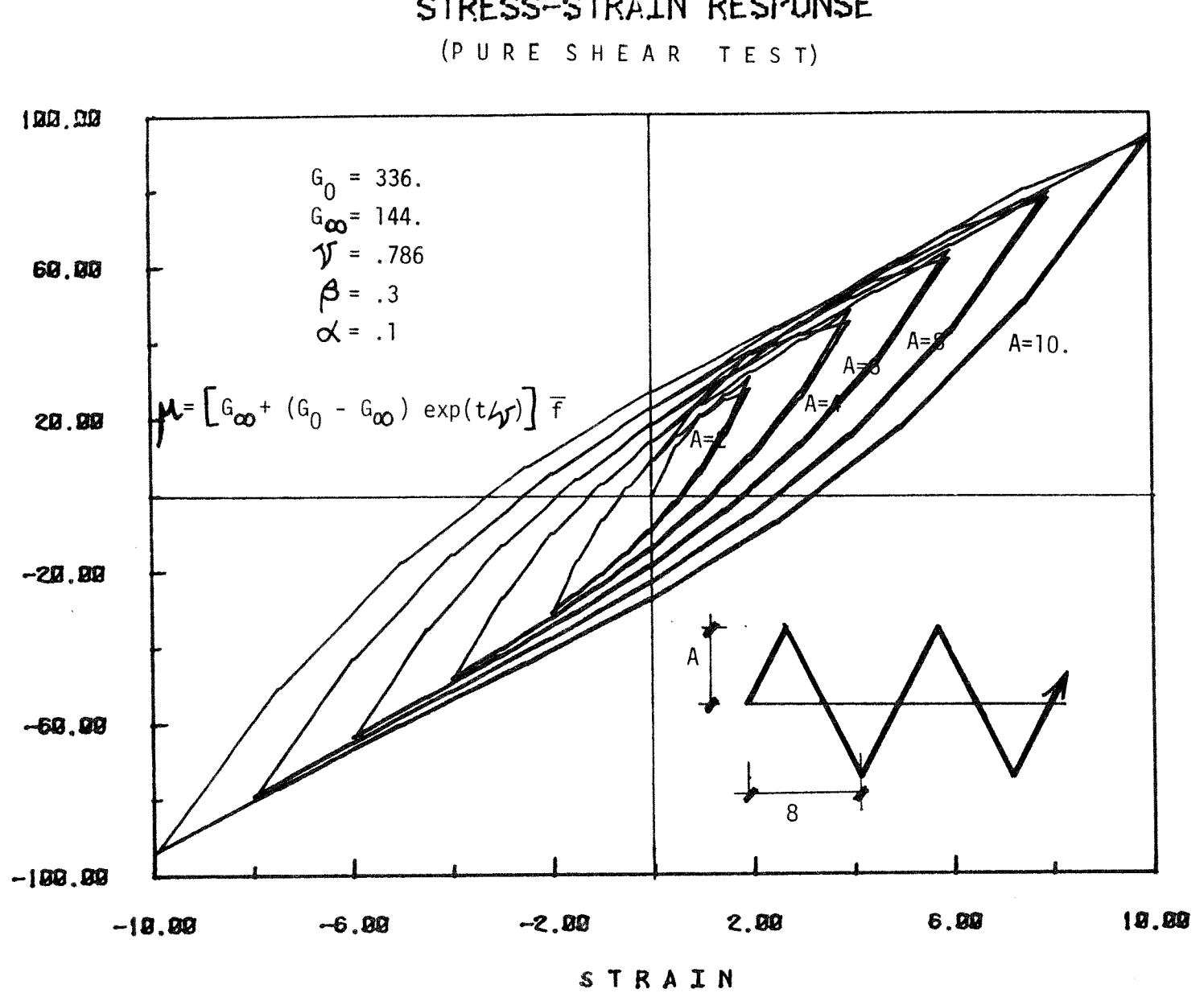
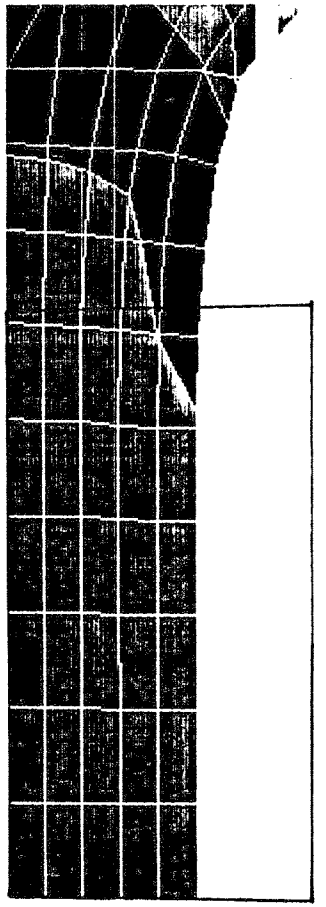


Fig.4.3. Numerical simulation in a pure shear test with data obtained from Fig.4.2. The strain history is a triangular pulse.

TABLE 2.

Values of the Euclidean Norm of the Residual

STEP #	MAXIMUM NORM
	$.3941.10^{5}$
2	$.6506.10^{3}$
	$.8008.10^{2}$
	$.2486.10^{1}$
	$.4285.10^{-2}$
	$.3705.10^{-8}$
	$.3746.10^{5}$
3	$.6159.10^{3}$
	$.2552.10^{2}$
	$.2526.10^{1}$
	$.4138.10^{-3}$
	$.2794.10^{-9}$
	$.3591.10^{5}$
4	$.5584.10^{3}$
	$.1005.10^{2}$
	$.4546.10^{0}$
	$.7936.10^{-5}$



CONTOURS OF PRINCIPAL (TRACTION) STRESS

< 1.300e+02

to 1.475e+02

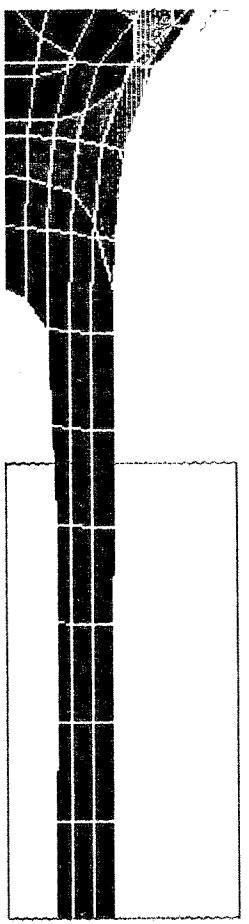
to 1.650e+02

to 1.825e+02

> 2.000e+02

VISCOELASTIC MATERIAL WITH DAMAGE

50% EXTENSION



CONTOURS OF PRINCIPAL (TRACTION) STRESS

< 2.400e+02

to 2.650e+02

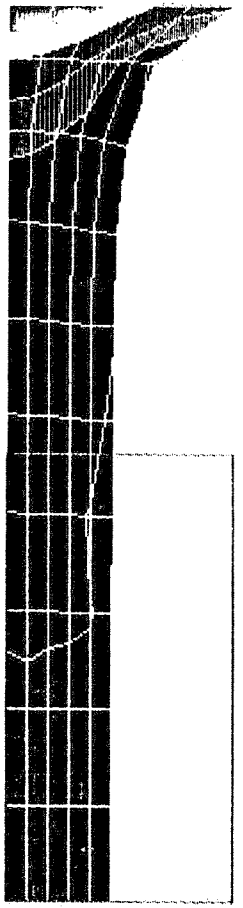
to 2.900e+02

to 3.150e+02

> 3.400e+02

VISCO-ELASTIC MATERIAL WITH DAMAGE

100 % EXTENSION.



CONTOURS OF PRINCIPAL
(COMPRESSION) STRESS

< -1.000e-02

to 3.749e+01

to 7.500e+01

to 1.125e+02

> 1.500e+02

VISCOELASTIC MATERIAL
WITH DAMAGE

100 % EXTENSION

Mar 9 15:29 1985 elmt10.f Page 1

```
      subroutine elmt10(d,ul,xl,ix,tl,s,p,ndf,ndm,nst,isw)
      implicit double precision (a-h,o-z)
c
c.... plane FINITE DEFORMATION VISCOELASTIC element routine
c          ** Mixed Model    ** Version 2/13/85
c
      character*4 o,head
      common /bdata/ o,head(20)
      common /cdata/ numnp,numel,nummat,nen,neq,ipr
      common /eldata/ dm,n,ma,mct,ie,nel
      common /hdata/ nh1,nh2
      common /iofile/ ifile,jfile
      common /tdata/ time,dt,c1,c2,c3,c4,c5
      common /yield/ yldfn,phi1,gg1,phi2,gg2,phi3,gg3
      common /plstrs/ np
      common h(1),mh(1)
      dimension d(1),ul(ndf,1),xl(ndm,1),ix(1),tl(1),s(nst,1),
     1          p(1),sg(4),tg(4),wg(4),yld(4)
      dimension bbar(4,2,4),bbd(2,4),bbd2(2),xu(2,4),hl(65),
     1          aa(4,4),sigv(4),shpa(2,4),shps(3,4,4),dvol(4),
     2          ci(4,4),bi(4,4),fi(4,4),tri(4),xjac(4),sigv1(7)
      save /bdata/,/cdata/,/eldata/,/hdata/,/tdata/,/yield/,
     1     /plstrs/,sg,tg,wg,lint
c
c.... go to correct array processor
c
      go to(1,2,3,4,5,3,2,4), isw
c
c.... input material properties
c
    1 read(ifile,995)  imat,itype,isymm
      read(ifile,1000) (d(i),i=1,7)
      mct = 65
      d(13) = 0.0
      if(itype .ne. 0)  d(13) = 1.d0
      d(14) = 0.5d0
      if (isymm .ne. 0) d(14) = 1.d0
      lint = 0
      write(jfile,2000) imat,(d(i),i=1,7),d(13),isymm
c
    2 return
c
c.... compute tangent stiffness and residual force vector
c
    3 l     = 2
      type  = d(13)
      if(l*l.ne.lint) call pgauss(l,lint,sg,tg,wg)
      nlast = lint*16
      isym  = 0
      if (d(14) .ne. 0.5d0) isym = 1
c
c.... compute current geometry
      do 310 i = 1,2
      do 310 j = 1,nel
  310 xu(i,j) = xl(i,j) + ul(i,j)
c
```

```
c.... First loop over Gauss points to compute b-bar averages.
c
      vol0 = 0.d0
      voln = 0.d0
      call pzero(shpa,8)
      nn   = 0
      do 330 l = 1,lint
c
c.... get shape functions and derivatives in current geometry
      call shape(sg(l),tg(l),xu,shps(1,1,l),xsj,2,4,ix,.false.)
c
c.... Compute bi and det(fei) at conf t-n+1
      nnh  = nh1 + nn
      call kine10(d,type,xr1,xr,xz,xr0,xz0,shps(1,1,l),xl,ul(1,1),
     1            ndm,ndf,nel,nen,ci(1,l),bi(1,l),
     2            fi(1,l),tri(l),xjac(l))
      dvol(l) = xsj*wg(l)*xr1
c
c.... Compute averages of shape functions and volume element
      typer       = type/xr
      do 320 k = 1,nel
      shps(3,k,l) = shps(3,k,l)*typer
      shpa(1,k)   = shpa(1,k) + (shps(3,k,l) + shps(1,k,l))*dvol(l)
 320  shpa(2,k)   = shpa(2,k) + shps(2,k,l)*dvol(l)
      vol0        = vol0 + dvol(l)/xjac(l)
      voln        = voln + dvol(l)
 330  nn = nn + 16
c
c.... get volumetric terms and average jacobian
      do 340 i = 1,2
      do 340 k = 1,nel
 340  shpa(i,k) = shpa(i,k)/voln
      nfin      = nh2+nlast
      h(nfin)   = voln/vol0
c
c.... Loop over Gauss points to compute stiffness and R.H.S
c
      nn = 0
      do 400 l = 1,lint
c
c.... Compute Kirchhoff stresses and spatial tangent tensor at t-n+1
      call modl10(d,ci(1,l),bi(1,l),fi(1,l),tri(l),xjac(l),
     2            h(nn+nh1),h(nn+nh2),h(nfin),sigv,aa,imat)
c
c.... multiply tangent moduli and stress by volume element
      nnh = nh2 + nn
      do 350 i = 1,4
      sigv(i) = sigv(i)*dvol(l)
      do 350 j = 1,4
350   aa(i,j) = aa(i,j)*dvol(l)
c
c.... compute B-bar matrix
      do 355 k = 1,nel
      call bmat10(shps(1,k,l),bbar(1,1,k))
 355  continue
c
```

```
c.... compute the internal stress divergence term
      k1 = 1
      do 365 k = 1,nel
      k2 = k1 + 1
      do 360 i = 1,4
      p(k1) = p(k1) - bbar(i,1,k)*sigv(i)
360   p(k2) = p(k2) - bbar(i,2,k)*sigv(i)
365   k1 = k1 + ndf
c
c.... compute the tangent stiffness matrix
c     part 1. -- geometric part (based upon sigma-n+1)
      i1 = 1
      do 375 i = 1,nel
      j1 = i1 + isym*(1 - i1)
      ib = i  + isym*(1 -i)
      tb1   = shps(1,i,1)*sigv(1) + shps(2,i,1)*sigv(4)
      tb2   = shps(1,i,1)*sigv(4) + shps(2,i,1)*sigv(2)
      tb3   = shps(3,i,1)*sigv(3)
      do 370 j = ib,nel
      bdb   = tb1*shps(1,j,1)   + tb2*shps(2,j,1)
      bdb33 = shps(3,j,1)*tb3
      s(i1  ,j1  ) = s(i1  ,j1  ) + bdb + bdb33
      s(i1+1,j1+1) = s(i1+1,j1+1) + bdb
370   j1 = j1 + ndf
375   i1 = i1 + ndf
c.... part 2. -- tangent modulus part (based upon aa-array)
      i1 = 0
      do 395 i  = 1,nel
c.... compute bbar-t * aa * dvol
      do 380 ii = 1,2
      do 380 jj = 1,4
      bbd(ii,jj) = 0.0
      do 380 kk = 1,4
380   bbd(ii,jj) = bbd(ii,jj) + bbar(kk,ii,i)*aa(kk,jj)
c.... compute tangent stiffness
      j1 = i1 + isym*(0 - i1)
      ib = i  + isym*(1 - i)
      do 390 j  = ib,nel
      do 385 ii = 1,2
      do 385 jj = 1,2
      do 385 kk = 1,4
385   s(i1+ii,j1+jj) = s(i1+ii,j1+jj) + bbd(ii,kk)*bbar(kk,jj,j)
390   j1 = j1 + ndf
395   i1 = i1 + ndf
400   nn = nn + 16
c
c.... add constant pressure effect (Mixed model)
      aa2 = d(1)*voln
      i1 = 0
      do 410 i = 1,nel
      do 402 ii = 1,2
 402  bbd2(ii) = aa2*shpa(ii,i)
      j1 = i1 + isym*(0 - i1)
      ib = i  + isym*(1 - i)
      do 408 j = ib,nel
      do 406 ii = 1,2
```

```
      do 406 jj = 1,2
 406  s(i1+ii,j1+jj) = s(i1+ii,j1+jj) + bbd2(ii)*shpa(jj,j)
 408  j1 = j1 + ndf
 410  i1 = i1 + ndf
c
      if (isym .eq. 0) then
c.....   compute lower part by symmetry
         do 412 i = 1,nst
         do 412 j = i,nst
 412     s(j,i) = s(i,j)
      endif
      return
c
c.... Output stresses.
c
    4 l    = 2
      lint = l*l
      type = d(13)
      if(l*l.ne.lint) call pgauss(l,lint,sg,tg,wg)
      nlast = lint*16
c
c.... compute current geometry
      do 415 i = 1,2
      do 415 j = 1,nel
 415  xu(i,j) = xl(i,j) + ul(i,j)
c
c.... First loop over Gauss points
      rr   = 0.d0
      zz   = 0.d0
      vol0 = 0.d0
      voln = 0.d0
      nn   = 0
      do 420 l = 1,lint
      call shape(sg(l),tg(l),xu,shps(1,1,l),xsj,2,4,ix,.false.)
c
c.... Compute bi, and det(fei) at conf t-n+1
      nnh = nh1 + nn
      call kine10(d,type,xr1,xr,xz,xr0,xz0,shps(1,1,l),xl,ul(1,1),
     1            ndm,ndf,nel,nen,ci(1,l),bi(1,l),
     2            fi(1,l),tri(l),xjac(l))
      rr    = rr + 0.25*xr
      zz    = zz + 0.25*xz
      dvoln = xsj*wg(l)*xr1
      voln  = voln + dvoln
      vol0  = vol0 + dvoln/xjac(l)
 420  nn    = nn + 16
      xjca  = voln/vol0
      do 425 i = 1,4
 425  sigv1(i) = 0.d0
      ddm  = 0.d0
c
c.... Second loop over Gauss points
      nn = 0
      do 430 l = 1,lint
c
c.... Compute Cauchy stresses and spatial tangent tensor at t-n+1
```

```
      call modl10(d,ci(1,1),bi(1,1),fi(1,1),tri(1),xjac(1),
     2            h(nn+nh1),hl(nn+1),xjca,sigv,aa,imat)
c
c.... move stresses and jacobian for printing
      ddm     = ddm + 0.25*yldfn
      yld(1)  = yldfn
      sigv1(1) = sigv1(1) + 0.25*sigv(1)
      sigv1(2) = sigv1(2) + 0.25*sigv(4)
      sigv1(3) = sigv1(3) + 0.25*sigv(2)
      sigv1(4) = sigv1(4) + 0.25*sigv(3)
430   nn = nn + 16
      if (isw .eq. 8) go to 8
      call pstres(sigv1,sigv1(5),sigv1(6),sigv1(7))
c
c.... output stresses
      mct = mct - 2
      if(mct.gt.0) go to 440
      write(jfile,2004) phi1,phi2,phi3,gg1,gg2,gg3
      write(jfile,2001) o,head
      mct = 50
 440  write(jfile,2002) n,ma,(sigv1(ii),ii=1,6),rr,zz,xjca,ddm,sigv1(7)
      return
c
c.... compute consistent mass matrix
c
   5  l   = 2
      rho = d(7)
      if(l*l.ne.lint) call pgauss(l,lint,sg,tg,wg)
      do 500 l = 1,lint
c.... compute shape functions
      call shape(sg(l),tg(l),xl,shps,xsj,ndm,nel,ix,.false.)
      dm = wg(l)*xsj*rho
c.... for each node j compute db = rho*shape*dm
      j1 = 1
      do 500 j = 1,nel
      w11 = shps(3,j,1)*dm
c.... compute a lumped mass
      p(j1) = p(j1) + w11
c.... for each node k compute mass matrix (upper triangular part)
      k1 = j1
      do 510 k = j,nel
      s(j1,k1) = s(j1,k1) + shps(3,k,1)*w11
510   k1 = k1 + ndf
500   j1 = j1 + ndf
c.... compute missing parts and lower part by symmetries
      nsl = nel*ndf
      do 520 j = 1,nsl,ndf
      p(j+1) = p(j)
      do 520 k = j,nsl,ndf
      s(j+1,k+1) = s(j,k)
      s(k,j) = s(j,k)
520   s(k+1,j+1) = s(j,k)
      return
c
c.... project stresses onto the nodes
   8  npl = np + (numnp-1)*ipr
```

```
      call stcn10(ix,xl,yld,hl(2),mh(np-ipr),mh(np1),
     1            ndm,nel,numnp)
      return
c
c.... formats for input-output
 995  format(3i10)
1000  format(7f10.0)
2000  format(/5x,'plane strain-axisymmetric FINITE DEFORMATION'/
     * 5x,'ELASTIC-VISCOELASTIC Element  ** MIXED model'//
     1 10x,'type(1=ELAS. 2=VISC. 3,4,5=DAMAGE)',i3/
     2 10x,'bulk modulus',e25.5/
     3 10x,'short term shear modulus',e13.5/
     4 10x,'long term shear modulus',e14.5/
     5 10x,'time constant, beta',e18.5/
     6 10x,'damage limit',e25.5/10x,'damage exponent',e22.5/
     7 10x,'density',e30.5/
     8 10x,'type (0=plane strain, 1=axisym.)',f5.0/
     9 10x,'Symmetrized tangent  (0=Symmetry)',i3/)
2001  format(a1,20a4//5x,16helement stresses//'  elmt  matl'
     1    ,2x,9h11-stress,2x,9h12-stress,2x,9h22-stress,2x,
     2     9h33-stress,3x,8h1-stress,3x,8h2-stress/
     3     2x,7h1-coord,2x,7h2-coord,2x,8hjacobian,2x,8hyield fn,
     4     24x,5hangle)
2002  format(2i6,6e11.3/2f9.3,2f10.4,21x,f8.2/1x)
2004  format(/12x,'d a m a g e   v a r i a b l e s'/
     1 '||dev (C)|| =',f10.5,2x,'1/2 tr (C)  =',f10.5,2x,
     2 'sqrt[tr(C)] =',f10.5/'damage gg1  =',f10.5,2x,
     3 'damage gg2  =',f10.5,2x,'damage gg3  =',f10.5//)
      end
c
      subroutine stcn10(ix,xl,yld,sigh,dt,st,ndm,nel,nnp)
      implicit double precision (a-h,o-z)
      dimension dt(nnp),st(nnp,1),xl(ndm,1),sig(5),ss(4),tt(4),
     1          ix(1),sigh(16,1),yld(4)
      data ss/-1.0d0,1.0d0,1.0d0,-1.0d0/,tt/-1.0d0,-1.0d0,1.0d0,1.0d0/
c.... compute jacobian parts
      sqr3 = dsqrt(3.0d0)
      call qjac10 (xl,ndm, xsj0,xsj1,xsj2)
      do 200 ii = 1,4
c.... compute stresses at nodes from history terms
      call pzero (sig,5)
      do 100 jj = 1,4
      xsj = xsj0 + sqr3*ss(ii)*xsj1 + sqr3*tt(ii)*xsj2
      shpj = 0.25*(1.+sqr3*ss(ii)*ss(jj))*(1.+sqr3*tt(ii)*tt(jj))
      sig(1) = sig(1) + shpj*sigh(1,jj)*xsj
      sig(2) = sig(2) + shpj*sigh(4,jj)*xsj
      sig(3) = sig(3) + shpj*sigh(2,jj)*xsj
      sig(4) = sig(4) + shpj*sigh(3,jj)*xsj
100   sig(5) = sig(5) + shpj*yld(jj)*xsj
c.... compute stresses
      ll = ix(ii)
      if(ll.le.0) go to 200
      dt(ll) = dt(ll) + xsj0 + (ss(ii)*xsj1 + tt(ii)*xsj2)/3.
      st(ll,1) = st(ll,1) + sig(1)
      st(ll,2) = st(ll,2) + sig(2)
      st(ll,3) = st(ll,3) + sig(3)
```

```
      st(ll,4) = st(ll,4) + sig(4)
200   st(ll,5) = st(ll,5) + sig(5)
      return
      end
      subroutine qjac10(xl,ndm,xsj0,xsj1,xsj2)
      implicit double precision (a-h,o-z)
c.... jacobian routine for 4-node quadrilaterals
      dimension xl(ndm,4)
c.... set up natural coordinate functions (times 4)
      xo =  xl(1,1)-xl(1,2)+xl(1,3)-xl(1,4)
      xs = -xl(1,1)+xl(1,2)+xl(1,3)-xl(1,4)
      xt = -xl(1,1)-xl(1,2)+xl(1,3)+xl(1,4)
      yo =  xl(2,1)-xl(2,2)+xl(2,3)-xl(2,4)
      ys = -xl(2,1)+xl(2,2)+xl(2,3)-xl(2,4)
      yt = -xl(2,1)-xl(2,2)+xl(2,3)+xl(2,4)
c.... compute jacobian parts
      xsj0 = 0.0625*(xs*yt - xt*ys)
      xsj1 = 0.0625*(xs*yo - xo*ys)
      xsj2 = 0.0625*(xo*yt - xt*yo)
      return
      end
c
      subroutine bmat10(shp,bbar)
      implicit double precision (a-h,o-z)
      dimension shp(3),bbar(4,2)
c
c.... b matrix for plane and axisymmetric problems
c
      bbar(1,1) = shp(1)
      bbar(2,1) = 0.0d0
      bbar(3,1) = shp(3)
      bbar(4,1) = shp(2)
      bbar(1,2) = 0.0d0
      bbar(2,2) = shp(2)
      bbar(3,2) = 0.0d0
      bbar(4,2) = shp(1)
c
      return
      end
```

```
      subroutine modl10(d,ci,bi,fi,trbi,xjci,hn,hn1,xjca,sig,aa,imat)
      implicit double precision (a-h,o-z)
c
c.... finite deformation ELASTIC and VISCOELASTIC models for FEAP
c     Isotropic DAMAGE mechanism inluded.
c
      common /tdata/ time,dt,c1,c2,c3,c4,c5
      common /yield/ yldfn,phi1,gg1,phi2,gg2,phi3,gg3
      common /iofile/ ifile,jfile
      dimension d(1),ci(4),bi(4),fi(4),hn(16),
     1          hn1(16),h(16),sig(4),aa(4,4)
      save /yield/
c
c.... move history variables for start
c
      do 100 i = 1,16
100   h(i) = hn(i)
c
c.... compute KIRCHHOFF pressure
c
      pressj = xjci*d(1)*dlog(xjca)
c
      go to(1,2,2,3), imat
  1   call elcn10(d,ci,bi,trbi,xjci,pressj,
     *            h(1),h(11),h(2),sig,aa)
      go to 10
c
  2   call vsdd10(d,ci,bi,fi,trbi,xjci,pressj,
     *            h(1),h(11),h(2),h(6),h(10),sig,aa,imat)
      go to 10
c
  3   call vscd10(d,bi,fi,trbi,xjci,pressj,
     *            h(1),h(11),h(2),h(6),h(10),sig,aa,imat)
c
c.... get CAUCHY stresses and tangent moduli
c
  10  continue
      do 140 i = 1,4
      do 140 j = 1,4
 140  aa(i,j) = aa(i,j)/xjci
      do 150 i = 2,4
 150  h(i)    =  h(i)   + pressj
      do 200 i = 1,16
 200  hn1(i) = h(i)
      do 210 i = 1,3
 210  sig(i)  = (sig(i) + pressj)/xjci
      sig(4)  =  sig(4)/xjci
      return
      end
c
      subroutine elcn10 (d,ci,bi,trbi,xjci,pressj,
     1                   xjcn,cn,sigtn,sigi,aa)
      implicit double precision (a-h,o-z)
c
c.... FINITE DEFORMATION ELASTIC model for FEAP
c
```

```
      common /tdata/ time,dt,c1,c2,c3,c4,c5
      save /tdata/
      dimension d(1),cn(4),bi(4),ci(4),sigtn(4),
     1          sigi(4),aa(4,4)
c
c.... compute Kirchhoff stress deviator
c
      do 80 i = 1,4
 80   sigi(i) = d(2)*bi(i)
c
      do 90 i = 1,4
      sigtn(i) = sigi(i)
 90   cn(i)    = ci(i)
      xjcn     = xjci
c
c.... compute tangent moduli
c
      xmu    =  d(2)
      xmu2   = (xmu + xmu)*trbi
      xlambr =  - xmu2/3.0d0 + pressj
      xmubr2 =    xmu2       - pressj - pressj
c
      do 120 i = 1,3
      aa(i,4) = 0.0d0
      aa(4,i) = 0.0d0
      do 110 j = 1,3
110   aa(i,j) = xlambr
120   aa(i,i) = xlambr + xmubr2
      aa(4,4) = 0.5d0*xmubr2
c
      do 130 i = 1,4
      fact    = bi(i)*(xmu + xmu)/3.d0
      do 130 j = 1,3
      aa(j,i) = aa(j,i) - fact
 130  aa(i,j) = aa(i,j) - fact
c
      return
      end
c
      subroutine vedd10 (d,ci,bi,fi,trbi,xjci,pressj,
     1                   xjcn,cn,sigtn,hsn,phmax,sigi,aa,imat)
      implicit double precision (a-h,o-z)
c
c.... FINITE DEFORMATION VISCOELASTIC DAMAGE model for FEAP
c
c     Damage rule:    phi = 1/2 tr[J**(-2/3) C - I]      (imat=2)
c                     phi = sqrt(tr[J**(-2/3) C - I])    (imat=3)
c
      common /tdata/ time,dt,c1,c2,c3,c4,c5
      common /yield/ yldfn,phi1,gg1,phi2,gg2,phi3,gg3
      common /iofile/ ifile,jfile
      dimension d(1),ci(4),bi(4),fi(4),vi(4),ri(4),cn(4),sigtn(4),
     1          hsn(4),sigi(4),sigdn(4),aa(4,4),delta(4),signn(4)
      save /tdata/,/yield/
c
c.... set viscoelastic parameters
```

```
c
      qs = d(2)
      q1 = d(3)
      qb = d(4)*dt
      qd = qs - q1
      q1 = dexp(-qb)
      q2 = 1.0d0 - 0.5*qb + qb*qb/6.0d0
      if(q1.lt.0.99999d0) q2 = (1.0d0 - q1)/qb
c
c.... compute damage variables
c
      trei = trbi - 1.d0
      psi2 = 1.5*trei/d(6)
      psi3 = dsqrt(3.d0*trei)/d(6)
c
      gact = 3.d0
      if (imat.eq.2) gact = 2.d0
      phi = (3.d0 - gact)*psi2 + (gact - 2.d0)*psi3
      gg0  = damf10(phmax,d(5))
c
c.... check for loading
      if (phi.gt.phmax) then
         phi1   = dsqrt(dot(ci,ci,4) + ci(4)*ci(4))/d(6)
         phi2   = psi2
         phi3   = psi3
         gg1    = damf10(phi1,d(5))
         gg2    = damf10(phi2,d(5))
         gg3    = damf10(phi3,d(5))
         gg     =  (3.d0 - gact)*gg2 + (gact - 2.d0)*gg3
         iyld   = 1
         phmax  = phi
         f0     = phi*(-.5d0 + phi/3.0d0 - phi*phi*.125d0)
         xnorm  = 1.0d0
         if(phi.gt.0.00001d0) then
            f0    = dexp(-phi)
            f0    = f0 - (1.d0 - f0)/phi
            xnorm = phi*d(6)
         endif
         f0    = (1.0d0 - d(5))*f0
      else
         iyld   = 0
         gg     = gg0
      endif
c
c.... update pseudo-stresses (in ROTATED configuration)
c
      trstn    = (sigtn(1) + sigtn(2) + sigtn(3))/3.d0
      do 95 i = 1,3
 95   sigdn(i) = sigtn(i) - trstn
      sigdn(4) = sigtn(4)
      do 100 j = 1,4
 100  delta(j) = gg*ci(j) - gg0*cn(j)
c
c.... superpose viscous effects in ROTATED configuration
c
      xmu     = (q1 + qd*q2)*gg
```

```
      do 102 j = 1,4
      t        = hsn(j)
      hsn(j)   = q1*t      + q2*delta(j)
      sigi(j) = sigdn(j) + q1*delta(j) + qd*(hsn(j) - t)
102   signn(j)= sigi(j)  - xmu*ci(j)
c
c.... update ROTATED variables
c
      do 105 i = 1,4
      cn(i)    = ci(i)
105   sigtn(i) = sigi(i)
      xjcn     = xjcni
c
c.... Push-forward with ROTATION to obtain Kirchhoff stresses
c
      call pola10(fi,ri,vi)
      call pusr10(sigi,ri)
      call pusr10(signn,ri)
c
c.... compute the elastic (dameged) part of tangent moduli
c
      xmu2   =  (xmu + xmu)*trbi
      xlambr =- xmu2/3.0d0 + pressj
      xmubr2 =  xmu2       - pressj - pressj
c
      do 120 i = 1,3
      aa(i,4) = 0.0d0
      aa(4,i) = 0.0d0
      do 110 j = 1,3
110   aa(i,j) = xlambr
120   aa(i,i) = xlambr + xmubr2
      aa(4,4) = 0.5d0*xmubr2
c
      do 130 i = 1,4
      fact    = bi(i)*(xmu + xmu)/3.d0
      do 130 j = 1,3
      aa(j,i) = aa(j,i) - fact
 130  aa(i,j) = aa(i,j) - fact
c
      trvi = 1.d0/(vi(1) + vi(2))
      aa11 = signn(4)*vi(4)*2.d0*trvi
      aa44 =(signn(2)-signn(1))*(vi(2)-vi(1))*0.5d0*trvi
c
c.... symmetrize aa if d(14) = 0.5
      bb14 = signn(4)*(vi(2)-vi(1))*trvi
      bb41 =(signn(2)-signn(1))*vi(4)*trvi
      aa14 = d(14)*bb14 + (1.d0 - d(14))*bb41
      aa41 = d(14)*bb41 + (1.d0 - d(14))*bb14
c
      aa(1,1) = aa(1,1) - signn(1) - signn(1) + aa11
      aa(2,2) = aa(2,2) - signn(2) - signn(2) + aa11
      aa(3,3) = aa(3,3) - signn(3) - signn(3)
      aa(1,2) = aa(1,2) - aa11
      aa(2,1) = aa(2,1) - aa11
      aa(1,4) = aa(1,4) - signn(4) + aa14
      aa(2,4) = aa(2,4) - signn(4) - aa14
```

```
      aa(4,1) = aa(4,1) - signn(4) + aa41
      aa(4,2) = aa(4,2) - signn(4) - aa41
      aa(4,4) = aa(4,4) -(signn(1) + signn(2))*.5d0 + aa44
c
c.... update tangent matrix to account for loading
c
      if (iyld.eq.1) then
         xnorm0= (3.d0 - gact)*xnorm + (gact - 2.d0)
         gam1  = xnorm0*xmu*f0/gg
         do 135 i = 1,4
 135     sigdn(i) = bi(i)/xnorm
         do 140 i = 1,4
         temp     = gam1*sigdn(i)
         do 140 j = 1,4
140      aa(i,j)  = aa(i,j) + temp*sigdn(j)
      endif
c
      return
      end
c
      double precision function damf10(ph,beta)
      implicit double precision (a-h,o-z)
c
      gg       = 1.0d0 - 0.50d0*ph + ph*ph/6.0d0
           if(ph.gt..00001d0) gg = (1.0d0 - dexp(-ph))/ph
      damf10   = beta + (1.0d0 - beta)*gg
c
      return
      end
c
      subroutine pusr10(d,r)
      implicit double precision (a-h,o-z)
      dimension d(4),r(4)
c
c.... compute d = r * d -bar* r-t
c
      a11  = r(1)*d(1) + r(3)*d(4)
      a21  = r(2)*d(1) + r(4)*d(4)
      a12  = r(1)*d(4) + r(3)*d(2)
      a22  = r(2)*d(4) + r(4)*d(2)
      d(1) =  a11*r(1) + a12*r(3)
      d(2) =  a21*r(2) + a22*r(4)
      d(4) =  a11*r(2) + a12*r(4)
c
      return
      end
c
      subroutine vscd10 (d,bi,fi,trbi,xjci,pressj,
     1                   xjcn,cn,sigtn,hsn,phmax,sigi,aa,imat)
      implicit double precision (a-h,o-z)
c
c.... FINITE DEFORMATION VISCOELASTIC DAMAGE model for FEAP
c
c     Viscoelasticity in terms of SECOND PIOLA KIRCHHOFF
c     Damage rule:    phi = sqrt(tr[J**(-2/3) C - I])
c
```

```
      common /tdata/ time,dt,c1,c2,c3,c4,c5
      common /yield/ yldfn,phi1,gg1,phi2,gg2,phi,gg
      common /iofile/ ifile,jfile
      dimension d(1),ci(4),bi(4),fi(4),cn(4),sigtn(4),hsn(4),
     1          sigi(4),sigdn(4),aa(4,4),delta(4)
      save /tdata/,/yield/,/iofile/
c
c.... set viscoelastic parameters
c
      qs = d(2)
      ql = d(3)
      qb = d(4)*dt
      qd = qs - ql
      q1 = dexp(-qb)
      q2 = 1.0d0 - 0.5*qb + qb*qb/6.0d0
      if(q1.lt.0.99999d0) q2 = (1.0d0 - q1)/qb
c
c.... compute kinematic variable dev[G bar].
c
      xjci2 = xjci**(-2.d0/3.d0)
      cin11 = fi(1)*fi(1) + fi(2)*fi(2)
      cin22 = fi(3)*fi(3) + fi(4)*fi(4)
      cin33 = (bi(3) + trbi)/xjci2
      cin12 = fi(1)*fi(3) + fi(2)*fi(4)
      detc  = cin11*cin22 - cin12*cin12
      trci  = (cin11 + cin22 + cin33)/3.d0
      trcid = trci/detc
      ci(1) = (1.d0 - trcid*cin22)*xjci2
      ci(2) = (1.d0 - trcid*cin11)*xjci2
      ci(3) = (1.d0 -  trci/cin33)*xjci2
      ci(4) = (       trcid*cin12)*xjci2
c
c.... compute damage variables
c
      trei = trbi - 1.d0
      phi  = dsqrt(3.d0*trei)/d(6)
      gg0  = damf10(phmax,d(5))
c
c.... check for loading
      if (phi.gt.phmax) then
         phi1   = dsqrt(dot(bi,bi,4) + bi(4)*bi(4))/d(6)
         phi2   = 1.5d0*trei/d(6)
         gg1    = damf10(phi1,d(5))
         gg2    = damf10(phi2,d(5))
         gg     = damf10(phi,d(5))
         iyld   = 1
         phmax  = phi
         f0     = phi*(-.5d0 + phi/3.0d0 - phi*phi*.125d0)
         xnorm  = 1.0d0
         if(phi.gt.0.00001d0) then
            f0    = dexp(-phi)
            f0    = f0 - (1.d0 - f0)/phi
            xnorm = phi*d(6)
         endif
         f0    = (1.0d0 - d(5))*f0
      else
```

```
         iyld   = 0
         gg     = gg0
      endif
c
c.... update LAGRANGEAN pseudo-stresses
c
      do 95 j = 1,4
 95   delta(j) = gg*ci(j) - gg0*cn(j)
c
c.... superpose viscous effects in terms of second PIOLA-KIRCHHOFF
c
      do 100 j = 1,4
      t        = hsn(j)
      hsn(j)   = q1*t      + q2*delta(j)
100   sigi(j) = sigtn(j) + q1*delta(j) + qd*(hsn(j) - t)
c
c.... update LAGRANGIAN history variables
c
      do 105 i = 1,4
      cn(i)    = ci(i)
105   sigtn(i) = sigi(i)
      xjcn     = xjcni
c
c.... push-forward second P-K and get KIRCHHOFF stresses.
      call pusr10(sigi,fi)
c
c.... compute the elastic (damaged) part of tangent moduli
c
      xmu    =  (q1 + qd*q2)*gg
      xmu2   =  (xmu + xmu)*trbi
      xlambr=- xmu2/3.0d0 + pressj
      xmubr2=  xmu2       - pressj - pressj
c
      do 120 i = 1,3
      aa(i,4) = 0.0d0
      aa(4,i) = 0.0d0
      do 110 j = 1,3
110   aa(i,j) = xlambr
120   aa(i,i) = xlambr + xmubr2
      aa(4,4) = 0.5d0*xmubr2
c
      do 130 i = 1,4
      fact    = bi(i)*(xmu + xmu)/3.d0
      do 130 j = 1,3
      aa(j,i) = aa(j,i) - fact
 130  aa(i,j) = aa(i,j) - fact
c
c.... update tangent matrix to account for loading
c
      if (iyld.eq.1) then
         gam1     = xmu*f0/gg
         do 135 i = 1,4
 135     sigdn(i) = bi(i)/xnorm
         do 140 i = 1,4
         temp     = gam1*sigdn(i)
         do 140 j = 1,4
```

```
140         aa(i,j)  = aa(i,j) + temp*sigdn(j)
          endif
c
          return
          end
```

```
      subroutine kine10(d,type,xr1,xr,xz,xr0,xz0,shp,xl,ul,ndm,
     1                  ndf,nel,nen,ci,bi,fi,tri,xjci)
      implicit double precision (a-h,o-z)
c
c.... deformation gradient and volume preserving Cauchy-Green tensors
c
      dimension shp(3,1),xl(ndm,1),ul(ndf,1),fi(4),
     1          ci(4),bi(4),finv(2,2),d(1),fibar(4)
c
      ur  = 0.d0
      uz  = 0.d0
      xr0 = 0.d0
      xz0 = 0.d0
      do 305 k = 1,nel
      ur  = ur   + shp(3,k)*ul(1,k)
      uz  = uz   + shp(3,k)*ul(2,k)
      xz0 = xz0  + shp(3,k)*xl(2,k)
305   xr0 = xr0  + shp(3,k)*xl(1,k)
      xr  = xr0  + ur
      xz  = xz0 +  uz
      xr1 = 1.d0 + type*(xr - 1.d0)
c
c.... compute deformation gradient at t-n+1
c
      do 320 i = 1,2
      do 315 j = 1,2
      finv(i,j) = 0.0d0
      do 310 k = 1,nel
 310  finv(i,j) = finv(i,j) - ul(i,k)*shp(j,k)
 315  continue
 320  finv(i,i) = finv(i,i) + 1.0d0
c
      detinv= finv(1,1)*finv(2,2) - finv(1,2)*finv(2,1)
      deti  = 1.d0/detinv
      fi(1) = finv(2,2)*deti
      fi(3) =-finv(1,2)*deti
      fi(2) =-finv(2,1)*deti
      fi(4) = finv(1,1)*deti
c
c.... compute trace and deviator of VOLUME PRESERVING
c     Right and left Cauchy-Green tensors
c
      fact  = 1.d0 + type*(xr/xr0 - 1.d0)
      xjci  = deti*fact
      one3  = -1.d0/3.d0
      xjci3 = xjci**one3
c
      do 340 i = 1,4
 340  fibar(i) = xjci3*fi(i)
c
      bi(1) = fibar(1)*fibar(1) + fibar(3)*fibar(3)
      bi(2) = fibar(2)*fibar(2) + fibar(4)*fibar(4)
      bi(4) = fibar(1)*fibar(2) + fibar(3)*fibar(4)
      ci(1) = fibar(1)*fibar(1) + fibar(2)*fibar(2)
      ci(2) = fibar(3)*fibar(3) + fibar(4)*fibar(4)
      ci(4) = fibar(1)*fibar(3) + fibar(2)*fibar(4)
```

```
      bi(3) = xjci3*xjci3*fact*fact
      ci(3) = bi(3)
c
      tri  = (bi(1) + bi(2) + bi(3))/3.d0
      do 350 i = 1,3
      ci(i) = ci(i) - tri
 350  bi(i) = bi(i) - tri
c
      return
      end
c
      subroutine pola10(f,r,v)
      implicit double precision (a-h,o-z)
      dimension f(4),r(4),v(4)
c
c.... polar decomposition of a rank-2 tensor f = v*r
c
      trf  = f(1) + f(4)
      xnum = f(3) - f(2)
      if (dabs(trf) .lt. 1.0d-5) then
          s = 1.0d0
          c = 0.0d0
      else
          fact = dsqrt(trf*trf + xnum*xnum)
          fact = 1.d0/fact
          s = fact*xnum
          c = fact*trf
      endif
c
      r(1) = c
      r(2) =-s
      r(3) = s
      r(4) = c
c
      v(1) = c*f(1) + s*f(3)
      v(2) =-s*f(2) + c*f(4)
      v(4) =-s*f(1) + c*f(3)
c
      return
      end
```