SANDIA REPORT
SAND2007-5873
Unlimited Release
Printed September 2007

Constitutive Models in LAME

William M. Scherzinger and Daniel C. Hammerand

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd.
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2007-5873
Unlimited Release
Printed September 2007

Constitutive Models in LAME

William M. Scherzinger and Daniel C. Hammerand
Solid Mechanics – 1524
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-MS0372

Abstract

The Library of Advanced Materials for Engineering (LAME) provides a common repository for constitutive models that can be used in computational solid mechanics codes. A number of models including both hypoelastic (rate) and hyperelastic (total strain) constitutive forms have been implemented in LAME. Descriptions of the structure and testing of LAME reside in other reports, while this report details the material models that have been implemented thus far.

ACKNOWLEDGMENTS

The authors would like to acknowledge the help of a number of people at Sandia National Laboratories. The Adagio and Presto code development teams, including Arne Gullerud, Kendall Pierson, Jason Hales and Nathan Crane have been especially helpful in the development of LAME and the interface between Strumento and LAME. William Gilmartin wrote a large part of the initial implementation of LAME and its interface in Adagio and Presto. The SNTools team, especially Mark Hamilton and Kevin Brown, have helped with code management issues. A number of constitutive model developers, including Bob Chambers, Mike Neilsen and Shane Schumacher, have given very useful feedback on the design of LAME. Finally, analysts that have been willing to use LAME, Jeff Gruda, Matthew Neidigk and Frank Dempsey, have also helped guide its design.

CONTENTS

1. INTRODUCTION

Any modern structural mechanics finite element code whether it is an explicit transient dynamics code such as Presto [1] or a quasi-static code like Adagio [2] can be described as a number of interlinked parts. For instance, Adagio has routines associated with element calculations such as determining the strains from the gradient of the displacements and the internal forces from the divergence of the stresses. Another part of Adagio is concerned primarily with the enforcement between contact of various parts in a model. Still yet another portion of Adagio is concerned primarily with the numerical solution of the resulting governing equations. A critical part of Adagio as well as any solid mechanics code is the portion where the constitutive response of the material models is computed. A recent change to Adagio and Presto has grouped the material models into a library called LAME. Having such a library where all most of the calculations involved with various material models is sectioned-off from, but still interfaced with the rest of the Adagio/Presto coding allows the material model developers to focus on their portion of the coding tasks in a more efficient manner. Also, the same material models contained in LAME are used in several codes. Currently this includes both Adagio and Presto. Further plans call for using these same models in Salinas. The importance of this can not be overstated. Many of our mechanics problems concern the structural response across a number of loading regimes. Being able to use the same finite element model with exactly the same material models in each code can only increase our confidence in the results in the consistency of each analysis. Finally, employing LAME across our solid mechanics codes results in the analysts having access to the latest version of each model incorporating the latest bug fixes no matter which solid mechanics code they are running for their analysis.

The structure and testing of LAME is described Scherzinger and Hammerand ([3] and [4]). The purpose of the present report is to describe the material models which have already been implemented into LAME. The descriptions are designed to give useful information to both analysts and code developers. Thus far, 33 non-ITAR/non-CRADA protected material models have been incorporated. These include everything from the simple isotropic linear elastic models to a number of elastic-plastic models for metals to models for honeycomb, foams, potting epoxies and rubber. A complete description of each model is outside the scope of the current report. Rather, the aim here is to delineate the properties, state variables, functions, and methods for each model. However, a brief description of some of the constitutive details is provided for a number of the material models. Where appropriate, the SAND reports available for each model have been cited. Many models have state variable aliases for some or all of their state variables. These alias names can be used for outputting desired quantities. The state variable aliases available for results output have been listed in this report. However, not all models use these aliases. For those models, no state variable names are listed. Nevertheless, the number of state variables employed by each model is always given.

Currently, there are four possible functions for a material model. This report lists which of these four methods are employed in each material model. As far as analysts are concerned, this information is included only for the awareness purposes. The analyst can take confidence in the fact that model has been properly implemented and the methods necessary for achieving accurate and efficient solutions have been incorporated. The most important method is the **`getStress`**

function where the actual material model evaluation takes place. Obviously, all material models incorporate this function. The **`initialize`** function is included in most material models. The **`initialize`** function is called once at the beginning of an analysis and its primary purpose is to initialize the material state variables associated with the model. Many times, there is some information which can be set once per load step. For instance, we may have temperature dependent material properties in an analysis where temperature is prescribed. Instead of setting those parameters at each iteration in a time step, it is much more efficient to set them once per time step at the beginning of the step. These types of load step initializations are performed in the **`loadStepInit`** method. The final function used by many models is the **`pcElasticModuli`** method which changes the moduli that are to be used by the elastic preconditioner in Adagio. The moduli for the elastic preconditioner are set during the initialization of Adagio. Sometimes, better convergence can be achieved by changing these moduli for the elastic preconditioner. For instance, it typically helps to modify the preconditioner when the material model has temperature dependent moduli. For many material models, it is not necessary to change the values of the moduli that are set initially in the code. Hence, those models do not have **`pcElasticModuli`** functions. All four of these methods receive information from the **`matParams`** structure as described by Scherzinger and Hammerand [3].

2. CONSTITUTIVE MODELS

2.1 BCJ Model

The **BCJ** model is a plasticity model that was developed at SNL/CA. Information on the **BCJ** model can be found in reference [5].

This constitutive law is a state variable model used to simulate the finite deformation behavior of metals. It uses a multiplicative decomposition of the deformation gradient into elastic, volumetric plastic and deviatoric parts. The model considers the natural configuration defined by this decomposition and its associated thermodynamics. The model is also capable of modeling strain rate and temperature sensitivity along with damage evolution.

Properties:

YOUNGS_MODULUS	C17
POISSONS_RATIO	C18
C1	C19
C2	C20
C3	DAMAGE_EXPONENT
C4	THETA_OPT
C5	FACTOR
C6	RHO
C7	SPECIFIC_HEAT
C8	INITIAL_ALPHA_XX
C9	INITIAL_ALPHA_YY
C10	INITIAL_ALPHA_ZZ
C11	INITIAL_ALPHA_XY
C12	INITIAL_ALPHA_YZ
C13	INITIAL_ALPHA_ZX
C14	INITIAL_KAPPA
C15	INITIAL_DAMAGE
C16	TEMP0

State Variables (14):

Functions:

YOUNGS_MODULUS_FUNCTION
POISSONS_RATIO_FUNCTION

Methods:

```
initialize( matParams * p );
loadStepInit( matParams * p );
getStress( matParams * p );
```

2.2 BCJ MEM Model

The **BCJ_MEM** model is a plasticity model that was developed at SNL/CA and is a variation of the **BCJ** model. Information on the **BCJ_MEM** model can be found in reference [5].

This constitutive law is a state variable model used to simulate the finite deformation behavior of metals. It uses a multiplicative decomposition of the deformation gradient into elastic, volumetric plastic and deviatoric parts. The model considers the natural configuration defined by this decomposition and its associated thermodynamics. The model is also capable of modeling strain rate and temperature sensitivity along with damage evolution.

Properties:

YOUNGS_MODULUS
POISSONS_RATIO
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12
C13
C14
C15
C16
C17
C18
C19
C20
C21
C22
C23
C24
C25
C26
C27
C28
C29
C30
C31
DAMAGE_EXPONENT
BZ
SMZ
CZ
FX
CXA
CXB
HZ
RZ
INITIAL_ALPHA_XX
INITIAL_ALPHA_YY
INITIAL_ALPHA_ZZ
INITIAL_ALPHA_XY
INITIAL_ALPHA_YZ
INITIAL_ALPHA_ZX
INITIAL_KAPPA
INITIAL_GRAIN_SIZE
INITIAL_REX_VOL_FRAC
INITIAL_ZETA
INITIAL_DAMAGE

State Variables (27):

Functions:

none

Methods:

initialize(matParams * p);
loadStepInit(matParams * p);
getStress(matParams * p);

2.3 Ductile Fracture Model

The **DUCTILE_FRACTURE** model is a plasticity model based on the power law hardening model (see **EP_POWER_LAW**) that calculates a failure parameter.

The hardening law for the plasticity model is a power law fit

$$\bar{\sigma} = \sigma_y + A\left\langle \bar{\varepsilon}_p - \varepsilon_L \right\rangle^n \ , \tag{1}$$

where $\bar{\sigma}$ is the von Mises stress, σ_y is the yield stress, $\bar{\varepsilon}_p$ is the equivalent plastic strain, ε_L is the Lüders strain, A is the hardening constant, n is the hardening exponent, and $\langle \bullet \rangle$ denotes the Heaviside step function.

Two other parameters, the critical crack opening strain and the critical tearing parameter, describe the failure of the material. The critical tearing parameter, t_p, is given by

$$t_p = \int_0^{\varepsilon_f} \left\langle \frac{2\sigma_{\max}}{3(\sigma_{\max} - \sigma_m)} \right\rangle^4 d\varepsilon_p \ , \tag{2}$$

where $\sigma_{\max}$ is the maximum principal stress, σ_m is the mean stress and $\langle \bullet \rangle$ denotes the Heaviside step function.

There are no user input functions for this model.

Properties:

LAMBDA
SHEAR_MODULUS
YIELD_STRESS
HARDENING_CONSTATN
HARDENING_EXPONENT
LUDERS_STRAIN
CRITICAL_TEARING_PARAMETER
CRITICAL_CRACK_OPENING_STRAIN

State Variables (8):

Functions:

none

Methods:

initialize(matParams * p);
getStress(matParams * p);

2.4 Elastic Model

The **ELASTIC** model is the simplest constitutive model in LAME. This model is finite deformation, hypoelastic constitutive model that is an extension of linear elasticity. The stress rate is related to the rate of deformation by

$$\hat{\sigma}_{ij} = \lambda \delta_{ij} D_{kk} + 2\mu D_{ij} \ , \tag{3}$$

where λ and μ are the Lamé constants, D_{ij} are the components of the rate of deformation and the stress rate, $\hat{\sigma}_{ij}$, is an objective stress rate. In Adagio and Presto, this is the Green-McInnis rate.

The input for the model is the **YOUNGS_MODULUS** and the **POISSONS_RATIO**. The Lamé constants can be calculated from these parameters.

$$\begin{aligned} \lambda &= \frac{E\nu}{(1-2\nu)(1+\nu)} \\ \mu &= \frac{E}{2(1+\nu)} \end{aligned} \tag{4}$$

There are no state variables or user input functions for this model.

Properties:

YOUNGS_MODULUS
POISSONS_RATIO

State Variables:

none

Functions:

none

Methods:

getStress(matParams * p);

2.5 Elastic-Plastic Model

The **ELASTIC_PLASTIC** model is an elastic-plastic, linear hardening model. A description of the model can be found in [6].

The model is used to model metal plasticity where the hardening curve is described with a linear fit. The hardening curve is given by

$$\bar{\sigma} = \sigma_y + H'\bar{\varepsilon}_p \ , \tag{5}$$

where $\bar{\sigma}$ is the von Mises stress, σ_y is the yield stress, $\bar{\varepsilon}_p$ is the equivalent plastic strain and H' is the hardening modulus. This is the simplest model for metal plasticity, but unfortunately very few materials follow linear hardening curves. This model owes its popularity to its easy implementation (the radial return algorithm requires no iterations) and to the fact that it is the simplest plasticity model to model hardening. It is useful for quick scoping studies and some analytical problems where one wants to take into account the effects of plasticity.

The model is also capable of modeling kinematic hardening through the parameter **BETA**. If **BETA** is equal to one then the hardening is isotropic, i.e. the center of the yield surface is fixed, while if **BETA** is equal to zero the hardening is kinematic, i.e. the center of the yield surface moves. Values between zero and one involve a combination of kinematic and isotropic hardening. This is especially important in modeling problems that have cyclic loading.

There are eight state variables for this model: the equivalent plastic strain, the radius of the yield surface and the six components of the back-stress tensor that defines the center of the yield surface.

There are no user input functions for this model.

Properties:

YOUNGS_MODULUS
POISSONS_RATIO
YIELD_STRESS
HARDENING_MODULUS
BETA

State Variables (8):

EQPS
ALPHA_XX
ALPHA_YY
ALPHA_ZZ
ALPHA_XY
ALPHA_YZ
ALPHA_ZX
RADIUS

Functions:

none

Methods:

initialize(matParams * p);
getStress(matParams * p);

2.6 Elastic-Plastic Power Law Hardening Model

The **EP_POWER_HARD** model is an elastic-plastic, power law hardening model. A description of the model can be found in [6] and [7].

The model is used to model metal plasticity where the hardening curve is described by a power law fit. The hardening curve is given by

$$\bar{\sigma} = \sigma_y + A\left\langle \bar{\varepsilon}_p - \varepsilon_L \right\rangle^n , \tag{6}$$

where $\bar{\sigma}$ is the von Mises stress, σ_y is the yield stress, $\bar{\varepsilon}_p$ is the equivalent plastic strain, ε_L is the Lüders strain, A is the hardening constant, n is the hardening exponent, and $\langle \bullet \rangle$ denotes the Heaviside step function. This is a widely used model for metal plasticity and there are a number of parameter fits in the literature for various materials. This particular implementation of the model does not model kinematic hardening – a capability that could easily be added.

The model has two state variables: the equivalent plastic strain and the radius of the yield surface.

There are no user input functions for this model.

Properties:

YOUNGS_MODULUS
POISSONS_RATIO
YIELD_STRESS
HARDENING_CONSTANT
HARDENING_EXPONENT
LUDERS_STRAIN

State Variables (2):

EQPS
RADIUS

Functions:

none

Methods:

initialize(matParams * p);
getStress(matParams * p);

2.7 Elastic Fracture Model

The **ELASTIC_FRACTURE** model is used to model brittle fracture/failure. The model uses a maximum-principal-stress failure criterion. The stress decays isotropically based on the component of strain parallel to the maximum principal stress. The value of the component of strain over which the stress is decayed to zero is a user-defined parameter for the model. This strain parameter can be adjusted so that failure is mesh independent.

The model has six state variables, none of which are aliased for output.

There are no user input functions for this model.

Properties:

YOUNGS_MODULUS
POISSONS_RATIO
MAX_STRESS
CRITICAL_STRAIN

State Variables (6):

Functions:

none

Methods:

getStress(matParams * p);

2.8 Foam Plasticity Model

The **FOAM_PLASTICITY** model is used to model polyurethane foams. The model is particularly useful for modeling failure of polyurethane foams [8]. The model has a pressure dependent yield function and an associated flow rule. Therefore the integration of this model can be quite complex.

The yield surface looks like

$$\phi = \frac{\sigma_m^2}{a^2} + \frac{\sigma_s^2}{b^2} - 1 \ , \tag{7}$$

where σ_m is the mean stress, σ_s is the von Mises stress, a and b are the hydro strength and shear strength respectively. There are also hardening constants and exponents for both the hydro strength and the shear strength.

Properties:

YOUNGS_MODULUS
POISSONS_RATIO
PHI
SHEAR_STRENGTH
SHEAR_HARDENING
SHEAR_EXPONENT
HYDRO_STRENGTH
HYDRO_HARDENING
HYDRO_EXPONENT
BETA

State Variables (6):

Functions:

none

Methods:

initialize(matParams * p);
getStress(matParams * p);

2.9 Honeycomb Model

The **HONEYCOMB** model is used to model reinforced aluminum honeycomb [9]. The model is orthotropic with the three principal axes of orthotropy labeled the T, L and W axes. This model is the latest in an evolution of models that include the orthotropic crush and orthotropic rate models. For a complete description of the constitutive model, along with procedures for fitting test data to the model, the reader is directed to [9].

Properties:

LAMBDA	C1
SHEAR_MODULUS	C2
YIELD_STRESS	C3
MODULUS_TTTT	TS
MODULUS_TTLL	LS
MODULUS_TTWW	WS
MODULUS_LLLL	TLS
MODULUS_LLWW	LWS
MODULUS_WWWW	WTS
A1	ESTW
A2	ESTL
A3	ESLT
B1	ESLW
B2	ESWT
B3	ESWL

State Variables (39):

Functions:

MODULUS_FUNCTION	TLTLP_FUNCTION
RATE_FUNCTION	LWLWP_FUNCTION
T_FUNCTION	WTWTP_FUNCTION
L_FUNCTION	TTP_FUNCTION
W_FUNCTION	LLP_FUNCTION
TL_FUNCTION	WWP_FUNCTION
LW_FUNCTION	TTLP_FUNCTION
WT_FUNCTION	TTWP_FUNCTION

Methods:

```
initialize( matParams * p );
getStress( matParams * p );
```

2.10 Hyperfoam Model

The **HYPERFOAM** model is a hyperelastic model that is used to model foams. It is based on an Ogden [10] type model and is the same as the Hyperfoam model that is in ABAQUS [11].

The stress in this model is derived from a strain energy density that is only dependent on the principal stretch ratios. As a result, it is simple to derive the principal stresses from the strain energy density. The principal stresses can be converted back into the global coordinates. The strain energy density for this model is

$$W\left(\lambda_1,\lambda_2,\lambda_3\right)=\sum_{k=1}^{N}\frac{2\mu_k}{\alpha_k^2}\left[\lambda_1^{\alpha_k}+\lambda_2^{\alpha_k}+\lambda_3^{\alpha_k}-3+\frac{1}{\beta_i}\left(J^{-\alpha_k\beta_k}-1\right)\right], \tag{8}$$

where the λ_i are the principal stretch ratios and J is the Jacobian of the deformation. The material parameters are μ_k, α_k and ν_k with

$$\beta_k=\frac{\nu_k}{1-2\nu_k}\ . \tag{9}$$

The number of material properties depends on the number of terms, N, in the description of the strain energy density. The principal Cauchy stresses are given by

$$\sigma_I=\frac{\lambda_I}{J}\frac{\partial W}{\partial\lambda_I}\quad\left(\text{no sum on } I\right)\ . \tag{10}$$

Since the model is hyperelastic, if the host code is storing the stress components in another configuration (e.g. the un-rotated configuration), then the stress components must be converted to that configuration prior to sending them back to the host code.

Properties:

N
SHEAR
ALPHA
POISSON

State Variables:

none

Functions:

none

Methods:

getStress(matParams * p);

2.11 Incompressible Solid Model

The **INCOMPRESSIBLE_SOLID** model is a variation of the **ELASTIC** model that is used with Adagio's multilevel solver. The model is used to model nearly incompressible materials (where $\nu \approx 0.5$) in a quasistatic code, e.g. Adagio. The model is essentially the same as the **ELASTIC** model except that it scales the bulk modulus and/or the shear modulus. The model needs some way to account for the scaled response in the host code, e.g. it uses the multi-level solution control and augmented Lagrange wrappers in Adagio. This model is not intended for transient dynamics applications, e.g. Presto. However, if it is used in Presto, the scaling terms are ignored.

There are no state variables or user input functions for this model.

Properties:

YOUNGS_MODULUS
POISSONS_RATIO
BULK_SCALING
SHEAR_SCALING

State Variables:

none

Functions:

none

Methods:

getStress(matParams * p);
pcElasticModuli(matParams * p);

2.12 Johnson-Cook Model

The **JOHNSON_COOK** model is a rate and temperature dependent plasticity model. This model is generally used for high velocity impact calculations. The rate and temperature dependence is included in the hardening law as follows:

$$\bar{\sigma} = \left(\sigma_y + A\bar{\varepsilon}_p^n\right)\left(1 + C\ln\dot{\varepsilon}^*\right)\left(1 - T^{*m}\right) , \tag{11}$$

where $\bar{\sigma}$ is the von Mises stress, $\bar{\varepsilon}_p$ is the equivalent plastic strain, A is the hardening constant, n is the hardening exponent, C is the rate constant, $\dot{\varepsilon}^*$ is a normalized plastic strain rate, T^* is the effective temperature and m is the thermal exponent. The normalized plastic strain rate is defined as

$$\dot{\varepsilon}^* = \frac{\dot{\varepsilon}^p}{\dot{\varepsilon}_0} , \tag{12}$$

where $\dot{\varepsilon}^p$ is the equivalent plastic strain rate and $\dot{\varepsilon}_0$ is a reference strain rate. In the code $\dot{\varepsilon}_0$ is hard-coded to a value of 0.001. The effective temperature is given by

$$T^* = \frac{T - T_0}{T_m - T_0} , \tag{13}$$

where T is the temperature, T_0 is a reference temperature and T_m is the melt temperature.

There are five state variables for this model and three of them are aliased for output: the radius of the yield surface, the equivalent plastic strain and the equivalent plastic strain rate.

There are no user input functions for this model.

Properties:

YOUNGS_MODULUS
POISSONS_RATIO
YIELD_STRESS
HARDENING_CONSTANT
HARDENING_EXPONENT
RHOCV
RATE_CONSTANT
THERMAL_EXPONENT
REFERENCE_TEMPERATURE
MELT_TEMPERATURE

State Variables (5):

RADIUS
EQPS
EQDOT

Functions:

none

Methods:

getStress(matParams * p);

2.13 Low Density Foam Model

The **LOW_DENSITY_FOAM** model is a phenomenological model that is used to model low density polyurethane foams. This model was developed at Sandia National Laboratories and complete documentation can be found in [12].

The yield function for this model has the form

$$\bar{\sigma} = A\langle I_2' \rangle + B\left(1 + CI_1\right) , \tag{14}$$

where A, B and C are material properties, $\langle \bullet \rangle$ denotes the Heaviside step function, I_2' is the second invariant of the deviatoric strain and I_1 is the first invariant of the strain.

There are eight state variables for this model, only one of which is aliased for output. There are no user input functions for this model.

Properties:

YOUNGS_MODULUS
A
B
C
NAIR
P0
PHI

State Variables (8):

PAIR

Functions:

none

Methods:

getStress(matParams * p);

2.14 Mooney-Rivlin Model

The **MOONEY_RIVLIN** model is a hyperelastic model that is used to model rubber. The strain energy density as implemented in Adagio is given by

$$U = C_{10}(\bar{I}_1 - 3) + C_{01}(\bar{I}_2 - 3) + K(J_m \ln J_m - J_m) \tag{15}$$

where C_{10}, C_{01}, and K are temperature dependent material constants and the strain quantities are defined as follows. The strain tensor is nominally the left Cauchy-Green strain tensor excluding the volumetric change given as

$$\left[\bar{B}\right] = \left[\bar{F}\right]\left[\bar{F}\right]^T \tag{16}$$

where $\left[\bar{F}\right]$ is defined in terms of the total deformation gradient $\left[F\right]$ as follows:

$$\left[\bar{F}\right] = J^{-\frac{1}{3}}\left[F\right] \tag{17}$$

with J defining the relative volumetric change given in terms of differential volume dV as follows:

$$J = \det\left[F\right] = \frac{dV(t)}{dV(0)} \tag{18}$$

The first and second strain invariants are simply

$$\bar{I}_1 = \left[I\right] : \left[\bar{B}\right] \tag{19}$$

and

$$\bar{I}_2 = \frac{1}{2}\left(\bar{I}_1^{\,2} - \left[I\right] : \left(\left[\bar{B}\right]\left[\bar{B}\right]\right)\right) \tag{20}$$

The mechanical volumetric strain is given by

$$J_m = \frac{J}{J_{th}} \tag{21}$$

where the thermal volumetric strain is given by

$$J_{th} = \det\left[F_{th}\right] \tag{22}$$

with $[F_{th}]$ defined as the thermal deformation gradient. That is, the total deformation gradient is multiplicatively decomposed as

$$[F] = [F_m][F_{th}] \tag{23}$$

with isotropic thermal expansion defined by

$$[F_{th}] = J_{th}^{1/3}[I] \tag{24}$$

The deviatoric and volumetric stresses are respectively given as

$$[\sigma'] = \frac{2}{J_m} DEV\left[(C_{10} + \bar{I}_1 C_{01})\left[\bar{B}\right] - C_{01}\left[\bar{B}\right]\left[\bar{B}\right]\right] \tag{25}$$

and

$$p = K \ln(J_m) \tag{26}$$

Properties:

C10
C01
BULK_MODULUS
TARGET_E
MAX_POISSONS_RATIO
BULK_SCALING
SHEAR_SCALING

State Variables (12):

C10
C01
K
SFJTH
JTH
VMECHXX
VMECHYY
VMECHZZ
VMECHXY
VMECHYZ
VMECHZX
SFJTH_FLAG

Functions:

TARGET_E_FUNCTION
C10_FUNCTION
C01_FUNCTION
BULK_FUNCTION
THERMAL_EXPANSION_FUNCTION

Methods:

initialize(matParams * p);
getStress(matParams * p);
loadStepInit(matParams * p);
pcElasticModuli(matParams * p);

2.15 Multilinear Elastic-Plastic Model

The **MULTILINEAR_EP** model is an elastic-plastic model with a piecewise linear hardening curve. This model provides significant flexibility for an analyst in describing the hardening behavior of an elastic-plastic material. A hardening function, $H\left(\bar{\varepsilon}_p\right)$, describes equivalent stress – equivalent plastic strain pairs that are used to describe the hardening behavior for the material. The hardening curve for the model is given by

$$\bar{\sigma} = \sigma_y + H\left(\bar{\varepsilon}_p\right) , \tag{27}$$

where $\bar{\sigma}$ is the von Mises stress and σ_y is the yield stress.

In addition to supporting arbitrary hardening curves, the model allows for isotropic and/or kinematic hardening for the yield surface, temperature dependent elastic properties and a temperature dependent yield stress. The hardening curve, however, does not change shape with temperature.

There are 11 state variables for this model, eight of which are aliased for output. The other three correspond to temperature dependent properties. There are four user input functions for this model.

Properties:

YOUNGS_MODULUS
POISSONS_RATIO
YIELD_STRESS
BETA

State Variables (11):

EQPS
RADIUS
ALPHA_XX
ALPHA_YY
ALPHA_ZZ
ALPHA_XY
ALPHA_YZ
ALPHA_ZX

Functions:

YOUNGS_MODULUS_FUNCTION
POISSONS_RATIO_FUNCTION
YIELD_STRESS_FUNCTION
HARDENING_FUNCTION

Methods:

initialize(matParams * p);
getStress(matParams * p);
loadStepInit(matParams * p);

2.16 Multilinear Elastic Plastic with Failure Model

The **ML_EP_FAIL** is a ductile failure model that uses the multilinear elastic-plastic model. The model has a piecewise linear hardening curve which provides significant flexibility for an analyst in describing the hardening behavior of an elastic-plastic material. A hardening function, $H\left(\bar{\varepsilon}_p\right)$, describes equivalent stress – equivalent plastic strain pairs that are used to describe the hardening behavior for the material. The hardening curve for the model looks like

$$\bar{\sigma} = \sigma_y + H\left(\bar{\varepsilon}_p\right) \ , \tag{28}$$

where $\bar{\sigma}$ is the von Mises stress and σ_y is the yield stress.

In addition to supporting arbitrary hardening curves, the model allows for isotropic and/or kinematic hardening for the yield surface, temperature dependent elastic properties and a temperature dependent yield stress. The hardening curve, however, does not change shape with temperature.

Two other parameters, the critical crack opening strain and the critical tearing parameter, describe the failure of the material. The critical tearing parameter, t_p, is given by

$$t_p = \int_0^{\varepsilon_f} \left\langle \frac{2\sigma_{\max}}{3\left(\sigma_{\max} - \sigma_m\right)} \right\rangle^4 d\varepsilon_p \ , \tag{29}$$

where $\sigma_{\max}$ is the maximum principal stress, σ_m is the mean stress and $\langle \bullet \rangle$ denotes the Heaviside step function.

There are 14 state variables for this model, eight of which are aliased for output. There are four user input functions for this model.

Properties:

YOUNGS_MODULUS
POISSONS_RATIO
YIELD_STRESS
BETA
CRITICAL_TEARING_PARAMETER
CRITICAL_CRACK_OPENING_STRAIN

State Variables (14):

EQPS
RADIUS
ALPHA_XX
ALPHA_YY
ALPHA_ZZ
ALPHA_XY
ALPHA_YZ
ALPHA_ZX

Functions:

YOUNGS_MODULUS_FUNCTION
POISSONS_RATIO_FUNCTION
YIELD_STRESS_FUNCTION
HARDENING_FUNCTION

Methods:

initialize(matParams * p);
getStress(matParams * p);
loadStepInit(matParams * p);

2.17 Neo-Hookean Model

The **NEO_HOOKEAN** model is a hyperelastic model based on small strain linear elasticity. If a large deformation elasticity model is needed, this is probably a better model than the **ELASTIC** model for that purpose. The model that is implemented in LAME is the same as the model in [13].

The strain energy density for the neo-Hookean model is given by

$$W(\mathbf{C}) = \frac{1}{4}K\left[\det\mathbf{C} - \ln(\det\mathbf{C}) - 1\right] + \frac{1}{2}\mu\left(\frac{\mathrm{tr}\mathbf{C}}{\det\mathbf{C}^{1/3}} - 3\right), \tag{30}$$

where K is the bulk modulus, μ is the shear modulus and $\mathbf{C}$ is the right Cauchy-Green tensor. The stress can be derived directly from the strain energy density. The components of the Cauchy stress, σ_{ij}, are

$$\sigma_{ij} = \frac{1}{2}K\delta_{ij}\left(J - \frac{1}{J}\right) + \mu\left(B_{ij} - \frac{1}{3}\delta_{ij}B_{kk}\right)J^{-5/3}, \tag{31}$$

where B_{ij} are the components of the left Cauchy-Green tensor and $J = \det\mathbf{F}$ is the Jacobian of the deformation. For small strains it is easy to show that (31) reduces to the expression for small strain linear elasticity.

There are no state variables or user input functions for this model.

Properties:

BULK_MODULUS
SHEAR_MODULUS

State Variables:

none

Functions:

none

Methods:

getStress(matParams * p);

2.18 NLVE Polymer Model

The **NLVE_POLYMER** model is a nonlinear viscoelastic material model for analyzing stresses and strains in filled and unfilled polymers. It employs a finite strain measure (Hencky strain $\underline{\underline{H}}$) and is thermodynamically consistent so in principle can be used to analyze material performance all the way up to failure. It predicts a full range of behavior including yielding, stress relaxation, volume relaxation, physical aging, enthalpy relaxation, etc. The model uses a material clock driven by the potential part of the internal energy. The code inputs allow the user to specify either the mathematical terms in the constitutive equation directly or a set of physically measurable quantities from which the equation terms are computed. This model is the rigorously complete model for filled and unfilled polymers. It is intended for research purposes. Analysts are advised to use the simpler, but still accurate **Universal_Polymer** model.

Complete details of this model are given in References [14,15]. However, some of the relevant equations are presented here. This model is derived using a Rational Mechanics approach where all thermodynamic quantities are derived from a single potential, the Helmholtz free energy, Ψ. The internal energy U in the model is related to the Helmholtz free energy as

$$U = \Psi + T\eta \tag{32}$$

where η is the entropy and T is the temperature. The full expression for the internal energy is given as follows:

$$\begin{aligned}
U = U_{\infty c} &+ \frac{1}{2}\Psi_1 \int_0^t \int_0^t ds\ \mathrm{du}\ \mathrm{f}_1\left(t^*-s^*, t^*-u^*\right)\frac{dI_H}{ds}(s)\frac{dI_H}{du}(u) \\
&+\Psi_2(T)\int_0^t \int_0^t ds\ \mathrm{du}\ \mathrm{f}_2\left(t^*-s^*, t^*-u^*\right)\ \frac{\mathrm{d}\underline{\underline{\mathrm{H}}}}{\mathrm{ds}} : \frac{\mathrm{d}\underline{\underline{\mathrm{H}}}}{\mathrm{du}} \\
&+\Psi_3(T, I_H)\int_0^t \int_0^t ds\ \mathrm{du}\ \mathrm{f}_3\left(t^*-s^*, t^*-u^*\right)\frac{dI_H}{ds}(s)\frac{dT}{du}(u) \\
&+\frac{1}{2}\Psi_4(T)\int_0^t \int_0^t ds\ \mathrm{du}\ \mathrm{f}_4\left(t^*-s^*, t^*-u^*\right)\frac{dT}{ds}(s)\frac{dT}{du}(u) \\
&-T\left[\Psi_4(T)\int_0^t ds\ \mathrm{f}_4\left(t^*-s^*, 0\right)\frac{dT}{dt}(s)+\Psi_3(T, I_H)\int_0^t ds\ \mathrm{f}_3\left(t^*-s^*, 0\right)\frac{dI_H}{ds}(s)\right. \\
&\quad +\left(\frac{\partial\Psi_3}{\partial T}\right)_{I_H}\int_0^t \int_0^t ds\ \mathrm{du}\ \mathrm{f}_3\left(t^*-s^*, t^*-u^*\right)\frac{dI_H}{ds}(s)\frac{dT}{du}(u) \\
&\quad +\frac{1}{2}\left(\frac{\partial\Psi_4}{\partial T}\right)_{I_H}\int_0^t \int_0^t ds\ \mathrm{du}\ \mathrm{f}_4\left(t^*-s^*, t^*-u^*\right)\frac{dT}{ds}(s)\frac{dT}{du}(u) \\
&\quad \left.+\left(\frac{\partial\Psi_2}{\partial T}\right)_{I_H}\int_0^t \int_0^t ds\ \mathrm{du}\ \mathrm{f}_2\left(t^*-s^*, t^*-u^*\right)\ \frac{\mathrm{d}\underline{\underline{\mathrm{H}}}}{\mathrm{ds}} : \frac{\mathrm{d}\underline{\underline{\mathrm{H}}}}{\mathrm{du}}\right]
\end{aligned} \tag{33}$$

with the material clock defined by

$$t^* - s^* = \int_s^t \frac{dw}{a(w)} \qquad \text{and} \qquad \log\ \mathrm{a} = \mathrm{B}\left(\frac{1}{U_c} - \frac{1}{U_c^{ref}}\right) = \mathrm{C}_1\left(\frac{U_c^{ref}}{U_c} - 1\right) \tag{34}$$

and the equilibrium energy given by

$$\begin{aligned} U_{\infty_C}\left(I_H, \Delta T\right) = \; & U_{\infty_{pot}}^{ref} + \frac{1}{2}\psi_{I\,1} I_H^2 + \psi_{HH} II_H - \Psi_3^{ref}\ I_H\ \Delta T - \frac{1}{2}\Psi_4^{ref}\Delta T^2 - \left(\frac{\partial \Psi_3}{\partial I_H}\right)^{ref} I_H^2 \Delta T \\ & - \left(\frac{\partial \Psi_3}{\partial T}\right)^{ref} I_H \Delta T^2 - \frac{1}{2}\left(\frac{\partial \Psi_4}{\partial T}\right)^{ref} \Delta T^3 - \frac{1}{4}\left(\frac{\partial^2 \Psi_4}{\partial T^2}\right)^{ref} \Delta T^4 \\ & + \mathrm{T}\left[\begin{array}{l} \Psi_4^{ref}\Delta T + 2\left(\frac{\partial \Psi_3}{\partial T}\right)^{ref} I_H \Delta T + \frac{3}{2}\left(\frac{\partial \Psi_4}{\partial T}\right)^{ref} \Delta T^2 \\ \qquad + \left(\frac{\partial^2 \Psi_4}{\partial T^2}\right)^{ref} \Delta T^3 + \Psi_3^{ref}\ I_H + \left(\frac{\partial \Psi_3}{\partial I_\varepsilon}\right)^{ref} I_H^2 \end{array}\right] \end{aligned} \tag{35}$$

The Hencky stress $\underline{\underline{S}}_H$ is the stress measure that is work conjugate with the Hencky strain $\underline{\underline{\mathrm{H}}}$ and is given by

$$\begin{aligned} \underline{\underline{S}}_H = \; & \rho_{ref}\ \Psi_1 \int_0^t ds\ \mathrm{f}_1\left(t^* - s^*\right) \frac{d\mathrm{I}_H}{ds}(s)\ \underline{\underline{\mathrm{I}}} + 2\rho_{ref}\ \Psi_2\left(T\right) \int_0^t ds\ \mathrm{f}_2\left(t^* - s^*\right)\ \frac{d\underline{\underline{H}}}{ds}(s) \\ & + \rho_{ref}\ \Psi_3(T, \mathrm{I}_H) \int_0^t ds\ \mathrm{f}_3\left(t^* - s^*\right) \frac{dT}{ds}(s)\ \underline{\underline{\mathrm{I}}} \\ & + \rho_{ref} \left(\frac{\partial \Psi_3}{\partial \mathrm{I}_H}\right)_T \int_0^t ds \int_0^t du\ \mathrm{f}_3\left(t^* - s^*, t^* - u^*\right) \frac{dT}{ds}(s) \frac{d\mathrm{I}_H}{ds}(u)\ \underline{\underline{\mathrm{I}}} \\ & + \rho_{ref}\left[\psi_{\mathrm{I}}\ \underline{\underline{I}} + \psi_{\mathrm{II}}\ I_H + \psi_{\mathrm{IT}}\ \Delta T + \psi_{\mathrm{IIT}} I_H\ \Delta T + \frac{1}{2}\psi_{\mathrm{ITT}} \Delta T^2\right] \underline{\underline{I}} + 2\ \rho_{ref}\psi_{HH}\ \underline{\underline{H}} \end{aligned} \tag{36}$$

The Hencky stress is converted internally to the Cauchy stress for use in static and dynamic equilibrium equations. The relaxation functions are specified using Prony series as follows:

$$f_m(t) = \sum_{i=1}^{N} f_{mi}\ e^{-t/\tau_i} \tag{37}$$

Properties:

There are 253 properties that are input for this model. Some of these properties are input, but some are not and are simply computed from some of the other inputs. For example, the relaxation functions are usually input as Williams-Watts functions and then Prony series are determined from them. Since many of the inputs comprise a Prony series, only the name of the first term in each series is explicitly given here.

1PSI PRONY 1, 1PSI PRONY 2, etc: Property Numbers 1-30 are the exponential series prefactors (f_{1i}) that are used to define the normalized relaxation function f_1 associated with the Ψ_1 integral in the equation for $\underline{\underline{S}}_H$

2PSI PRONY 1, 2PSI PRONY 2, etc: Property Numbers 31-60 are the exponential series prefactors that are used to define the normalized relaxation function f_2 associated with the Ψ_2 integral in the equation for $\underline{\underline{S}}_H$

3PSI PRONY 1, 3PSI PRONY 2, etc: Property Numbers 61-90 are the exponential series prefactors that are used to define the normalized relaxation function f_3 associated with the Ψ_3 integral in the equation for $\underline{\underline{S}}_H$

4PSI PRONY 1, 4PSI PRONY 2, etc: Property Numbers 91-120 are the exponential series prefactors that are used to define the normalized relaxation function f_4 associated with the Ψ_4 integral in the equation for $\underline{\underline{S}}_H$

RELAX TIME 1, RELAX TIME 2, etc: Property Numbers 121-150 are the exponential series relaxation times (τ_i) that are used to define all the normalized relaxation functions in the equation for $\underline{\underline{S}}_H$

$$\Psi_1(I_H) = \Psi_1^{ref} + \frac{d\Psi_1}{dI_H} I_H$$

1PSI REF, Ψ_1^{ref}

I1 DERIV 1PSI, $\frac{d\Psi_1}{dI_H}$

$$\Psi_2(T, I_H, I_{HH}) = \Psi_2^{ref} + \frac{d\Psi_2}{dT}(T - T_{REF}) + \frac{d\Psi_1}{dI_H} I_H + \frac{d\Psi_1}{dI_{HH}} I_{HH} + \frac{1}{2}\frac{d^2\Psi_1}{dI_{HH}^2} I_{HH}^2$$

2PSI REF, Ψ_2^{ref}

T DERIV 2PSI, $\frac{d\Psi_2}{dT}$

I1 DERIV 2PSI, $\frac{d\Psi_2}{dI_H}$

I2 DERIV 2PSI, $\frac{d\Psi_2}{dI_{HH}}$, (I_{HH} is the 2nd Hencky strain invariant)

I2 2DERIV 2PSI, $\frac{d^2\Psi_2}{dI_{HH}^2}$

$$\Psi_3(T,I_H) = \Psi_3^{ref} + \frac{d\Psi_3}{dT}(T - T_{REF}) + \frac{d\Psi_3}{dI_H} I_H + \frac{d^2\Psi_3}{dT dI_H} I_H (T - T_{REF})$$

3PSI REF, Ψ_3^{ref}

T DERIV 3PSI, $\frac{d\Psi_3}{dT}$

I1 DERIV 3PSI, $\frac{d\Psi_3}{dI_H}$

I1T 2DERIV 3PSI, $\frac{d^2\Psi_3}{dI_H dT}$

$$\Psi_4(T,I_H) = \Psi_4^{ref} + \frac{d\Psi_4}{dT}(T - T_{REF}) + \frac{d\Psi_4}{dI_H} I_H + \frac{1}{2}\frac{d^2\Psi_4}{dT^2}(T - T_{REF})^2$$

4PSI REF, Ψ_4^{ref}

T DERIV 4PSI, $\frac{d\Psi_4}{dT}$

T 2DERIV 4PSI, $\frac{d^2\Psi_4}{dT^2}$

I1 DERIV 4PSI, $\frac{d\Psi_4}{dI_H}$

Reference temperature, T_{ref}

Reference density

WLF coefficient C1

WLF coefficient C2

B, Shift factor constant

U_c^{ref}, Shift factor constant

$$f_1(t) = \exp\left[-\left(\frac{t}{\tau_1}\right)^{\beta_1}\right]$$

WWBETA 1PSI, β_1

WWTAU 1PSI, τ_1

$$f_2(t) = \exp\left[-\left(\frac{t}{\tau_2}\right)^{\beta_2}\right]$$

WWBETA 2PSI, β_2
WWTAU 2PSI, τ_2

$$f_3(t) = \exp\left[-\left(\frac{t}{\tau_3}\right)^{\beta_3}\right]$$

WWBETA 3PSI, β_3
WWTAU 3PSI, τ_3

$$f_4(t) = \exp\left[-\left(\frac{t}{\tau_4}\right)^{\beta_4}\right]$$

WWBETA 4PSI, β_4
WWTAU 4PSI, τ_4
DOUBLE INTEG FACTOR (not used)
RUBBERY BULK MOD
I1 DERIV R_BULK
GLASSY BULK MOD
I1 DERIV G_BULK
RUBBERY SHEAR MOD
T DERIV R_SHEAR
I2 DERIV R_SHEAR
GLASSY SHEAR MOD
T DERIV G_SHEAR
RUBBERY VOL CTE
T DERIV R_CTE
GLASSY VOL CTE
T DERIV G_CTE
RUBBER HCAPACITY
T DERIV R_HCAPACITY
GLASSY HCAPACITY
T DERIV G_HCAPACITY
GLASS TRANSITION TEM
TG TEST PRESSURE
SHIFTED TG VALUE
ENERGY OPTION (0= temp specified; 1=adiabatic, compute temperature)
PSI EQ 2I, Ψ_{II}, (These terms are used in the Taylor series
PSI EQ IT, Ψ_{IT} expansion that defines the equilibrium
PSI EQ 2T, Ψ_{TT} stress)
PSI EQ 2H, Ψ_{HH}
PSI EQ 3I, Ψ_{III}
PSI EQ 2IT, Ψ_{IIT}
PSI EQ I2T, Ψ_{ITT}
PSI EQ 3T, Ψ_{TTT}
PSI EQ 2HT, Ψ_{HHT}

PSI EQ 4I, Ψ_{IIII}
PSI EQ 3IT, Ψ_{IIIT}
PSI EQ 2I2T, Ψ_{IITT}
PSI EQ I3T, Ψ_{ITTT}
PSI EQ 4T, Ψ_{TTTT}
PSI EQ 4H, Ψ_{HHHH}
PSI POT IT
PSI POT 2T
PSI POT 2IT
PSI POT I2T
PSI POT 3T
PSI POT 4T
PSI POT I
PSI POT 2I
PSI POT 3I
PSI POT 2H
PSI POT 2HT
1PSI POT FACTOR
2PSI POT FACTOR
3PSI POT FACTOR
4PSI POT FACTOR
PSI POT 2I2T

State Variables (281):

Functions:
none

Methods:
initialize(matParams * p);
getStress(matParams * p);

2.19 Orthotropic Crush Model

The **ORTHOTROPIC_CRUSH** model is useful for modeling energy absorbing materials like aluminum honeycomb. It is a fairy coarse model that gives reasonable results when the loading is aligned with the principal axes of orthotropy of the material. One restriction for this material is that the axes of orthotropy must be aligned with the global Cartesian axes.

Fully compacted elastic properties and a yield stress are given for the material. In addition to these properties, initial orthotropic elastic moduli and shear moduli are given with respect to the global Cartesian axes. Six functions are defined that give the crush strength of the material as a function of volumetric strain for the six stress components. This models the plateau strength for the material.

There is one state variable for this model, the volumetric strain. There are six user input functions corresponding to the crush strength curves for the six stress components.

Properties:

YOUNGS_MODULUS
POISSONS_RATIO
YIELD_STRESS
EX
EY
EZ
GXY
GYZ
GZX
VMIN

State Variables (1):

0 - EVOL

Functions:

CRUSH_XX
CRUSH_YY
CRUSH_ZZ
CRUSH_XY
CRUSH_YZ
CRUSH_ZX

Methods:

initialize(matParams * p);
getStress(matParams * p);

2.20 Orthotropic Rate Model

The **ORTHOTROPIC_RATE** model is an extension of the **ORTHOTROPIC_CRUSH** model that allows for rate dependent crush strengths. This model also allows for arbitrary orientation of the axes of orthotropy with respect to the global Cartesian axes. The model assumes three orthogonal axes of orthotropy – the T, L and W directions.

Normal and shear moduli with respect to the axes of orthotropy are input for the model along with the direction cosines for the T and L directions. The normal and shear crush strengths are defined through user input functions along with a rate multiplier function. Another function modifies the modulus of the material as a function of volumetric strain.

There are six state variables for this model, only one of which is aliased for output. There are eight user input functions for this model.

Properties:

LAMBDA	MODULUS_TLTL
SHEAR_MODULUS	MODULUS_LWLW
YIELD_STRESS	MODULUS_WTWT
MODULUS_TTTT	TX
MODULUS_TTLL	TY
MODULUS_TTWW	TZ
MODULUS_LLLL	LX
MODULUS_LLWW	LY
MODULUS_WWWW	LZ

State Variables (6):

0 - EVOL

Functions:

MODULUS_FUNCTION
RATE_FUNCTION
T_FUNCTION
L_FUNCTION
W_FUNCTION
TL_FUNCTION
LW_FUNCTION
WT_FUNCTION

Methods:

initialize(matParams * p);
getStress(matParams * p);

2.21 Power Law Creep Model

The **POWER_LAW_CREEP** model can be used to model the creep behavior of metals, like brazes and solders, and geologic materials like salt. The model is useful for capturing secondary creep in materials. The theory behind the **POWER_LAW_CREEP** model can be found in [6].

The power law creep model describes the evolution of the creep strain, ε_c, as a function of the von Mises stress, $\bar{\sigma}$

$$\dot{\varepsilon}_c = A\bar{\sigma}^m \exp(-B) \ , \tag{38}$$

where A is the creep constant, m is the creep exponent and B is the thermal constant.

There are two state variables for this model – the equivalent creep strain and the equivalent stress rate.

There are no user input functions for this model.

Properties:

YOUNGS_MODULUS
POISSONS_RATIO
CREEP_CONSTANT
CREEP_EXPONENT
THERMAL_CONSTANT

State Variables (2):

ECREEP
SEQDOT

Functions:

none

Methods:

getStress(matParams * p);

2.22 Soil and Foam Model

The **SOIL_FOAM** model is a model that was implemented in SANTOS [6] and JAS3D [16]. The model can be used as a simple model of a geologic material. The model has a pressure dependent yield surface which allows it to behave as, say, a Drucker-Prager model. The bulk behavior is modeled with a user input pressure-volumetric strain curve.

There are three state variables for this model. Only one of the state variables is aliased for output – volumetric strain.

There is one user input function. The pressure function gives the pressure as a function of volumetric strain.

Properties:

BULK_MODULUS
TWO_MU
PRESSURE_CUTOFF
A0
A1
A2

State Variables (3):

EVOL

Functions:

PRESSURE_FUNCTION

Methods:

initialize(matParams * p);
getStress(matParams * p);

2.23 Solder Model

The **SOLDER** model is a viscoplastic model that is used to model the mechanical response of solder. Detailed information on the model can be found in [17], [18], [19] and [20].

There are 28 state variables for this model. None of the state variables are aliased for output. State variables 12-28 are for temperature dependent material properties.

There are 17 user input functions for this model. These functions define the temperature dependence of various material properties.

Properties:

BULK_MODULUS	A3
SHEAR_MODULUS	A4
FLOW_STRESS	A5
FLOW_RATE	A6
SINH_EXPONENT	A7
GRAIN_SIZE	A8
GRAIN_EXPONENT	B1
ALPHA	B2
A1	B3
A2	

State Variables (28):

Functions:

BULK_FUNCTION
SHEAR_FUNCTION
RATE_FUNCTION
XP_FUNCTION
XM_FUNCTION
ALPHA_FUNCTION
A1_FUNCTION
A2_FUNCTION
A3_FUNCTION
A4_FUNCTION
A5_FUNCTION
A6_FUNCTION
A7_FUNCTION
A8_FUNCTION
B1_FUNCTION
B2_FUNCTION
B3_FUNCTION

Methods:

initialize(matParams * p);
getStress(matParams * p);
loadStepInit(matParams * p);

2.24 Solder with Damage Model

The **SOLDER_DAMAGE** model is a viscoplastic damage model that is used to model the mechanical response of solder. Microstructural damage is modeled through a damage evolution equation. Detailed information on the model can be found in [17], [18], [19] and [20].

There are 40 state variables for this model. None of the state variables are aliased for output. State variables 26-40 are for temperature dependent material properties.

There are 15 user input functions for this model. These functions define the temperature dependence of various material properties.

Properties:

BULK_MODULUS	A7
SHEAR_MODULUS	A8
FLOW_STRESS	B1
FLOW_RATE	B2
SINH_EXPONENT	B3
GRAIN_SIZE	D1
GRAIN_EXPONENT	D2
ALPHA	D3
A1	D4
A2	D5
A3	YIN
A4	YF
A5	GAMMA
A6	

State Variables (40):

Functions:

YOUNG_FUNCTION
POISSON_FUNCTION
RATE_FUNCTION
ALPHA_FUNCTION
A1_FUNCTION
A2_FUNCTION
A3_FUNCTION
A4_FUNCTION
A5_FUNCTION
A6_FUNCTION
A7_FUNCTION
B1_FUNCTION
B2_FUNCTION
YIN_FUNCTION
YF_FUNCTION

Methods:

initialize(matParams * p);
getStress(matParams * p);
loadStepInit(matParams * p);

2.25 Stiff Elastic Model

The **STIFF_ELASTIC** model is based on the **ELASTIC** model and is similar to the **INCOMPRESSIBLE_SOLID** model. However, rather than scale properties based on the notion of incompressibility (or the ratio of the shear modulus to the bulk modulus) this model scales properties based on relative the stiffness of different materials in the same problem – e.g. a "soft" and a "hard" material. This model requires some way to account for the scaled properties in the host code, e.g. the multi-level solution control and augmented-Lagrange wrappers in Adagio. All portions of the material response are softened by the same amount in this model. This model is not intended for transient dynamics applications, e.g. Presto. However, if it is used in Presto, the scaling is ignored.

There are no material state variables or user input functions for this model.

Properties:

YOUNGS_MODULUS
POISSONS_RATIO
SCALE_FACTOR

State Variables:

none

Functions:

none

Methods:

getStress(matParams * p);
pcElasticModuli(matParams * p);

2.26 Swanson

The Swanson model is a hyperelastic model typically used for modeling rubbers. By properly setting the material constants, a number of standard hyperelastic models can be recovered. The strain energy density is given by

$$U = \frac{3}{2}\frac{A_1}{P_1+1}\left(\frac{\bar{I}_1}{3}-1\right)^{P_1+1} + \frac{3}{2}\frac{B_1}{Q_1+1}\left(\frac{\bar{I}_2}{3}-1\right)^{Q_1+1} + \frac{3}{2}\frac{C_1}{R_1+1}\left(\frac{\bar{I}_1}{3}-1\right)^{R_1+1} + K(J_m \ln J_m - J_m)$$

where the strain quantities are defined as follows. The strain tensor is nominally the left Cauchy-Green strain tensor excluding the volumetric change given as

$$\left[\bar{B}\right] = \left[\bar{F}\right]\left[\bar{F}\right]^T \tag{39}$$

where $\left[\bar{F}\right]$ is defined in terms of the total deformation gradient $\left[F\right]$ as follows:

$$\left[\bar{F}\right] = J^{-\frac{1}{3}}\left[F\right] \tag{40}$$

The first and second strain invariants are simply

$$\bar{I}_1 = \left[I\right]:\left[\bar{B}\right] \tag{41}$$

and

$$\bar{I}_2 = \frac{1}{2}\left(\bar{I}_1^{\,2} - \left[I\right]:\left(\left[\bar{B}\right]\left[\bar{B}\right]\right)\right) \tag{42}$$

The mechanical volumetric strain is given by

$$J_m = \frac{J}{J_{th}} \tag{43}$$

where the total and thermal volumetric strains are given by

$$J = \det\left[F\right] \quad \text{and} \quad J_{th} = \det\left[F_{th}\right] \tag{44}$$

with $\left[F_{th}\right]$ defined as the isotropic thermal deformation gradient as follows:

$$\left[F_{th}\right] = J_{th}^{1/3}\left[I\right] \tag{45}$$

That is, the total deformation gradient is broken into mechanical and thermal parts using

$$[F] = [F_m][F_{th}] \tag{46}$$

The deviatoric stresses are given by

$$[\sigma'] = \frac{2}{J_m} DEV\left[\left(\frac{\partial U}{\partial \bar{I}_1} + \bar{I}_1 \frac{\partial U}{\partial \bar{I}_2}\right)[\bar{B}] - \frac{\partial U}{\partial \bar{I}_2}[\bar{B}][\bar{B}]\right] \tag{47}$$

where the partial derivatives of the strain energy are simply

$$\frac{\partial U}{\partial \bar{I}_1} = \frac{1}{2} A_1 \left(\frac{\bar{I}_1}{3} - 1\right)^{P_1} + \frac{1}{2} C_1 \left(\frac{\bar{I}_1}{3} - 1\right)^{R_1} \tag{48}$$

$$\frac{\partial U}{\partial \bar{I}_2} = \frac{1}{2} B_1 \left(\frac{\bar{I}_2}{3} - 1\right)^{Q_1} \tag{49}$$

The pressure is given by

$$p = K \ln(J_m) \tag{50}$$

As previously noted, a number of standard hyperelastic models can be recovered by properly setting the material constants. For instance, a Neo-Hookean model can be simulated by using

$$\begin{aligned} A_1 &= 2C_{10} & P_1 &= 0 \\ B_1 &= 0 & Q_1 &= 0 \\ C_1 &= 0 & R_1 &= 0 \\ K &= K \end{aligned} \tag{51}$$

The Mooney-Rivlin constitutive equation results from using

$$\begin{aligned} A_1 &= 2C_{10} & P_1 &= 0 \\ B_1 &= 2C_{01} & Q_1 &= 0 \\ C_1 &= 0 & R_1 &= 0 \\ K &= K \end{aligned} \tag{52}$$

However, unlike the native Mooney-Rivlin implementation in LAME, the Swanson model does not incorporate temperature dependent moduli.

Properties:

A1
P1
B1
Q1
C1
R1
BULK_MODULUS
CUT_OFF_STRAIN
TARGET_E
MAX_POISSONS_RATIO
BULK_SCALING
SHEAR_SCALING

State Variables (9):

SFJTH
JTH
VMECHXX
VMECHYY
VMECHZZ
VMECHXY
VMECHYZ
VMECHZX
SFJTH_FLAG

Functions:

TARGET_E_FUNCTION
THERMAL_EXPANSION_FUNCTION

Methods:

initialize(matParams * p);
getStress(matParams * p);
loadStepInit(matParams * p);
pcElasticModuli(matParams * p);

2.27 Thermoelastic Model

The **THERMOELASTIC** model is a temperature dependent version of the **ELASTIC** model where the elastic constants are temperature dependent. A description of the thermoelastic model can be found in [6].

There are no state variables for this model.

There are two user input functions: the Young's modulus function, the Poisson's ratio function. These functions define these properties as a function of temperature.

Properties:

YOUNGS_MODULUS
POISSONS_RATIO

State Variables:

none

Functions:

YOUNGS_MODULUS_FUNCTION
POISSONS_RATIO_FUNCTION

Methods:

initialize(matParams * p);
getStress(matParams * p);
loadStepInit(matParams * p);

2.28 Thermoelastic-Plastic Power Law Hardening Model

The **THERMO_EP_POWER** model is a thermoelastic-plastic power law hardening model. It is a temperature dependent version of the **EP_POWER_HARD** model that allows for the elastic properties and the yield stress to vary as a function of temperature. As an additional feature, this model allows for isotropic and/or kinematic hardening – a feature that is not in the **EP_POWER_HARD** model.

This model has 11 state variables, 8 that are available for output: the equivalent plastic strain, radius of the yield surface and six components of the back stress tensor that gives the location of the center of the yield surface. The other state variables keep track of the temperature dependent properties for the model.

There are three user input functions for this model: the Young's modulus function, the Poisson's ratio function and the yield stress function. These functions define these properties as a function of temperature.

Properties:

YOUNGS_MODULUS
POISSONS_RATIO
YIELD_STRESS
HARDENING_CONSTANT
HARDENING_EXPONENT
LUDERS_STRAIN
BETA

State Variables (11):

EQPS
RADIUS
ALPHA_XX
ALPHA_YY
ALPHA_ZZ
ALPHA_XY
ALPHA_YZ
ALPHA_ZX

Functions:

YOUNGS_MODULUS_FUNCTION
POISSONS_RATIO_FUNCTION
YIELD_STRESS_FUNCTION

Methods:

initialize(matParams * p);
getStress(matParams * p);
loadStepInit(matParams * p);

2.29 Thermoelastic-Plastic Power Law Hardening Weld Model

The **THERMO_EP_POWER_WELD** model is a variation of the **THERMO_EP_POWER** model that can be used to model, in a crude way, a weld. The added feature in this model is the transition temperature, T_t. The model has elastic properties that are a function of temperature. At the start of an analysis, the elastic properties have their values evaluated at the transition temperature. The first time that the temperature exceeds the transition temperature the material behaves as the **THERMO_EP_POWER** model.

This model has 12 state variables, 9 that are available for output: the equivalent plastic strain, radius of the yield surface, six components of the back stress tensor that gives the location of the center of the yield surface and a flag that turns on if the temperature has gone beyond the transition temperature. The other state variables keep track of the temperature dependent properties for the model.

There are three user input functions for this model: the Young's modulus function, the Poisson's ratio function and the yield stress function. These functions define these properties as a function of temperature.

Properties:

YOUNGS_MODULUS
POISSONS_RATIO
YIELD_STRESS
HARDENING_CONSTANT
HARDENING_EXPONENT
LUDERS_STRAIN
BETA
TRANSITION_TEMPERATURE

State Variables (12):

EQPS
RADIUS
ALPHA_XX
ALPHA_YY
ALPHA_ZZ
ALPHA_XY
ALPHA_YZ
ALPHA_ZX
WELD_FLAG

Functions:

YOUNGS_MODULUS_FUNCTION
POISSONS_RATIO_FUNCTION
YIELD_STRESS_FUNCTION

Methods:

initialize(matParams * p);
getStress(matParams * p);
loadStepInit(matParams * p);

2.30 Universal Polymer Model

This is a phenomenological, nonlinear viscoelastic material model for analyzing stresses and strains in filled and unfilled polymers. It represents a simplification of the **NLVE_POLYMER** model, and it was developed for production analyses of encapsulated components. It predicts a full range of behavior including yielding, stress relaxation, volume relaxation, and physical aging. The model uses a material clock driven by temperature, volume and strain (similar to potential internal energy of **NLVE_POLYMER**). The strain measure is obtained from the integration of the rate of deformation tensor. As a special feature, it does allow the user to initiate an analysis from a stress-free temperature, T_{sf}, that is different from the reference temperature, T_{ref}, at which the material properties are defined.

The Cauchy stress is given by

$$\begin{aligned}
\underline{\underline{\sigma}} = & \left\{\mathrm{K_g}(T(t)) - K_\infty(T(t))\right\} \int_0^t ds\ \mathrm{f_1}\left(t^* - s^*\right) \frac{d\mathrm{I_1}}{ds}(s)\ \underline{\underline{\mathrm{I}}} \\
& - \left\{\mathrm{K_g}(T(t))\delta_g(T(t)) - K_\infty(T(t))\delta_\infty(T(t))\right\} \int_0^t ds\ \mathrm{f_1}\left(t^* - s^*\right) \frac{dT}{ds}(s)\ \underline{\underline{\mathrm{I}}} \\
& + 2\left\{G_\mathrm{g}(T(t)) - G_\infty(T(t))\right\} \int_0^t ds\ \mathrm{f_2}\left(t^* - s^*\right)\ \frac{d\underline{\underline{\varepsilon}}_{dev}}{ds}(s) \\
& + \left\{K_\infty(T(t))\, I_1 - K_\infty(T(t))\, \delta_\infty(T(t))\left[\mathrm{T(t) - T_{sf}}\right]\right\}\ \underline{\underline{\mathrm{I}}}\ + 2\, G_\infty(T(t))\ \underline{\underline{\varepsilon}}_{dev}
\end{aligned} \tag{53}$$

where

$$\begin{aligned}
I_1 &= \underline{\underline{I}} : \underline{\underline{\varepsilon}} = trace(\underline{\underline{\varepsilon}}) \\
\underline{\underline{\varepsilon}}_{dev} &= \underline{\underline{\varepsilon}} - \frac{1}{3} I_1\ \underline{\underline{I}}
\end{aligned} \tag{54}$$

and

$$K_g(T) = K_{g\,ref} + \frac{dK_g}{dT}(T - T_{ref}) = K_{g\,ref} + \frac{dK_g}{dT}(T_{sf} - T_{ref}) + \frac{dK_g}{dT}(T - T_{sf}) = K_{g\,sf} + \frac{dK_g}{dT}(T - T_{sf})$$

$$K_\infty(T) = K_{\infty ref} + \frac{dK_\infty}{dT}(T - T_{ref}) = K_{\infty ref} + \frac{dK_\infty}{dT}(T_{sf} - T_{ref}) + \frac{dK_\infty}{dT}(T - T_{sf}) = K_{\infty sf} + \frac{dK_\infty}{dT}(T - T_{sf})$$

$$\alpha_g(T) = \alpha_{g\,ref} + \frac{d\alpha_g}{dT}(T - T_{ref}) = \alpha_{g\,ref} + \frac{d\alpha_g}{dT}(T_{sf} - T_{ref}) + \frac{d\alpha_g}{dT}(T - T_{sf}) = \alpha_{g\,sf} + \frac{d\alpha_g}{dT}(T - T_{sf})$$

$$\delta_g(T) = \left[\alpha_{g\,sf} + \frac{1}{2}\frac{d\alpha_g}{dT}(T - T_{sf}) \right]$$

$$\alpha_\infty(T) = \alpha_{\infty ref} + \frac{d\alpha_\infty}{dT}(T - T_{ref}) = \alpha_{\infty ref} + \frac{d\alpha_\infty}{dT}(T_{sf} - T_{ref}) + \frac{d\alpha_\infty}{dT}(T - T_{sf}) = \alpha_{\infty sf} + \frac{d\alpha_\infty}{dT}(T - T_{sf})$$

$$\delta_\infty(T) = \left[\alpha_{\infty sf} + \frac{1}{2}\frac{d\alpha_\infty}{dT}(T - T_{sf}) \right]$$

(55)

$$G_g(T) = G_{g\,ref} + \frac{dG_g}{dT}(T - T_{ref}) = G_{g\,ref} + \frac{dG_g}{dT}(T_{sf} - T_{ref}) + \frac{dG_g}{dT}(T - T_{sf}) = G_{g\,sf} + \frac{dG_g}{dT}(T - T_{sf})$$

$$G_\infty(T) = G_{\infty ref} + \frac{dG_\infty}{dT}(T - T_{ref}) = G_{\infty ref} + \frac{dG_\infty}{dT}(T_{sf} - T_{ref}) + \frac{dG_\infty}{dT}(T - T_{sf}) = G_{\infty sf} + \frac{dG_\infty}{dT}(T - T_{sf})$$

(56)

The material clock is defined by

$$t^* - s^* = \int_s^t \frac{dw}{a(w)} \qquad \textit{and} \qquad \log\ \mathrm{a} = -\hat{C}_1 \left(\frac{N}{\hat{C}_2 + N} \right) \tag{57}$$

where

$$N = \left\{ \left[T(t) - T_{ref} \right] - \int_0^t ds\ \mathrm{f}_1\left(t^* - s^*\right) \frac{dT}{ds}(s) \right\} + C_3 \left\{ I_1(t)_{ref} - \int_0^t ds\ \mathrm{f}_1\left(t^* - s^*\right) \frac{d\mathrm{I}_1}{ds}(s) \right\} + C_4 \left\{ \int_0^t \int_0^t ds\, du\, f(t^* - s^*, t^* - u^*) \frac{d\underline{\underline{\varepsilon}}_{dev}(s)}{ds} : \frac{d\underline{\underline{\varepsilon}}_{dev}(u)}{du} \right\} \tag{58}$$

$$N = \left\{ \left[T(t) - T_{ref} \right] - \int_0^t ds\ \mathrm{f}_1\left(t^* - s^*\right) \frac{dT}{ds}(s) \right\} + C_3 \left\{ \left[I_1(t) - I_{1_{sf}} \right] + \left[I_{1_{sf}} - I_{1_{ref}} \right] - \int_0^t ds\ \mathrm{f}_1\left(t^* - s^*\right) \frac{d\mathrm{I}_1}{ds}(s) \right\} + C_4 \left\{ I_2(t) \right\}_{dev}^{eff}$$

(59)

Note that the clock invariant $I_1(t)_{ref}$ is measured from the reference temperature, not the stress free state. In order for the clock to reduce to the WLF equation at temperatures above T_g, we must define the clock coefficients consistently. To do so, first approximate N for and equilibrated material slightly above T_g:

$$\begin{aligned} N &= (T - T_{ref}) + C_3 \left\{ \left[I_1(t) - I_{1_{sf}} \right] + \left[I_{1_{sf}} - I_{1_{ref}} \right] \right\} \approx (T - T_{ref}) + C_3 \left[\alpha_\infty (T - T_{ref}) \right] \\ &= (1 + C_3 \alpha_\infty)(T - T_{ref}) \end{aligned} \tag{60}$$

To make the UPM clock reproduce the WLF equation above T_g, we must enforce a condition that

$$\begin{aligned} &(\log a)_{WLF} = (\log a)_{UPM} \\ &\frac{-C_1 (T - T_{ref})}{C_2 + (T - T_{ref})} = \frac{-\hat{C}_1 (1 + C_3 \alpha_\infty)(T - T_{ref})}{\hat{C}_2 + (1 + C_3 \alpha_\infty)(T - T_{ref})} = \frac{-\hat{C}_1 (T - T_{ref})}{\dfrac{\hat{C}_2}{(1 + C_3 \alpha_\infty)} + (T - T_{ref})} \end{aligned} \tag{61}$$

From this condition, we can define the UPM clock parameters in terms of the WLF coefficients.

$$\begin{aligned} \hat{C}_1 &= C_1 \\ \hat{C}_2 &= C_2 \left(1 + C_3 \alpha_{\infty ref} \right) \end{aligned} \tag{62}$$

Note that if the volumetric coefficient of thermal expansion varies as a linear function of temperature, then the volume strain is quadratic:

$$\begin{aligned} I_1(T) &= \int_{T_0}^{T} \alpha(s)\, ds = \int_{T_0}^{T} \left[\alpha_0 + \frac{d\alpha}{dT}(s - T_0) \right] ds \\ &= \left[\left(\alpha_0 - \frac{d\alpha}{dT} T_0 \right) s \right]_{T_0}^{T} + \left[\frac{1}{2} \frac{d\alpha}{dT} s^2 \right]_{T_0}^{T} = \left(\alpha_0 - \frac{d\alpha}{dT} T_0 \right)(T - T_0) + \frac{1}{2} \frac{d\alpha}{dT} (T^2 - T_0^{\,2}) \\ &= \alpha_0 (T - T_0) + \frac{1}{2} \frac{d\alpha}{dT} (T - T_0)^2 \\ &= \left[\alpha_0 + \frac{1}{2} \frac{d\alpha}{dT} (T - T_0) \right] (T - T_0) \end{aligned} \tag{63}$$

where we adopt the notation that

$$I_1(T) = \delta(T) \left[T - T_o \right] \qquad \textit{and} \qquad \delta(T) \equiv \alpha_0 + \frac{1}{2} \frac{d\alpha}{dT} (T - T_0) \tag{64}$$

Properties:

There are 134 properties for this model. Some are input, but some are derived from other inputs. For example, the relaxation functions are usually input as Williams-Watts functions and then Prony series are determined from them.

WWBETA 1
WWTAU 1
WWBETA 2
WWTAU 2
SPECTRUM START TIME
SPECTRUM END TIME
LOG TIME INCREMENT
BULK GLASSY 0
BULK GLASSY 1
BULK GLASSY 2
BULK RUBBERY 0
BULK RUBBERY 1
BULK RUBBERY 2
VOLCTE GLASSY 0
VOLCTE GLASSY 1
VOLCTE GLASSY 2
VOLCTE RUBBERY 0
VOLCTE RUBBERY 1
VOLCTE RUBBERY 2
SHEAR GLASSY 0
SHEAR GLASSY 1
SHEAR GLASSY 2
SHEAR RUBBERY 0
SHEAR RUBBERY 1
SHEAR RUBBERY 2
REFERENCE TEMPERATURE
WLF C1
WLF C2
CLOCK C1
CLOCK C2
CLOCK C3
CLOCK C4
CLOCK C5
CLOCK C6
FILLER VOL FRACTION
STRESS FREE TEMPERATURE
RELAX TIME 1
RELAX TIME 2
RELAX TIME 3
RELAX TIME 4
RELAX TIME 5
RELAX TIME 6
RELAX TIME 7
RELAX TIME 8
RELAX TIME 9
RELAX TIME 10
RELAX TIME 11
RELAX TIME 12
RELAX TIME 13
RELAX TIME 14
RELAX TIME 15
RELAX TIME 16
RELAX TIME 17
RELAX TIME 18
RELAX TIME 19
RELAX TIME 20
RELAX TIME 21
RELAX TIME 22
RELAX TIME 23
RELAX TIME 24
RELAX TIME 25
RELAX TIME 26
RELAX TIME 27
RELAX TIME 28
RELAX TIME 29
RELAX TIME 30
F1 1
F1 2
F1 3
F1 4
F1 5
F1 6
F1 7
F1 8
F1 9
F1 10
F1 11
F1 12

F1 13
F1 14
F1 15
F1 16
F1 17
F1 18
F1 19
F1 20
F1 21
F1 22
F1 23
F1 24
F1 25
F1 26
F1 27
F1 28
F1 29
F1 30
F2 1
F2 2
F2 3
F2 4
F2 5
F2 6
F2 7
F2 8
F2 9
F2 10
F2 11
F2 12
F2 13
F2 14
F2 15
F2 16
F2 17
F2 18
F2 19
F2 20
F2 21
F2 22
F2 23
F2 24
F2 25
F2 26
F2 27
F2 28
F2 29
F2 30

State Variables (249):

AEND
IGXX1
IGYY1
IGXY1
IGYZ1
IGZX1
IGXX2
IGYY2
IGXY2
IGYZ2
IGZX2
IGXX3
IGYY3
IGXY3
IGYZ3
IGZX3
IGXX4
IGYY4
IGXY4
IGYZ4
IGZX4
IGXX5
IGYY5
IGXY5
IGYZ5
IGZX5
IGXX6
IGYY6
IGXY6
IGYZ6
IGZX6
IGXX7
IGYY7
IGXY7

IGYZ7
IGZX7
IGXX8
IGYY8
IGXY8
IGYZ8
IGZX8
IGXX9
IGYY9
IGXY9
IGYZ9
IGZX9
IGXX10
IGYY10
IGXY10
IGYZ10
IGZX10
IGXX11
IGYY11
IGXY11
IGYZ11
IGZX11
IGXX12
IGYY12
IGXY12
IGYZ12
IGZX12
IGXX13
IGYY13
IGXY13
IGYZ13
IGZX13
IGXX14
IGYY14
IGXY14
IGYZ14
IGZX14
IGXX15
IGYY15
IGXY15
IGYZ15
IGZX15
IGXX16
IGYY16
IGXY16
IGYZ16
IGZX16
IGXX17
IGYY17
IGXY17
IGYZ17
IGZX17
IGXX18
IGYY18
IGXY18
IGYZ18
IGZX18
IGXX19
IGYY19
IGXY19
IGYZ19
IGZX19
IGXX20
IGYY20
IGXY20
IGYZ20
IGZX20
IGXX21
IGYY21
IGXY21
IGYZ21
IGZX21
IGXX22
IGYY22
IGXY22
IGYZ22
IGZX22
IGXX23
IGYY23
IGXY23
IGYZ23
IGZX23
IGXX24
IGYY24
IGXY24
IGYZ24
IGZX24
IGXX25
IGYY25
IGXY25

IGYZ25
IGZX25
IGXX26
IGYY26
IGXY26
IGYZ26
IGZX26
IGXX27
IGYY27
IGXY27
IGYZ27
IGZX27
IGXX28
IGYY28
IGXY28
IGYZ28
IGZX28
IGXX29
IGYY29
IGXY29
IGYZ29
IGZX29
IGXX30
IGYY30
IGXY30
IGYZ30
IGZX30
IKI11
IKI12
IKI13
IKI14
IKI15
IKI16
IKI17
IKI18
IKI19
IKI110
IKI111
IKI112
IKI113
IKI114
IKI115
IKI116
IKI117
IKI118
IKI119
IKI120
IKI121
IKI122
IKI123
IKI124
IKI125
IKI126
IKI127
IKI128
IKI129
IKI130
IKAT1
IKAT2
IKAT3
IKAT4
IKAT5
IKAT6
IKAT7
IKAT8
IKAT9
IKAT10
IKAT11
IKAT12
IKAT13
IKAT14
IKAT15
IKAT16
IKAT17
IKAT18
IKAT19
IKAT20
IKAT21
IKAT22
IKAT23
IKAT24
IKAT25
IKAT26
IKAT27
IKAT28
IKAT29
IKAT30
IF1P1
IF1P2
IF1P3

IF1P4
IF1P5
IF1P6
IF1P7
IF1P8
IF1P9
IF1P10
IF1P11
IF1P12
IF1P13
IF1P14
IF1P15
IF1P16
IF1P17
IF1P18
IF1P19
IF1P20
IF1P21
IF1P22
IF1P23
IF1P24
IF1P25
IF1P26
IF1P27
IF1P28
IF1P29
IF1P30
EPSXX
EPSYY
EPSZZ
EPSXY
EPSYZ
EPSZX
LOGA
EFFI2

Functions:

none

Methods:

initialize(matParams * p);
getStress(matParams * p);

2.31 Viscoelastic Swanson Model

The viscoelastic Swanson is a finite strain viscoelastic model which has an initial elastic response that matches the Swanson material model [21, 22, 23]. The model is typically employed in calculating the response of rubber materials. The bulk response is elastic, while the deviatoric response is viscoelastic. Such a constitutive modeling approach is commonly used in simulating the response of rubbers.

The Cauchy stress $[\sigma(t)]$ is computed from the following equation

$$[\sigma(t)] = p(t)[I] + \left[\sigma_0^{dev}(t)\right] + DEV\left\{\frac{1}{J}[F(t)]\left\{\int_0^t \frac{1}{G_0}\frac{dG(t^*-\tau^*)}{d(t-\tau)}\left[S_0^{dev}(\tau)\right]d\tau\right\}[F(t)]^T\right\} \tag{65}$$

which is rewritten as follows:

$$[\sigma(t)] = p(t)[I] + \left[\sigma_0^{dev}(t)\right] + DEV\left\{\frac{1}{J}[F(t)]\left\{\int_0^{t^*} \frac{1}{G_0}\frac{dG(\tau^*)}{d\tau^*}\left[S_0^{dev}(t-\tau)\right]d\tau^*\right\}[F(t)]^T\right\} \tag{66}$$

where $[F(t)]$ is the total deformation gradient at time t, J is the determinant of $[F(t)]$, $p(t)$ is the elastic pressure computed as

$$p = K\ln(J_m) \tag{67}$$

J_m is the determinant of the mechanical part of $[F(t)]$, $\left[\sigma_0^{dev}(t)\right]$ is the deviatoric Cauchy stress at time t computed using the elastic Swanson model with initial moduli values, $\left[S_0^{dev}(t-\tau)\right]$ is the deviatoric second Piola-Kirchhoff stress computed using the elastic Swanson model with initial moduli values at time $t-\tau$, and *G(t)* is the shear relaxation modulus. Here the reference state for the two state tensors ($[F(t)]$ and $\left[S_0^{dev}(t-\tau)\right]$) is the original configuration at $t=0$. The shear relaxation modulus is represented using a Prony series as follows:

$$G(t) = G_0\left(g_\infty + \sum_{i=1}^{N_G} g_i \exp\left[-\frac{t}{\lambda_i}\right]\right) \tag{68}$$

Finally, the reduced material time t^* is related to the physical time t as follows:

$$dt^* = \frac{1}{A_{WLF} A_N} dt \tag{69}$$

or

$$t^* = \int_0^t \frac{dx}{A_{WLF}(x) A_N(x)} \tag{70}$$

where A_{WLF} is the WLF shift factor given by

$$\log_{10} A_{WLF} = \frac{-C_1 \left(T - T_{ref} \right)}{C_2 + \left(T - T_{ref} \right)} \tag{71}$$

and A_N is a numerical shift factor which the model user can specify arbitrarily to slow or speed the viscoelastic relaxations as desired.

Properties:

A1	PRONY_SHEAR_6
P1	PRONY_SHEAR_7
B1	PRONY_SHEAR_8
Q1	PRONY_SHEAR_9
C1	PRONY_SHEAR_10
R1	SHEAR_RELAX_TIME_1
BULK_MODULUS	SHEAR_RELAX_TIME_2
CUT_OFF_STRAIN	SHEAR_RELAX_TIME_3
TARGET_E	SHEAR_RELAX_TIME_4
MAX_POISSONS_RATIO	SHEAR_RELAX_TIME_5
BULK_SCALING	SHEAR_RELAX_TIME_6
SHEAR_SCALING	SHEAR_RELAX_TIME_7
PRONY_SHEAR_INFINITY	SHEAR_RELAX_TIME_8
PRONY_SHEAR_1	SHEAR_RELAX_TIME_9
PRONY_SHEAR_2	SHEAR_RELAX_TIME_10
PRONY_SHEAR_3	WLF_COEF_C1
PRONY_SHEAR_4	WLF_COEF_C2
PRONY_SHEAR_5	WLF_TREF

State Variables (77):

SFJTH
JTH
VMECHXX
VMECHYY
VMECHZZ
VMECHXY
VMECHYZ
VMECHZX
SFJTH_FLAG
VSXXDEV1
VSYYDEV1
VSZZDEV1
VSXYDEV1
VSYZDEV1
VSZXDEV1
VSXXDEV2
VSYYDEV2
VSZZDEV2
VSXYDEV2
VSYZDEV2
VSZXDEV2
VSXXDEV3
VSYYDEV3
VSZZDEV3
VSXYDEV3
VSYZDEV3
VSZXDEV3
VSXXDEV4
VSYYDEV4
VSZZDEV4
VSXYDEV4
VSYZDEV4
VSZXDEV4
VSXXDEV5
VSYYDEV5
VSZZDEV5
VSXYDEV5
VSYZDEV5
VSZXDEV5
VSXXDEV6
VSYYDEV6
VSZZDEV6
VSXYDEV6
VSYZDEV6
VSZXDEV6
VSXXDEV7
VSYYDEV7
VSZZDEV7
VSXYDEV7
VSYZDEV7
VSZXDEV7
VSXXDEV8
VSYYDEV8
VSZZDEV8
VSXYDEV8
VSYZDEV8
VSZXDEV8
VSXXDEV9
VSYYDEV9
VSZZDEV9
VSXYDEV9
VSYZDEV9
VSZXDEV9
VSXXDEV10
VSYYDEV10
VSZZDEV10
VSXYDEV10
VSYZDEV10
VSZXDEV10
SOXXDEV
SOYYDEV
SOZZDEV
SOXYDEV
SOYZDEV
SOZXDEV
WLF_AAVG
ANAVG

Functions:

NUMERICAL_SHIFT_FUNCTION
THERMAL_EXPANSION_FUNCTION

Methods:

initialize(matParams * p);
getStress(matParams * p);
loadStepInit(matParams * p);
pcElasticModuli(matParams * p);

2.32 Viscoplastic Model

The **VISCOPLASTIC** model is used to model braze joint materials with state variables to model hardening and recovery. A detailed presentation of the theory can be found in [24].

There are 19 state variables for this model. The state variables describe the back stress, strain rate, and iteration counts. State variables 11-19 are used for temperature dependent material properties.

There are 9 user input functions for this model. These functions define the temperature dependence of various material properties.

Properties:

SHEAR_MODULUS
BULK_MODULUS
ISO_EXPONENT
KIN_EXPONENT
FLOW_STRESS
FLOW_RATE
SINH_EXPONENT
ALPHA
ISO_HARDENING
ISO_RECOVERY
KIN_HARDENING
KIN_RECOVERY

State Variables (19):

0 - SVBXX
1 - SVBYY
2 - SVBZZ
3 - SVBXY
4 - SVBYZ
5 - SVBZX
8 - EQDOT
9 – COUNT
10 - SHEAR
11 - BULK
12 - RATE
13 - EXP
14 - ALPHA
15 – A1
16 – A2
17 – A4
18 – A5

Functions:

SHEAR_FUNCTION
BULK_FUNCTION
RATE_FUNCTION
EXPONENT_FUNCTION
ALPHA_FUNCTION
IHARD_FUNCTION
IREC_FUNCTION
KHARD_FUNCTION
KREC_FUNCTION YOUNGS_MODULUS_FUNCTION

Methods:

initialize(matParams * p);
getStress(matParams * p);
loadStepInit(matParams * p);

2.33 Viscoplastic Foam Model

The **VISCOPLASTIC_FOAM** model is a plasticity model that is used to model rigid foams. It used a non-associated flow rule. The model is documented in [25].

There are 14 state variables for this model. None of the state variables are aliased for output.

There are eight user input functions for this model that described temperature dependent properties, rate effects and hardening behavior.

Properties:

YOUNGS_MODULUS
POISSONS_RATIO
FLOW_RATE
POWER_EXPONENT
BETA
SHEAR_STRENGTH
SHEAR_HARDENING
HYDRO_STRENGTH
HYDRO_HARDENING
SHEAR_EXPONENT
HYDRO_EXPONENT
PHI

State Variables (14):

Functions:

YOUNGS_FUNCTION
POISSONS_FUNCTION
RATE_FUNCTION
EXPONENT_FUNCTION
SS_FUNCTION
SH_FUNCTION
HS_FUNCTION
HH_FUNCTION

Methods:

```
initialize( matParams * p );
getStress( matParams * p );
loadStepInit( matParams * p );
```

3. SUMMARY

A range of material models have been implemented in LAME. These include simple linear elastic models with and without temperature dependent moduli. Another set of models incorporate different forms of plasticity with linear or power law hardening with yield surfaces that correspond to isotropic, kinematic or mixed hardening. Several model options are available in LAME for modeling foams. Likewise, several hyperelastic models for the large strain elastic response of rubbers are available. Finally, several viscoelastic models appropriate for potting epoxies or rubbers exist in LAME. Currently, all of the LAME models are available in the ASC codes Adagio and Presto. Several of the LAME models have been hooked-up to the augmented-Lagrange wrappers in Adagio such that the conjugate gradient solver in Adagio can be employed efficiently. Such wrappers allow part or all of the material response to be stiffened or softened. Such scaling of material model behavior is intended for use in Adagio. If these particular models are employed in Presto, the specified scalings are ignored.

A brief overview of each model in LAME has been presented in this report. The required material parameters, the state variables available for named output, and the name of any user input functions are given. Finally, for informational purposes for both developers and analysts, the particular LAME routines that are actually implemented for each model are listed. All models employ a **`getStress()`** method for determining the stresses used for equilibrium calculations. Many use an **`initialize()`** method for initializing state variables. Some models incorporate a **`loadStepInit()`** method for initializing pieces of information once per time step. For instance, temperature dependent moduli only need to be set once per step when the temperature is known for all time steps. Some models need to change the moduli that are used for the elastic preconditioner in Adagio at each time step. For instance, in the case of temperature-dependent moduli, it is better to use the moduli at the current temperature in the elastic preconditioner. Another example of when the elastic moduli for the preconditioner may need to be changed is for materials which soften dramatically at large strains. Such elastic preconditioner moduli changes are performed using the **`pcElasticModuli()`** method. Materials that do not employ a **`pcElasticModuli()`** method have appropriate moduli for the elastic preconditioner set once at the beginning of the Adagio analysis.

4. REFERENCES

1. Koteras, J. R., A. S. Gullerud, N. K. Crane, and J. D. Hales, *Presto User's Guide Version 2.6*, SAND2006-6093, Sandia National Laboratories, Albuquerque, NM, October 2006.
2. Pierson, K. H. and J. D. Hales, "ADAGIO/ANDANTE User's Guide Version 2.0", Memo, Sandia National Laboratories, Albuquerque, NM, September 2004.
3. Scherzinger, W.M. and D. C. Hammerand, *Library of Advanced Materials for Engineering - LAME*, SAND07-5515, Sandia National Laboratories, Albuquerque, NM, October 2007.
4. Scherzinger, W. M. and D. C. Hammerand, *Testing of Constitutive Models in LAME*, SAND07-5872, Sandia National Laboratories, Albuquerque, NM, October 2007.
5. Bammann, D. J., M. L. Chiesa and G. C. Johnson, "Modeling Large Deformation and Failure in Manufacturing Processes," *Proceedings of the 19th International Congress of Theoretical and Applied Mechanics*, ed. T. Tatsumi, E. Watanabe and T. Kambe, pp. 359-376, Elsevier Science Publishers, Amsterdam, 1997
6. Stone, C. M., *SANTOS – A Two-Dimensional Finite Element Program for the Quasistatic, Large Deformation, Inelastic Response of Solids*, SAND90-0543, Sandia National Laboratories, Albuquerque, NM, July 1997.
7. Stone, C. M., G. W. Wellman and R. D. Krieg, *A Vectorized Elastic/Plastic Power Law Hardening Material Model Including Lüders Strain*, SAND90-0153, Sandia National Laboratories, Albuquerque, NM, March 1990.
8. Neilsen, M. K. and W. Y. Lu, "Failure Predicted by a Plasticity Model for Polyurethane Foam," 6th U. S. Congress on Computational Mechanics, Dearborn, MI, August 1-3, 2001.
9. Hinnerichs, T. D., T. G. Carne, W. Y. Lu, E. C. Stasiunas, M. K. Neilsen, W. M. Scherzinger and B. R. Rogillio, *Characterization of Aluminum Honeycomb and Experimentation for Model Development and Validation, Volume II: Honeycomb Experimentation for Model Development and Validation*, SAND2006-4455, Sandia National Laboratories, Albuquerque, NM, August 2006.
10. Ogden, R. W., *Non-Linear Elastic Deformations*, Dover, Mineola, NY, 1997.
11. HKS, *ABAQUS Version 6.6, User's Manual, Vol. III, Materials*, Hibbitt, Karlsson and Sorensen, Providence, RI, 2006.
12. Neilsen, M.K., H. S. Morgan, and R. D. Krieg, *A Phenomenological Constitutive Model for Low Density Polyurethane Foams*, SAND86-2927, Sandia National Laboratories, Albuquerque, NM, April 1987.
13. Simo, J. C. and T. J. R. Hughes, *Computational Inelasticity*, Springer, New York, 1998.
14. Caruthers, J. M., D. B. Adolf, R. S. Chambers, and P. Shrikhande, "A Thermodynamically Consistent, Nonlinear Viscoelastic Approach for Modeling Glassy Polymers", *Polymer*, **45**, pp. 4577-4597, 2004.
15. Adolf, D. B., R. S. Chambers and J. M. Caruthers, "Extensive Validation of a Thermodynamically Consistent, Nonlinear Viscoelastic Model for Glassy Polymers", *Polymer*, **45**, pp. 4599-4621, 2004.
16. Blanford, M.L., Heinstein, M.W., Key, S.W., 'JAS3D – A Multi-Strategy Iterative Code for Solid Mechanics Analysis Users' Instructions, Release 2.0', Memo, Sandia National Laboratories, Albuquerque, NM, September 2001.

17. Fossum, A.F., P.T. Vianco, M. K. Neilsen and D. M. Pierce, "A Practical Viscoplastic Damage Model for Lead-Free Solder," *Journal of Electronic Packaging*, **128**, pp. 71-81, 2006.
18. Vianco, P. T., J. A. Rejent, A. F. Fossum and M. K. Neilsen, "Compression Stress-Strain and Creep Properties of the 52 In – 48 Sn and 97 In – 3 Ag Low-Temperature Pb-Free Solders," *Journal of Materials Science: Materials in Electronics*, **18** (1-3), pp. 93-119, 2006.
19. Wei, Y., C. L. Chow, M. K. Neilsen and H. E. Fang, "Characteristics of Creep Damage for 60 Sn – 40 Pb Solder Material," *Journal of Electronic Packaging*, **123**, pp. 278-283, 2001.
20. Wei, Y., C. L. Chow, M. K. Neilsen and H. E. Fang, "Failure Analysis of Solder Joints with a Damage-Coupled Viscoplastic Model," *International Journal for Numerical Methods in Engineering*, **56**, pp. 2199-2211, 2003.
21. HKS, *ABAQUS Version 6.6, Theory Manual*, Hibbitt, Karlsson and Sorensen, Providence, RI, 2006.
22. Hammerand, D. C., 'ABAQUS Style Finite Strain Viscoelasticity in Adagio', Memo, Sandia National Laboratories, Albuquerque, NM, March 2003.
23. Hammerand, D. C., 'Finite Strain Viscoelasticity in Adagio and ABAQUS', Memo, Sandia National Laboratories, Albuquerque, NM, July 2003.
24. Neilsen, M. K., S. N. Burchett, C. M. Stone and J. J. Stephens, *A Viscoplastic Theory for Braze Alloys*, SAND96-0984, Sandia National Laboratories, Albuquerque, NM, April 1996.
25. Neilsen, M. K., W. Y. Lu, B. Olsson and T. D. Hinnerichs, "A Viscoplastic Constitutive Model for Polyurethane Foams," *Proceedings of the IMECE2006*, November 5-10, 2006, Chicago, IL.

Distribution

1	MS0346	Tom Baca	1523
1	MS0346	Robert Chambers	1524
1	MS0346	Jim Cox	1526
1	MS0346	Stephen Montgomery	1524
1	MS0346	Matthew Neidigk	1524
1	MS0346	Dave Reedy	1526
1	MS0346	Mike Starr	1526
1	MS0372	Lupe Arguello	1525
1	MS0372	Joe Bishop	1525
1	MS0372	Nicole Breivik	1524
1	MS0372	Frank Dempsey	1524
1	MS0372	Kristin Dion	1524
1	MS0372	Brenton Elisberg	1524
1	MS0372	Jeff Gruda	1524
1	MS0372	Kenneth Gwinn	1524
3	MS0372	Daniel Hammerand	1524
1	MS0372	Terry Hinnerichs	1524
1	MS0372	John Holland	1524
1	MS0372	David Lo	1524
1	MS0372	Don Longcope	1524
1	MS0372	Kurt Metzinger	1524
1	MS0372	Jake Ostien	1524
1	MS0372	John Pott	1524
1	MS0372	Jim Redmond	1525
3	MS0372	William Scherzinger	1524
1	MS0372	Mike Stone	1525
1	MS0372	Leah Tuttle	1524
1	MS0372	Gerald Wellman	1525
1	MS0376	James Bean	1524
1	MS0376	Jonathon Rath	1524
1	MS0380	Manoj Bhardwaj	1542
1	MS0380	Nathan Crane	1542
1	MS0380	Arne Gullerud	1542
1	MS0380	Jason Hales	1542
1	MS0380	Martin Heinstein	1542
1	MS0380	Joe Jung	1542
1	MS0380	Richard Koteras	1542
1	MS0380	Hal Morgan	1540
1	MS0380	Kendall Pierson	1542
1	MS0380	Vicki Porter	1542
1	MS0380	Garth Reese	1542

1	MS0380	Ben Spencer	1542	
1	MS0380	Tim Walsh	1542	
1	MS0382	Carter Edwards	1543	
1	MS0382	Mike Glass	1541	
1	MS0384	Art Ratzel	1500	
1	MS0555	Rod May	1522	
1	MS0557	Dave Clauss	1521	
1	MS0557	Dan Segalman	1525	
1	MS0660	Chris Lamb	9512	
1	MS0836	Shane Schumacher	1516	
1	MS0847	Marcus Billings	1524	
1	MS0847	Pavel Chaplya	1526	
1	MS0847	Mike Neilsen	1526	
1	MS0847	Pete Wilson	1520	
1	MS1070	Channy Wong	1526	
1	MS1070	Jordan Massad	1526	
1	MS1070	Phillip Reu	1526	
1	MS1070	Anton Sumali	1526	
1	MS9018	Central Technical Files	8944	(electronic)
1	MS0899	Technical Library	9536	(electronic)


Sandia National Laboratories