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.

Approved for public release; further dissemination unlimited.
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 ................................................................................................................................ 7

2. Constitutive Models .................................................................................................................. 9
   2.1 BCJ Model .......................................................................................................................... 9
   2.2 BCJ MEM Model .............................................................................................................. 10
   2.3 Ductile Fracture Model ..................................................................................................... 12
   2.4 Elastic Model .................................................................................................................... 14
   2.5 Elastic-Plastic Model ........................................................................................................ 15
   2.6 Elastic-Plastic Power Law Hardening Model ................................................................... 17
   2.7 Elastic Fracture Model ...................................................................................................... 18
   2.8 Foam Plasticity Model ....................................................................................................... 19
   2.9 Honeycomb Model ........................................................................................................... 20
   2.10 Hyperfoam Model .......................................................................................................... 21
   2.11 Incompressible Solid Model ........................................................................................... 23
   2.12 Johnson-Cook Model ...................................................................................................... 24
   2.13 Low Density Foam Model ............................................................................................. 26
   2.14 Mooney-Rivlin Model .................................................................................................... 27
   2.15 Multilinear Elastic-Plastic Model ................................................................................... 30
   2.16 Multilinear Elastic Plastic with Failure Model ............................................................... 32
   2.17 Neo-Hookean Model ....................................................................................................... 34
   2.18 NLVE Polymer Model .................................................................................................... 35
   2.19 Orthotropic Crush Model .............................................................................................. 41
   2.20 Orthotropic Rate Model ............................................................................................... 42
   2.21 Power Law Creep Model ............................................................................................... 44
   2.22 Soil and Foam Model ..................................................................................................... 45
   2.23 Solder Model .................................................................................................................... 46
   2.24 Solder with Damage Model ............................................................................................ 48
   2.25 Stiff Elastic Model .......................................................................................................... 50
   2.26 Swanson .......................................................................................................................... 51
   2.27 Thermoelastic Model ...................................................................................................... 54
   2.28 Thermoelastic-Plastic Power Law Hardening Model ..................................................... 55
   2.29 Thermoelastic-Plastic Power Law Hardening Weld Model .......................................... 57
   2.30 Universal Polymer Model ............................................................................................. 59
   2.31 Viscoelastic Swanson Model .......................................................................................... 67
   2.32 Viscoplastic Model ......................................................................................................... 71
   2.33 Viscoplastic Foam Model ............................................................................................... 73

3. Summary .................................................................................................................................. 75

4. References ................................................................................................................................ 77
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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YOUNGS_MODULUS</td>
<td>C17</td>
</tr>
<tr>
<td>POISSONS_RATIO</td>
<td>C18</td>
</tr>
<tr>
<td>C1</td>
<td>C19</td>
</tr>
<tr>
<td>C2</td>
<td>C20</td>
</tr>
<tr>
<td>C3</td>
<td>DAMAGE_EXPOENT</td>
</tr>
<tr>
<td>C4</td>
<td>THETA_OPT</td>
</tr>
<tr>
<td>C5</td>
<td>FACTOR</td>
</tr>
<tr>
<td>C6</td>
<td>RHO</td>
</tr>
<tr>
<td>C7</td>
<td>SPECIFIC_HEAT</td>
</tr>
<tr>
<td>C8</td>
<td>INITIAL_ALPHA_XX</td>
</tr>
<tr>
<td>C9</td>
<td>INITIAL_ALPHA_YY</td>
</tr>
<tr>
<td>C10</td>
<td>INITIAL_ALPHA_ZZ</td>
</tr>
<tr>
<td>C11</td>
<td>INITIAL_ALPHA_XY</td>
</tr>
<tr>
<td>C12</td>
<td>INITIAL_ALPHA_YZ</td>
</tr>
<tr>
<td>C13</td>
<td>INITIAL_ALPHA_ZX</td>
</tr>
<tr>
<td>C14</td>
<td>INITIAL_KAPPA</td>
</tr>
<tr>
<td>C15</td>
<td>INITIAL_DAMAGE</td>
</tr>
<tr>
<td>C16</td>
<td>TEMP0</td>
</tr>
</tbody>
</table>

State Variables (14):

Functions:

YOUNGS_MODULUS_FUNCTION
POISSONS_RATIO_FUNCTION

Methods:

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

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

\[
\bar{\sigma} = \sigma_y + A \left( \varepsilon_p - \varepsilon_L \right)^n ,
\]

where \( \bar{\sigma} \) is the von Mises stress, \( \sigma_y \) is the yield stress, \( \varepsilon_p \) is the equivalent plastic strain, \( \varepsilon_L \) is the Lüders strain, \( A \) is the hardening constant, \( n \) is the hardening exponent, and \( \langle \cdot \rangle \) 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

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

where \( \sigma_{\max} \) is the maximum principal stress, \( \sigma_m \) is the mean stress and \( \langle \cdot \rangle \) 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

\[ \dot{\sigma}_{ij} = \lambda \dot{\varepsilon}_{ij} D_{kk} + 2\mu \dot{D}_{ij}, \]

where \( \lambda \) and \( \mu \) are the Lamé constants, \( D_{ij} \) are the components of the rate of deformation and the stress rate, \( \dot{\sigma}_{ij} \), 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.

\[ \lambda = \frac{E \nu}{(1-2\nu)(1+\nu)} \]
\[ \mu = \frac{E}{2(1+\nu)} \]

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

\[ \bar{\sigma} = \sigma_y + H' \bar{\varepsilon}_p, \]  

(5)

where \( \bar{\sigma} \) is the von Mises stress, \( \sigma_y \) 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

\[ \sigma = \sigma_y + A \left( \bar{\varepsilon}_p - \varepsilon_L \right)^n, \]

where \( \sigma \) is the von Mises stress, \( \sigma_y \) is the yield stress, \( \bar{\varepsilon}_p \) is the equivalent plastic strain, \( \varepsilon_L \) is the Lüders strain, \( A \) is the hardening constant, \( n \) is the hardening exponent, and \( \left\langle \bullet \right\rangle \) 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

\[ \phi = \frac{\sigma_m^2}{a^2} + \frac{\sigma_s^2}{b^2} - 1, \]  

(7)

where \( \sigma_m \) is the mean stress, \( \sigma_s \) 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
SHEAR_MODULUS
YIELD_STRESS
MODULUS_TTTT
MODULUS_TLLL
MODULUS_TTWW
MODULUS_LLLL
MODULUS_LLWW
MODULUS_WWWW
A1
A2
A3
B1
B2
B3

State Variables (39):

Functions:

MODULUS_FUNCTION
RATE_FUNCTION
T_FUNCTION
L_FUNCTION
W_FUNCTION
TL_FUNCTION
LW_FUNCTION
WT_FUNCTION

Methods:

initialize( matParams * p );
gesture( 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

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

where the \( \lambda_i \) are the principal stretch ratios and \( J \) is the Jacobian of the deformation. The material parameters are \( \mu_k, \alpha_k \) and \( \nu_k \) with

\[
\beta_k = \frac{\nu_k}{1 - 2\nu_k}. \quad (9)
\]

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

\[
\sigma_i = \frac{\lambda_i}{J} \frac{\partial W}{\partial \lambda_i} \quad \text{(no sum on } l) \quad . \quad (10)
\]

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:

```c
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:

\[
\sigma = \left(\sigma_y + AE_p^n\right)\left(1 + C \ln \dot{\varepsilon}^*\right)(1 - T^m),
\]

where \(\sigma\) is the von Mises stress, \(E_p\) 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

\[
\dot{\varepsilon}^* = \frac{\dot{\varepsilon}^p}{\dot{\varepsilon}_0},
\]

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

\[
T^* = \frac{T - T_0}{T_m - T_0},
\]

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
    EQDOTT

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

$$\sigma = A \langle I_2' \rangle + B (1 + CI_1),$$  \hspace{1cm} (14)

where \( A, B \) and \( C \) are material properties, \( \langle \cdot \rangle \) 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

\[
U = C_{10} (I_1 - 3) + C_{01} (I_2 - 3) + K (J_m \ln J_m - J_m)
\]  

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

\[
\begin{bmatrix}
\bar{B}
\end{bmatrix} = \begin{bmatrix}
\bar{F}
\end{bmatrix}^T
\]  

where \(\begin{bmatrix}
\bar{F}
\end{bmatrix}\) is defined in terms of the total deformation gradient \(\begin{bmatrix}
F
\end{bmatrix}\) as follows:

\[
\begin{bmatrix}
\bar{F}
\end{bmatrix} = J^{-\frac{1}{3}} \begin{bmatrix}
F
\end{bmatrix}
\]  

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

\[
J = \text{det}\begin{bmatrix}
F
\end{bmatrix} = \frac{dV(t)}{dV(0)}
\]  

The first and second strain invariants are simply

\[
I_1 = [I] : \begin{bmatrix}
\bar{B}
\end{bmatrix}
\]  

and

\[
I_2 = \frac{1}{2} \left( I_1^2 - [I] : \left( \begin{bmatrix}
\bar{B}
\end{bmatrix} \begin{bmatrix}
\bar{B}
\end{bmatrix} \right) \right)
\]  

The mechanical volumetric strain is given by

\[
J_m = \frac{J}{J_{th}}
\]  

where the thermal volumetric strain is given by

\[
J_{th} = \text{det}\begin{bmatrix}
F_{th}
\end{bmatrix}
\]
with \([F_{th}]\) defined as the thermal deformation gradient. That is, the total deformation gradient is multiplicatively decomposed as

\[
[F] = [F_m][F_{th}]
\]

(23)

with isotropic thermal expansion defined by

\[
[F_{th}] = J^{1/3} [I]
\]

(24)

The deviatoric and volumetric stresses are respectively given as

\[
[\sigma'] = \frac{2}{J_m} DEV \left[ (C_{10} + I_1 C_{01}) [\bar{B}] - C_{01} [\bar{B}] [\bar{B}] \right]
\]

(25)

and

\[
p = K \ln (J_m)
\]

(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, \( H(\bar{\varepsilon}_p) \), 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

\[
\bar{\sigma} = \sigma_y + H(\bar{\varepsilon}_p) ,
\]  

(27)

where \( \bar{\sigma} \) is the von Mises stress and \( \sigma_y \) 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:

- YOUNG'S_MODULUS
- POISSON'S_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, \( H(\varepsilon_p) \), 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

\[
\bar{\sigma} = \sigma_y + H(\varepsilon_p),
\]

(28)

where \( \bar{\sigma} \) is the von Mises stress and \( \sigma_y \) 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

\[
t_p = \int_0^{\varepsilon_f} \left( \frac{2\sigma_{\text{max}}}{3(\sigma_{\text{max}} - \sigma_m)} \right)^4 d\varepsilon_p,
\]

(29)

where \( \sigma_{\text{max}} \) is the maximum principal stress, \( \sigma_m \) 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

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

where \( K \) is the bulk modulus, \( \mu \) is the shear modulus and \( C \) is the right Cauchy-Green tensor.

The stress can be derived directly from the strain energy density. The components of the Cauchy stress, \( \sigma_{ij} \), are

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

where \( B_{ij} \) are the components of the left Cauchy-Green tensor and \( J = \det F \) 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 );`
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 $H$) 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, $\Psi$. The internal energy $U$ in the model is related to the Helmholtz free energy as

$$U = \Psi + T \eta$$

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

$$U = U_{\infty} + \frac{1}{2} \Psi_1 \int_0^T \int_0^{t^*} ds \ du \ f_1(t^* - s^*, t^* - u^*) \frac{dI_H}{ds}(u) \frac{dI_H}{du}(u)$$

$$+ \Psi_2(T) \int_0^t \int_0^{t^*} ds \ du \ f_2(t^* - s^*, t^* - u^*) \frac{dH}{ds} \frac{dH}{du}$$

$$+ \Psi_3(T, I_H) \int_0^T \int_0^{t^*} ds \ du \ f_3(t^* - s^*, t^* - u^*) \frac{dT}{ds} \frac{dT}{du}$$

$$+ \frac{1}{2} \Psi_4(T) \int_0^t \int_0^{t^*} ds \ du \ f_4(t^* - s^*, 0) \frac{dT}{ds}(s) + \Psi_5(T, I_H) \int_0^t \int_0^{t^*} ds \ du \ f_5(t^* - s^*, 0) \frac{dI_H}{ds}(s)$$

$$- T \left[ \Psi_3(T) \int_0^T ds \ f_3(t^* - s^*, 0) \frac{dT}{dt}(s) + \Psi_4(T, I_H) \int_0^t \int_0^{t^*} ds \ du \ f_4(t^* - s^*, 0) \frac{dT}{ds}(s) \frac{dT}{du}(u) \right]$$

$$+ \left( \frac{\partial \Psi_3}{\partial T} \right)_{I_H} \int_0^t \int_0^{t^*} ds \ du \ f_3(t^* - s^*, t^* - u^*) \frac{dT}{ds}(s) \frac{dT}{du}(u)$$

$$+ \frac{1}{2} \left( \frac{\partial \Psi_4}{\partial T} \right)_{I_H} \int_0^t \int_0^{t^*} ds \ du \ f_4(t^* - s^*, t^* - u^*) \frac{dT}{ds}(s) \frac{dT}{du}(u)$$

$$+ \left( \frac{\partial \Psi_5}{\partial T} \right)_{I_H} \int_0^t \int_0^{t^*} ds \ du \ f_5(t^* - s^*, t^* - u^*) \frac{dH}{ds} \frac{dH}{du}$$

with the material clock defined by
\[ t^* - s^* = \int \frac{dw}{s \alpha(w)} \quad \text{and} \quad \log a = B \left( \frac{1}{U_c} - \frac{1}{U_c^{ref}} \right) = C_1 \left( \frac{U_c^{ref}}{U_c} - 1 \right) \]  

(34)

and the equilibrium energy given by

\[ U_{eq} (I_H, \Delta T) = U_{eq}^{ref} + \frac{1}{2} \psi_1 I_H^2 + \psi_{\Pi H} \Pi_H - \psi_3^{ref} I_H \Delta T - \frac{1}{2} \psi_4^{ref} \Delta T^2 - \left( \frac{\partial \psi_3^{ref}}{\partial I_H} \right) I_H^2 \Delta T 
- \left( \frac{\partial \psi_4^{ref}}{\partial T} \right) I_H \Delta T^2 - \frac{1}{2} \left( \frac{\partial \psi_4^{ref}}{\partial T} \right) \Delta T^3 - \frac{1}{4} \left( \frac{\partial^2 \psi_4^{ref}}{\partial T^2} \right) \Delta T^4 \]  

(35)

\[ + T \left[ \psi_3^{ref} \Delta T + 2 \left( \frac{\partial \psi_3^{ref}}{\partial T} \right) I_H \Delta T + \frac{3}{2} \left( \frac{\partial \psi_4^{ref}}{\partial T} \right) \Delta T^2 \right] 
+ \left( \frac{\partial^3 \psi_4^{ref}}{\partial T^2} \right) \Delta T^3 + \psi_3^{ref} I_H + \left( \frac{\partial \psi_3^{ref}}{\partial I_c} \right) I_H^2 \]  

The Hencky stress \( S_H \) is the stress measure that is work conjugate with the Hencky strain \( \bar{H} \) and is given by

\[ S_H = \rho_{ref} \int_0^t ds f_1 (t^* - s^*) \frac{dI_H}{ds} (s) - 2 \rho_{ref} \int_0^t ds f_2 (t^* - s^*) \frac{dH}{ds}(s) \]

\[ + \rho_{ref} \int_0^t ds f_3 (t^* - s^*) \frac{dT}{ds} (s) \]

\[ + \rho_{ref} \int_0^t ds f_4 (t^* - s^*) \frac{dI_H}{ds} (s) \]

\[ + \rho_{ref} \int_0^t ds f_5 (t^* - s^*) \frac{dH}{ds}(s) \]

\[ + \rho_{ref} \left[ \psi_1 I_H^2 + \psi_{\Pi H} \Pi_H + \psi_3 I_H \Delta T + \psi_4 \Delta T^2 \right] \]

\[ I^2 + 2 \rho_{ref} \psi_{\Pi H} \bar{H} \]

(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:

\[ f_m (t) = \sum_{i=1}^{N} f_{mi} e^{-t/\tau_i} \]  

(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 \((f_1)\) that are used to define the normalized relaxation function \(f_1\) associated with the \(Ψ_1\) integral in the equation for \(\underline{S}_H\)

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 \(\underline{S}_H\)

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 \(\underline{S}_H\)

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 \(\underline{S}_H\)

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

\[
Ψ_1(I_H) = Ψ_{1\text{ref}}^r + \frac{dΨ_1}{dI_H} I_H
\]

1PSI REF, \(Ψ_{1\text{ref}}^r\)

I1 DERIV 1PSI, \(\frac{dΨ_1}{dI_H}\)

\[
Ψ_2(T,I_H,I_{HH}) = Ψ_{2\text{ref}}^r + \frac{dΨ_2}{dT}(T - T_{\text{REF}}) + \frac{dΨ_1}{dI_H} I_H + \frac{dΨ_1}{dI_{HH}} I_{HH} + 1 \frac{d^2Ψ_1}{dI_{HH}^2} I_{HH}^2
\]

2PSI REF, \(Ψ_{2\text{ref}}^r\)

T DERIV 2PSI, \(\frac{dΨ_2}{dT}\)

I1 DERIV 2PSI, \(\frac{dΨ_2}{dI_H}\)
\[ I_2 \text{ DERIV 2PSI, } \frac{d\Psi}{dI_{HH}}, \quad (I_{HH} \text{ is the 2}^{nd} \text{ Hencky strain invariant}) \]

\[ I_2 \text{ 2DERIV 2PSI, } \frac{d^2\Psi}{dI_{HH}^2} \]

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

\[ 3\text{PSI REF, } \Psi_3^{\text{ref}} \]

\[ T \text{ DERIV 3PSI, } \frac{d\Psi_3}{dT} \]

\[ I_1 \text{ DERIV 3PSI, } \frac{d\Psi_3}{dI_H} \]

\[ I_1T \text{ 2DERIV 3PSI, } \frac{d^2\Psi_3}{dI_H dT} \]

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

\[ 4\text{PSI REF, } \Psi_4^{\text{ref}} \]

\[ T \text{ DERIV 4PSI, } \frac{d\Psi_4}{dT} \]

\[ T \text{ 2DERIV 4PSI, } \frac{d^2\Psi_4}{dT^2} \]

\[ I_1 \text{ DERIV 4PSI, } \frac{d\Psi_4}{dI_H} \]

Reference temperature, \( T_{\text{ref}} \)
Reference density
WLF coefficient \( C_1 \)
WLF coefficient \( C_2 \)
\( B \), Shift factor constant
\( U_c^{\text{ref}} \), Shift factor constant

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

WWBETA 1PSI, \( \beta_1 \)
WWTAU 1PSI, \( \tau_1 \)

\[ f_2(t) = \exp \left[ -\left( \frac{t}{\tau_2} \right)^\beta_2 \right] \]
WWBETA 2PSI, $\beta_2$
WWTAU 2PSI, $\tau_2$

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

WWBETA 3PSI, $\beta_3$
WWTAU 3PSI, $\tau_3$

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

WWBETA 4PSI, $\beta_4$
WWTAU 4PSI, $\tau_4$

DOUBLE INTEG FACTOR (not used)
RUBBERY BULK MOD
$I_1$ DERIV $R\_BULK$
GLASSY BULK MOD
$I_1$ DERIV $G\_BULK$
RUBBERY SHEAR MOD
$T$ DERIV $R\_SHEAR$
$I_2$ 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, $\Psi_{II}$, (These terms are used in the Taylor series
PSI EQ IT, $\Psi_{IT}$ expansion that defines the equilibrium
PSI EQ 2T, $\Psi_{TT}$ stress)
PSI EQ 2H, $\Psi_{IIH}$
PSI EQ 3I, $\Psi_{III}$
PSI EQ 2IT, $\Psi_{IIT}$
PSI EQ 12T, $\Psi_{IITT}$
PSI EQ 3T, $\Psi_{TTT}$
PSI EQ 2HT, $\Psi_{IIIT}$
PSI EQ 4I, \( \Psi_{III} \)
PSI EQ 3IT, \( \Psi_{IIT} \)
PSI EQ 2I2T, \( \Psi_{IIT} \)
PSI EQ 13T, \( \Psi_{III} \)
PSI EQ 4T, \( \Psi_{III} \)
PSI EQ 4H, \( \Psi_{IIIH} \)
PSI POT IT
PSI POT 2T
PSI POT 2IT
PSI POT 12T
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 fairly 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:

<table>
<thead>
<tr>
<th>LAMBDA</th>
<th>MODULUS_TTLT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHEAR_MODULUS</td>
<td>MODULUS_LWLW</td>
</tr>
<tr>
<td>YIELD_STRESS</td>
<td>MODULUS_WTWT</td>
</tr>
<tr>
<td>MODULUS_TTTT</td>
<td>TX</td>
</tr>
<tr>
<td>MODULUS_TLLL</td>
<td>TY</td>
</tr>
<tr>
<td>MODULUS_TTWW</td>
<td>TZ</td>
</tr>
<tr>
<td>MODULUS_LLLL</td>
<td>LX</td>
</tr>
<tr>
<td>MODULUS_LLWW</td>
<td>LY</td>
</tr>
<tr>
<td>MODULUS_WWWW</td>
<td>LZ</td>
</tr>
</tbody>
</table>

State Variables (6):

0 - EVOL

Functions:

<table>
<thead>
<tr>
<th>MODULUS_FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>RATE_FUNCTION</td>
</tr>
<tr>
<td>T_FUNCTION</td>
</tr>
<tr>
<td>L_FUNCTION</td>
</tr>
<tr>
<td>W_FUNCTION</td>
</tr>
<tr>
<td>TL_FUNCTION</td>
</tr>
<tr>
<td>LW_FUNCTION</td>
</tr>
<tr>
<td>WT_FUNCTION</td>
</tr>
</tbody>
</table>
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, \( \varepsilon_c \), 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:

```plaintext
BULK_MODULUS
TWO_MU
PRESSURE_CUTOFF
A0
A1
A2
```

State Variables (3):

```plaintext
E VOL
```

Functions:

```plaintext
PRESSURE_FUNCTION
```

Methods:

```plaintext
initialize( matParams * p );
g etStress( 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
- SHEAR_MODULUS
- FLOW_STRESS
- FLOW_RATE
- SINH_EXPONENT
- GRAIN_SIZE
- GRAIN_EXPONENT
- ALPHA
- A1
- A2
- A3
- A4
- A5
- A6
- A7
- A8
- B1
- B2
- B3

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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>BULK_MODULUS</td>
<td>A7</td>
</tr>
<tr>
<td>SHEAR_MODULUS</td>
<td>A8</td>
</tr>
<tr>
<td>FLOW_STRESS</td>
<td>B1</td>
</tr>
<tr>
<td>FLOW_RATE</td>
<td>B2</td>
</tr>
<tr>
<td>SINH_EXPONENT</td>
<td>B3</td>
</tr>
<tr>
<td>GRAIN_SIZE</td>
<td>D1</td>
</tr>
<tr>
<td>GRAIN_EXPONENT</td>
<td>D2</td>
</tr>
<tr>
<td>ALPHA</td>
<td>D3</td>
</tr>
<tr>
<td>A1</td>
<td>D4</td>
</tr>
<tr>
<td>A2</td>
<td>D5</td>
</tr>
<tr>
<td>A3</td>
<td>YIN</td>
</tr>
<tr>
<td>A4</td>
<td>YF</td>
</tr>
<tr>
<td>A5</td>
<td>GAMMA</td>
</tr>
<tr>
<td>A6</td>
<td></td>
</tr>
</tbody>
</table>

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

\[
U = \frac{3}{2} A_1 \left( \frac{\tilde{I}_1}{3} - 1 \right)^{n_1+1} + \frac{3}{2} B_1 \left( \frac{\tilde{I}_2}{3} - 1 \right)^{n_2+1} + \frac{3}{2} C_1 \left( \frac{\tilde{I}_3}{3} - 1 \right)^{n_3+1} + K (J_m \ln J_m - J_m)
\]

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

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

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

\[
\left[ \bar{F} \right] = J^{1/3} \left[ F \right]
\]

The first and second strain invariants are simply

\[
\tilde{I}_1 = \left[ I \right] : \left[ \bar{B} \right]
\]

and

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

The mechanical volumetric strain is given by

\[
J_m = \frac{J}{J_{th}}
\]

where the total and thermal volumetric strains are given by

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

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

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

That is, the total deformation gradient is broken into mechanical and thermal parts using
The deviatoric stresses are given by
\[
\begin{bmatrix} \sigma' \end{bmatrix} = \frac{2}{J_m} \text{DEV} \begin{bmatrix} \frac{\partial U}{\partial I_1} + \frac{1}{3} \frac{\partial U}{\partial I_2} \end{bmatrix} \begin{bmatrix} B \end{bmatrix} - \begin{bmatrix} \frac{\partial U}{\partial I_2} \end{bmatrix} \begin{bmatrix} B \end{bmatrix} \begin{bmatrix} B \end{bmatrix}
\]  \tag{47}

where the partial derivatives of the strain energy are simply
\[
\frac{\partial U}{\partial I_1} = \frac{1}{2} A_i \left( \frac{I_1}{3} - 1 \right)^{r_i} + \frac{1}{2} C_i \left( \frac{I_1}{3} - 1 \right)^{r_i}
\]  \tag{48}
\[
\frac{\partial U}{\partial I_2} = \frac{1}{2} B_i \left( \frac{I_2}{3} - 1 \right)^{r_i}
\]  \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
\[
A_i = 2C_{10} \quad P_i = 0
\]
\[
B_i = 0 \quad Q_i = 0
\]
\[
C_i = 0 \quad R_i = 0
\]
\[
K = K
\]  \tag{51}

The Mooney-Rivlin constitutive equation results from using
\[
A_i = 2C_{10} \quad P_i = 0
\]
\[
B_i = 2C_{01} \quad Q_i = 0
\]
\[
C_i = 0 \quad R_i = 0
\]
\[
K = K
\]  \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 \texttt{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 \texttt{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, \( T_{\text{sf}} \), that is different from the reference temperature, \( T_{\text{ref}} \), at which the material properties are defined.

The Cauchy stress is given by

\[
\sigma = \left\{ K_g(T(t)) - K_\infty(T(t)) \right\} \int_0^1 ds f_1(t^* - s*) \frac{dI_1}{ds}(s) \right| + \\
- \left\{ K_\delta(T(t)) \delta_g(T(t)) - K_\infty(T(t)) \delta_\infty(T(t)) \right\} \int_0^1 ds f_1(t^* - s*) \frac{dT}{ds}(s) \right| \\
+ 2 \left\{ G_g(T(t)) - G_\infty(T(t)) \right\} \int_0^1 ds f_2(t^* - s*) \frac{d\varepsilon_{\text{dev}}}{ds}(s) \\
+ \left\{ K_\infty(T(t)) I_1 - K_\infty(T(t)) \delta_\infty(T(t)) [T(t) - T_{\text{sf}}] \right\} \right| + 2 G_\infty(T(t)) \varepsilon_{\text{dev}} \\
\tag{53}
\]

where

\[
I_1 = I : \varepsilon = \text{trace} (\varepsilon) \\
\varepsilon_{\text{dev}} = \varepsilon - \frac{1}{3} I_1 I \\
\tag{54}
\]

and
\[
K_g(T) = K_{g_{\text{ref}}} + \frac{dK_g}{dT}(T - T_{\text{ref}}) = K_{g_{\text{ref}}} + \frac{dK_g}{dT}(T_{sf} - T_{\text{ref}}) + \frac{dK_g}{dT}(T - T_{sf}) = K_{g_{sf}} + \frac{dK_g}{dT}(T - T_{sf})
\]

\[
K_{m}(T) = K_{m_{\text{ref}}} + \frac{dK_m}{dT}(T - T_{\text{ref}}) = K_{m_{\text{ref}}} + \frac{dK_m}{dT}(T_{sf} - T_{\text{ref}}) + \frac{dK_m}{dT}(T - T_{sf}) = K_{m_{\text{sf}}} + \frac{dK_m}{dT}(T - T_{sf})
\]

\[
\alpha_{g}(T) = \alpha_{g_{\text{ref}}} + \frac{d\alpha_{g}}{dT}(T - T_{\text{ref}}) = \alpha_{g_{\text{ref}}} + \frac{d\alpha_{g}}{dT}(T_{sf} - T_{\text{ref}}) + \frac{d\alpha_{g}}{dT}(T - T_{sf}) = \alpha_{g_{\text{sf}}} + \frac{d\alpha_{g}}{dT}(T - T_{sf})
\]

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

\[
\alpha_{m}(T) = \alpha_{m_{\text{ref}}} + \frac{d\alpha_{m}}{dT}(T - T_{\text{ref}}) = \alpha_{m_{\text{ref}}} + \frac{d\alpha_{m}}{dT}(T_{sf} - T_{\text{ref}}) + \frac{d\alpha_{m}}{dT}(T - T_{sf}) = \alpha_{m_{\text{sf}}} + \frac{d\alpha_{m}}{dT}(T - T_{sf})
\]

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

\[(55)\]

\[
G_g(T) = G_{g_{\text{ref}}} + \frac{dG_g}{dT}(T - T_{\text{ref}}) = G_{g_{\text{ref}}} + \frac{dG_g}{dT}(T_{sf} - T_{\text{ref}}) + \frac{dG_g}{dT}(T - T_{sf}) = G_{g_{\text{sf}}} + \frac{dG_g}{dT}(T - T_{sf})
\]

\[
G_{m}(T) = G_{m_{\text{ref}}} + \frac{dG_m}{dT}(T - T_{\text{ref}}) = G_{m_{\text{ref}}} + \frac{dG_m}{dT}(T_{sf} - T_{\text{ref}}) + \frac{dG_m}{dT}(T - T_{sf}) = G_{m_{\text{sf}}} + \frac{dG_m}{dT}(T - T_{sf})
\]

\[(56)\]

The material clock is defined by

\[
t^* - s^* = \int \frac{dw}{s \ a(w)} \quad \text{and} \quad \log \ a = - \hat{C}_1 \left( \frac{N}{\hat{C}_2 + N} \right)
\]

\[(57)\]

where

\[
N = \left\{ \left[ T(t) - T_{\text{ref}} \right] - \int_0^t ds \ f_i(t^* - s^*) dt/s \right\} + C_3 \left\{ \left[ I_1(t)_{\text{ref}} - \int_0^t ds \ f_i(t^* - s^*) \frac{dI_1}{ds} (s) \right] + C_4 \left\{ \left[ \int_0^t ds \ f_i(t^* - s^*) \frac{d\epsilon_{\text{dev}}}{ds} (s) - \frac{d\epsilon_{\text{dev}}}{du} (u) \right] \right\}
\]

\[(58)\]

\[
N = \left\{ \left[ T(t) - T_{\text{ref}} \right] - \int_0^t ds \ f_i(t^* - s^*) dt/s \right\} + C_3 \left\{ \left[ I_1(t) - I_{1_{\text{ref}}} \right] + \left[ I_{1_{\text{ref}}} - I_{1_{\text{ref}}} \right] - \int_0^t ds \ f_i(t^* - s^*) \frac{dI_1}{ds} (s) \right\}
\]

\[(59)\]
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 $T_g$:

$$N = (T - T_{ref}) + C_3 \left[ \left( I_1(t) - I_{1_{ref}} \right) + \left( I_{1_{ref}} - I_{1_{ref}} \right) \right] \approx (T - T_{ref}) + C_3 \left[ \alpha_{\infty} (T - T_{ref}) \right]$$

(60)

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

$$\frac{-C_1(T - T_{ref})}{C_2 + (T - T_{ref})} = \frac{-\hat{C}_1(1 + C_3\alpha_{\infty})(T - T_{ref})}{\hat{C}_2 + (1 + C_3\alpha_{\infty})(T - T_{ref})} = \frac{-\hat{C}_1(T - T_{ref})}{\hat{C}_2 + (1 + C_3\alpha_{\infty})(T - T_{ref})}$$

(61)

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

$$\hat{C}_1 = C_1$$

$$\hat{C}_2 = C_2 \left( 1 + C_3\alpha_{\infty} \right)$$

(62)

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

$$I_1(T) = \int_{T_0}^{T} \alpha(s) \, ds = \int_{T_0}^{T} \alpha_0 + \frac{d\alpha}{dT} (s - T_0) \, ds$$

$$= \left[ \alpha_0 - \frac{d\alpha}{dT} T_0 \right] s_{T_0}^{T} + \left[ \frac{1}{2} \frac{d\alpha}{dT} s^2 \right]_{T_0}^{T} = \left( \alpha_0 - \frac{d\alpha}{dT} T_0 \right) (T - T_0) + \frac{1}{2} \frac{d\alpha}{dT} (T^2 - T_0^2)$$

(63)

where we adopt the notation that

$$I_1(T) = \delta(T) [T - T_0] \quad \text{and} \quad \delta(T) \equiv \alpha_0 + \frac{1}{2} \frac{d\alpha}{dT} (T - T_0)$$

(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.

<p>| WWBETA 1 | RELAX TIME 4 |
| WWTAU 1  | RELAX TIME 5 |
| WWBETA 2 | RELAX TIME 6 |
| WWTAU 2  | RELAX TIME 7 |
| SPECTRUM START TIME | RELAX TIME 8 |
| SPECTRUM END TIME   | RELAX TIME 9 |
| LOG TIME INCREMENT  | RELAX TIME 10|
| BULK GLASSY 0       | RELAX TIME 11|
| BULK GLASSY 1       | RELAX TIME 12|
| BULK GLASSY 2       | RELAX TIME 13|
| BULK RUBBERY 0      | RELAX TIME 14|
| BULK RUBBERY 1      | RELAX TIME 15|
| BULK RUBBERY 2      | RELAX TIME 16|
| VOLCTE GLASSY 0     | RELAX TIME 17|
| VOLCTE GLASSY 1     | RELAX TIME 18|
| VOLCTE GLASSY 2     | RELAX TIME 19|
| VOLCTE RUBBERY 0    | RELAX TIME 20|
| VOLCTE RUBBERY 1    | RELAX TIME 21|
| VOLCTE RUBBERY 2    | RELAX TIME 22|
| SHEAR GLASSY 0      | RELAX TIME 23|
| SHEAR GLASSY 1      | RELAX TIME 24|
| SHEAR GLASSY 2      | RELAX TIME 25|
| SHEAR RUBBERY 0     | RELAX TIME 26|
| SHEAR RUBBERY 1     | RELAX TIME 27|
| SHEAR RUBBERY 2     | RELAX TIME 28|
| REFERENCE TEMPERATURE | RELAX TIME 29|
| WLF C1              | RELAX TIME 30|
| WLF C2              | F1 1 |
| CLOCK C1            | F1 2 |
| CLOCK C2            | F1 3 |
| CLOCK C3            | F1 4 |
| CLOCK C4            | F1 5 |
| CLOCK C5            | F1 6 |
| CLOCK C6            | F1 7 |
| FILLER VOL FRACTION | F1 8 |
| STRESS FREE TEMPERATURE | F1 9 |
| RELAX TIME 1        | F1 10|
| RELAX TIME 2        | F1 11|
| RELAX TIME 3        | F1 12|</p>
<table>
<thead>
<tr>
<th>F1</th>
<th>F2</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>7</td>
</tr>
<tr>
<td>14</td>
<td>8</td>
</tr>
<tr>
<td>15</td>
<td>9</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
</tr>
<tr>
<td>26</td>
<td>20</td>
</tr>
<tr>
<td>27</td>
<td>21</td>
</tr>
<tr>
<td>28</td>
<td>22</td>
</tr>
<tr>
<td>29</td>
<td>23</td>
</tr>
<tr>
<td>30</td>
<td>24</td>
</tr>
</tbody>
</table>

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  IGYZ16
IGZX7  IGZX16
IGXX8  IGXX17
IGYY8  IGYY17
IGXY8  IGXY17
IGYZ8  IGYZ17
IGZX8  IGZX17
IGXX9  IGXX18
IGYY9  IGYY18
IGXY9  IGXY18
IGYZ9  IGYZ18
IGZX9  IGZX18
IGXX10  IGXX19
IGYY10  IGYY19
IGXY10  IGXY19
IGYZ10  IGYZ19
IGZX10  IGZX19
IGXX11  IGXX20
IGYY11  IGYY20
IGXY11  IGXY20
IGYZ11  IGYZ20
IGZX11  IGZX20
IGXX12  IGXX21
IGYY12  IGYY21
IGXY12  IGXY21
IGYZ12  IGYZ21
IGZX12  IGZX21
IGXX13  IGXX22
IGYY13  IGYY22
IGXY13  IGXY22
IGYZ13  IGYZ22
IGZX13  IGZX22
IGXX14  IGXX23
IGYY14  IGYY23
IGXY14  IGXY23
IGYZ14  IGYZ23
IGZX14  IGZX23
IGXX15  IGXX24
IGYY15  IGYY24
IGXY15  IGXY24
IGYZ15  IGYZ24
IGZX15  IGZX24
IGXX16  IGXX25
IGYY16  IGYY25
IGXY16  IGXY25
<table>
<thead>
<tr>
<th>IGYZ25</th>
<th>IKI119</th>
</tr>
</thead>
<tbody>
<tr>
<td>IGZX25</td>
<td>IKI120</td>
</tr>
<tr>
<td>IGXX26</td>
<td>IKI121</td>
</tr>
<tr>
<td>IGYY26</td>
<td>IKI122</td>
</tr>
<tr>
<td>IGXY26</td>
<td>IKI123</td>
</tr>
<tr>
<td>IGYZ26</td>
<td>IKI124</td>
</tr>
<tr>
<td>IGZX26</td>
<td>IKI125</td>
</tr>
<tr>
<td>IGXX27</td>
<td>IKI126</td>
</tr>
<tr>
<td>IGYY27</td>
<td>IKI127</td>
</tr>
<tr>
<td>IGXY27</td>
<td>IKI128</td>
</tr>
<tr>
<td>IGYZ27</td>
<td>IKI129</td>
</tr>
<tr>
<td>IGZX27</td>
<td>IKI130</td>
</tr>
<tr>
<td>IGXX28</td>
<td>IKAT1</td>
</tr>
<tr>
<td>IGYY28</td>
<td>IKAT2</td>
</tr>
<tr>
<td>IGXY28</td>
<td>IKAT3</td>
</tr>
<tr>
<td>IGYZ28</td>
<td>IKAT4</td>
</tr>
<tr>
<td>IGZX28</td>
<td>IKAT5</td>
</tr>
<tr>
<td>IGXX29</td>
<td>IKAT6</td>
</tr>
<tr>
<td>IGYY29</td>
<td>IKAT7</td>
</tr>
<tr>
<td>IGXY29</td>
<td>IKAT8</td>
</tr>
<tr>
<td>IGYZ29</td>
<td>IKAT9</td>
</tr>
<tr>
<td>IGZX29</td>
<td>IKAT10</td>
</tr>
<tr>
<td>IGXX30</td>
<td>IKAT11</td>
</tr>
<tr>
<td>IGYY30</td>
<td>IKAT12</td>
</tr>
<tr>
<td>IGXY30</td>
<td>IKAT13</td>
</tr>
<tr>
<td>IGYZ30</td>
<td>IKAT14</td>
</tr>
<tr>
<td>IGZX30</td>
<td>IKAT15</td>
</tr>
<tr>
<td>IKI11</td>
<td>IKAT16</td>
</tr>
<tr>
<td>IKI12</td>
<td>IKAT17</td>
</tr>
<tr>
<td>IKI13</td>
<td>IKAT18</td>
</tr>
<tr>
<td>IKI14</td>
<td>IKAT19</td>
</tr>
<tr>
<td>IKI15</td>
<td>IKAT20</td>
</tr>
<tr>
<td>IKI16</td>
<td>IKAT21</td>
</tr>
<tr>
<td>IKI17</td>
<td>IKAT22</td>
</tr>
<tr>
<td>IKI18</td>
<td>IKAT23</td>
</tr>
<tr>
<td>IKI19</td>
<td>IKAT24</td>
</tr>
<tr>
<td>IKI110</td>
<td>IKAT25</td>
</tr>
<tr>
<td>IKI111</td>
<td>IKAT26</td>
</tr>
<tr>
<td>IKI112</td>
<td>IKAT27</td>
</tr>
<tr>
<td>IKI113</td>
<td>IKAT28</td>
</tr>
<tr>
<td>IKI114</td>
<td>IKAT29</td>
</tr>
<tr>
<td>IKI115</td>
<td>IKAT30</td>
</tr>
<tr>
<td>IKI116</td>
<td>IF1P1</td>
</tr>
<tr>
<td>IKI117</td>
<td>IF1P2</td>
</tr>
<tr>
<td>IKI118</td>
<td>IF1P3</td>
</tr>
</tbody>
</table>
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

$$[\sigma(t)] = p(t)I + [\sigma_0^{\text{dev}}(t)] +$$

$$\text{DEV} \left\{ \frac{1}{J} \left[ F(t) \right] \left\{ \int_0^t \frac{1}{G_0} \frac{dG}{d(t-\tau)} \left[ S_0^{\text{dev}}(\tau) \right] d\tau \right\} \left[ F(t) \right]^T \right\}$$

which is rewritten as follows:

$$[\sigma(t)] = p(t)I + [\sigma_0^{\text{dev}}(t)] +$$

$$\text{DEV} \left\{ \frac{1}{J} \left[ F(t) \right] \left\{ \int_0^{t^*} \frac{1}{G_0} \frac{dG}{d(t^*)} \left[ S_0^{\text{dev}}(t-\tau) \right] d\tau^* \right\} \left[ F(t) \right]^T \right\}$$

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

$$p = K \ln(J_m)$$

$J_m$ is the determinant of the mechanical part of $[F(t)]$, $[\sigma_0^{\text{dev}}(t)]$ is the deviatoric Cauchy stress at time $t$ computed using the elastic Swanson model with initial moduli values, $[S_0^{\text{dev}}(t-\tau)]$ 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 $[S_0^{\text{dev}}(t-\tau)]$) is the original configuration at $t = 0$. The shear relaxation modulus is represented using a Prony series as follows:

$$G(t) = G_0 \left\{ g_\infty + \sum_{i=1}^{N_0} g_i \exp \left[ -\frac{t}{\lambda_i} \right] \right\}$$

Finally, the reduced material time $t^*$ is related to the physical time $t$ as follows:
\[ dt^* = \frac{1}{A_{WLF} A_N} dt \]  \hspace{1cm} (69)

or

\[ t^* = \int_0^t \frac{dx}{A_{WLF}(x) A_N(x)} \]  \hspace{1cm} (70)

where \( A_{WLF} \) is the WLF shift factor given by

\[ \log_{10} A_{WLF} = \frac{-C_1 (T - T_{ref})}{C_2 + (T - T_{ref})} \]  \hspace{1cm} (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
- P1
- B1
- Q1
- C1
- R1
- BULK_MODULUS
- CUT_OFF_STRAIN
- TARGET_E
- MAX_POISSONS_RATIO
- BULK_SCALING
- SHEAR_SCALING
- PRONY_SHEAR_INFINITY
- PRONY_SHEAR_1
- PRONY_SHEAR_2
- PRONY_SHEAR_3
- PRONY_SHEAR_4
- PRONY_SHEAR_5
- SHEAR_RELAX_TIME_1
- SHEAR_RELAX_TIME_2
- SHEAR_RELAX_TIME_3
- SHEAR_RELAX_TIME_4
- SHEAR_RELAX_TIME_5
- SHEAR_RELAX_TIME_6
- SHEAR_RELAX_TIME_7
- SHEAR_RELAX_TIME_8
- SHEAR_RELAX_TIME_9
- SHEAR_RELAX_TIME_10
- WLF_COEF_C1
- WLF_COEF_C2
- 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
VSXXDEV
VSYYDEV
VSZZDEV
VSXYDEV
VSYZDEV
VSZXDEV
WLF_AAVG
ANA AVG
Functions:

NUMERICAL_SHIFT_FUNCTION
THERMAL_EXPANSION_FUNCTION

Methods:

initialize( matParams * p );
getStress( matParams * p );
loadStepInit( matParams * p );
peElasticModuli( 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
- SHEAR_FUNCTION
- BULK_FUNCTION
- RATE_FUNCTION
- EXPONENT_FUNCTION
- ALPHA_FUNCTION
- ISO_HARDENING
- ISO_RECOVERY
- KIN_HARDENING
- KIN_RECOVERY

State Variables (19):

<table>
<thead>
<tr>
<th>State Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - SVBXX</td>
<td></td>
</tr>
<tr>
<td>1 - SVBYY</td>
<td></td>
</tr>
<tr>
<td>2 - SVBZZ</td>
<td></td>
</tr>
<tr>
<td>3 - SVBXY</td>
<td></td>
</tr>
<tr>
<td>4 - SVBYZ</td>
<td></td>
</tr>
<tr>
<td>5 - SVBZX</td>
<td></td>
</tr>
<tr>
<td>8 - EQDOT</td>
<td></td>
</tr>
<tr>
<td>9 – COUNT</td>
<td></td>
</tr>
<tr>
<td>10 - SHEAR</td>
<td></td>
</tr>
</tbody>
</table>

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


<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>MS0346</td>
<td>Tom Baca</td>
<td>1523</td>
</tr>
<tr>
<td>MS0346</td>
<td>Robert Chambers</td>
<td>1524</td>
</tr>
<tr>
<td>MS0346</td>
<td>Jim Cox</td>
<td>1526</td>
</tr>
<tr>
<td>MS0346</td>
<td>Stephen Montgomery</td>
<td>1524</td>
</tr>
<tr>
<td>MS0346</td>
<td>Matthew Neidigk</td>
<td>1524</td>
</tr>
<tr>
<td>MS0346</td>
<td>Dave Reedy</td>
<td>1526</td>
</tr>
<tr>
<td>MS0346</td>
<td>Mike Starr</td>
<td>1526</td>
</tr>
<tr>
<td>MS0372</td>
<td>Lupe Arguello</td>
<td>1525</td>
</tr>
<tr>
<td>MS0372</td>
<td>Joe Bishop</td>
<td>1525</td>
</tr>
<tr>
<td>MS0372</td>
<td>Nicole Breivik</td>
<td>1524</td>
</tr>
<tr>
<td>MS0372</td>
<td>Frank Dempsey</td>
<td>1524</td>
</tr>
<tr>
<td>MS0372</td>
<td>Kristin Dion</td>
<td>1524</td>
</tr>
<tr>
<td>MS0372</td>
<td>Brenton Elisberg</td>
<td>1524</td>
</tr>
<tr>
<td>MS0372</td>
<td>Jeff Gruda</td>
<td>1524</td>
</tr>
<tr>
<td>MS0372</td>
<td>Kenneth Gwinn</td>
<td>1524</td>
</tr>
<tr>
<td>MS0372</td>
<td>Daniel Hammerand</td>
<td>1524</td>
</tr>
<tr>
<td>MS0372</td>
<td>Terry Hinnerichs</td>
<td>1524</td>
</tr>
<tr>
<td>MS0372</td>
<td>John Holland</td>
<td>1524</td>
</tr>
<tr>
<td>MS0372</td>
<td>David Lo</td>
<td>1524</td>
</tr>
<tr>
<td>MS0372</td>
<td>Don Longcope</td>
<td>1524</td>
</tr>
<tr>
<td>MS0372</td>
<td>Kurt Metzinger</td>
<td>1524</td>
</tr>
<tr>
<td>MS0372</td>
<td>Jake Ostien</td>
<td>1524</td>
</tr>
<tr>
<td>MS0372</td>
<td>John Pott</td>
<td>1524</td>
</tr>
<tr>
<td>MS0372</td>
<td>Jim Redmond</td>
<td>1525</td>
</tr>
<tr>
<td>MS0372</td>
<td>William Scherzinger</td>
<td>1524</td>
</tr>
<tr>
<td>MS0372</td>
<td>Mike Stone</td>
<td>1525</td>
</tr>
<tr>
<td>MS0372</td>
<td>Leah Tuttle</td>
<td>1524</td>
</tr>
<tr>
<td>MS0372</td>
<td>Gerald Wellman</td>
<td>1525</td>
</tr>
<tr>
<td>MS0376</td>
<td>James Bean</td>
<td>1524</td>
</tr>
<tr>
<td>MS0376</td>
<td>Jonathon Rath</td>
<td>1524</td>
</tr>
<tr>
<td>MS0380</td>
<td>Manoj Bhardwaj</td>
<td>1542</td>
</tr>
<tr>
<td>MS0380</td>
<td>Nathan Crane</td>
<td>1542</td>
</tr>
<tr>
<td>MS0380</td>
<td>Arne Gullerud</td>
<td>1542</td>
</tr>
<tr>
<td>MS0380</td>
<td>Jason Hales</td>
<td>1542</td>
</tr>
<tr>
<td>MS0380</td>
<td>Martin Heinstein</td>
<td>1542</td>
</tr>
<tr>
<td>MS0380</td>
<td>Joe Jung</td>
<td>1542</td>
</tr>
<tr>
<td>MS0380</td>
<td>Richard Koteras</td>
<td>1542</td>
</tr>
<tr>
<td>MS0380</td>
<td>Hal Morgan</td>
<td>1540</td>
</tr>
<tr>
<td>MS0380</td>
<td>Kendall Pierson</td>
<td>1542</td>
</tr>
<tr>
<td>MS0380</td>
<td>Vicki Porter</td>
<td>1542</td>
</tr>
<tr>
<td>MS0380</td>
<td>Garth Reese</td>
<td>1542</td>
</tr>
<tr>
<td>1</td>
<td>MS0380</td>
<td>Ben Spencer</td>
</tr>
<tr>
<td>1</td>
<td>MS0380</td>
<td>Tim Walsh</td>
</tr>
<tr>
<td>1</td>
<td>MS0382</td>
<td>Carter Edwards</td>
</tr>
<tr>
<td>1</td>
<td>MS0382</td>
<td>Mike Glass</td>
</tr>
<tr>
<td>1</td>
<td>MS0384</td>
<td>Art Ratzel</td>
</tr>
<tr>
<td>1</td>
<td>MS0555</td>
<td>Rod May</td>
</tr>
<tr>
<td>1</td>
<td>MS0557</td>
<td>Dave Clauss</td>
</tr>
<tr>
<td>1</td>
<td>MS0557</td>
<td>Dan Segalman</td>
</tr>
<tr>
<td>1</td>
<td>MS0660</td>
<td>Chris Lamb</td>
</tr>
<tr>
<td>1</td>
<td>MS0836</td>
<td>Shane Schumacher</td>
</tr>
<tr>
<td>1</td>
<td>MS0847</td>
<td>Marcus Billings</td>
</tr>
<tr>
<td>1</td>
<td>MS0847</td>
<td>Pavel Chaplya</td>
</tr>
<tr>
<td>1</td>
<td>MS0847</td>
<td>Mike Neilsen</td>
</tr>
<tr>
<td>1</td>
<td>MS0847</td>
<td>Pete Wilson</td>
</tr>
<tr>
<td>1</td>
<td>MS1070</td>
<td>Channy Wong</td>
</tr>
<tr>
<td>1</td>
<td>MS1070</td>
<td>Jordan Massad</td>
</tr>
<tr>
<td>1</td>
<td>MS1070</td>
<td>Phillip Reu</td>
</tr>
<tr>
<td>1</td>
<td>MS1070</td>
<td>Anton Sumali</td>
</tr>
<tr>
<td>1</td>
<td>MS9018</td>
<td>Central Technical Files</td>
</tr>
<tr>
<td>1</td>
<td>MS0899</td>
<td>Technical Library</td>
</tr>
</tbody>
</table>