LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-TR-233699

MatProps: Material Properties Database and Associated Access Library

J. K. Durrenberger, R. C. Becker, D. M. Goto, J. R. Neely, B. K. Wallin

August 15, 2007

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

MatProp: An XML Material Property Database and Associated Access Library (Version 1.0)

Kevin Durrenberger*, Richard Becker, Dana Goto, Rob Neely, Brad Wallin
Lawrence Livermore National Laboratory
August 2007

Coefficients for analytic constitutive and equation of state models (EOS), which are used by many hydro codes at LLNL, are currently stored in a legacy material database (Steinberg, UCRL-MA-106349). Parameters for numerous materials are available through this database, and include Steinberg-Guinan and Steinberg-Lund constitutive models for metals, JWL equations of state for high explosives, and Mie-Gruniesen equations of state for metals. These constitutive models are used in most of the simulations done by ASC codes today at Livermore. Analytic EOSs are also still used, but have been superseded in many cases by tabular representations in LEOS (http://leos.llnl.gov).

Numerous advanced constitutive models have been developed and implemented into ASC codes over the past 20 years. These newer models have more physics and better representations of material strength properties than their predecessors, and therefore more model coefficients. However, a material database of these coefficients is not readily available. Therefore incorporating these coefficients with those of the legacy models into a portable database that could be shared amongst codes would be most welcome. The goal of this paper is to describe the *MatProp* effort at LLNL to create such a database and associated access library that could be used by codes throughout the DOE complex and beyond.

We have written an initial version of the *MatProp* database and access library and our DOE/ASC code ALE3D (Nichols et. al., UCRL-MA-152204) is able to import information from the database. The database, a link to which exists on the Sourceforge server at LLNL, contains coefficients for many materials and models (see Appendix), and includes material parameters in the following categories – flow stress, shear modulus, strength, damage, and equation of state.

Future versions of the *Matprop* database and access library will include the ability to read and write material descriptions that can be exchanged between codes. It will also include an ability to do unit changes, i.e. have the library return parameters in user-specified unit systems. In addition to these, additional material categories can be added (e.g., phase change kinetics, etc.).

The *Matprop* database and access library is part of a larger set of tools used at LLNL for assessing material model behavior. One of these is *MSlib*, a shared constitutive material model library. Another is the Material Strength Database (*MSD*), which allows users to compare parameter fits for specific constitutive models to available experimental data. Together with Matprop, these tools create a suite of capabilities that provide state-of-the-art models and parameters for those models to integrated simulation codes.

This document is broken into several appendices. Appendix A contains a code example to retrieve several material coefficients. Appendix B contains the API for the *Matprop* data access library. Appendix C contains contains a list of the material names and model types currently

available in the *Matprop* database. Appendix D contains a list of the parameter names for the currently recognized model types. Appendix E contains a full xml description of the material Tantalum.

For additional information about *Matprop,* email Kevin Durrenberger at durrenberger1@llnl.gov

Appendix A: Code examples

Following are some examples of how material is accessed via the MatProp C++ API (a C/Fortran API is planned for the future). The underlying file format (XML) is invisible to the client. Many of these examples build on the ones prior

- Open a database of standard shared materials, access the material Tantalum

```
#include "MatProp.h"
using namespace MatProp ;
DataFile *db = DataFile::Open("/usr/apps/MatProps/data");
if (db->IsValid()) {
      Material *mpmat = db->GetMaterialByName("Tantalum");
}
else {
      cout << "Error reading database" << endl;
      DataFile::Close(db);
}
```

- Given a material, read some of the material properties

```
// Function style
double c0   = mpmat->GetProperty("c0");
double rho0 = mpmat->GetProperty("rho0");
// Call by reference style
double ezero;
mpmat->GetProperty("e0", ezero);
```

- Access a material using an identifier from a legacy database.

```
// Access material "25" using steinberg-guinan ID
Material *sgeos_mat = db->GetMaterialByLegacyId(MatProp::SGEOS, 25)
```

- Select the EOS model from that material and begin querying its parameters

```
EOSModel *mpeos = sgeos_mat->SetEOSModel("mie_gruniesen");
formNum = mpeos->GetIntParameter("eosform");
switch(formNum) {
      …
```

- Select the standard "blessed" types for a given material model, and load all of the parameters for that model

```
mpmat->SetStandardModels();
const char *eostype = mpmat->QueryEOSModelType();
cout << "Your current EOS model is " << eostype << endl;

EOSModel *eosm = material->GetEOSModel();
int nump, char **p;
eosm->QueryParameterList(p, nump);

for(int i=0 ; i<nump ; ++i) {
      hash->insert(p[i], eosm->GetParameter(p[i]);
}
```

Appendix B: *Matprop* data access library API

MatProp::DataFile Class Reference

The **DataFile** class provides an interface for accessing a **MatProp** "database" (which could be a single monolithic XML file, or a collection of files - one-per-material.

Public Member Functions

bool	**IsValid** () const *True if the database was successfully read in.*
void	**SetUnits** (**Units** units) *Allows the user to specify which units they prefer requests returned in.*
void	**SetVersion** (const char *version) *Set the version of data to retrieve.*
void	**Write** (const char *path) *Writes out a database.*
Material *	**GetMaterialByName** (const char *materialName) *Get a material by full name of the material.*
Material *	**GetMaterialByLegacyId** (**LEGACYID**, const char *) *Get **Material** by the legacy ID.*

Static Public Member Functions

DataFile *	**Open** (const char *) *Static method for loading a default DataBase in a specific directory.*
DataFile *	**Open** (const char *, const char *) *Static method for loading a DataBase by giving the file name and directory path.*
void	**Close** (**DataFile** *db) *Use this function in lieu of the destructor Closes out a database and destroys the object.*

MatProp::Material Class Reference

A **Material** is a collection of models, which together defines a consistent material model.

Public Member Functions

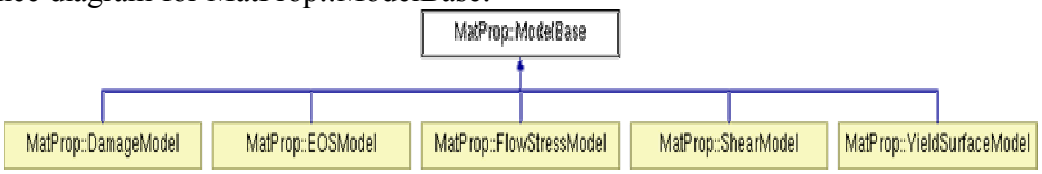
	Material (XNode *materialRoot) *Constructor.*
	~Material () *Destructor.*
const char *	**GetName** () const *Return the name of the material.*
void	**FreeName** (const char *matname) const *Free the memory allocated by the **GetName()** call.*
void	**QueryPropertyList** (char **&properties, int &numProperties) *Return a list of the property names in this material.*

void	**FreePropertyList** (char **&properties) const *Free resources (memory) associated with the propery list query.*
double	**GetProperty** (const char *propertyName) const *Get a property parameter of a material.*
int	**GetIntProperty** (const char *propertyName) const *Get an integer property parameter of a material.*
void	**GetProperty** (const char *propertyName, double &value) const *Get a property parameter of a material.*
void	**GetIntProperty** (const char *propertyName, int &value) const *Get an integer property parameter of a material.*
void	**SetStandardModels** (const char *shortcutName) *Specify a standard set of models for this material based on one of the "shortcuts".*
ModelBase *	**SetModel** (const char *modelType, const char *modelName) *Specify the type of model based on the modelType and modelName.*
EOSModel *	**SetEOSModel** (const char *modelName) *Specify the type of EOS (equation of state) model that this material should use.*
EOSModel *	**GetEOSModel** () *Return a pointer to the EOS model for this material.*
FlowStressModel *	**SetFlowStressModel** (const char *modelName) *Specify the type of flow stress model that this material should use.*
FlowStressModel *	**GetFlowStressModel** () *Return a pointer to the flow stress model for this material.*
DamageModel *	**SetDamageModel** (const char *modelName) *Specify the type of damage model that this material should use.*
DamageModel *	**GetDamageModel** () *Return a pointer to the damage model for this material.*
ShearModel *	**SetShearModel** (const char *modelName) *Specify the type of shear model that this material should use.*
ShearModel *	**GetShearModel** () *Return a pointer to the shear modulus model for this material.*
YieldSurfaceModel *	**SetYieldSurfaceModel** (const char *modelName) *Specify the type of yield surface model that this material should use.*
YieldSurfaceModel *	**GetYieldSurfaceModel** () *Return a pointer to the yield surface model for this material.*
const char *	**QueryEOSModelType** () const *Returns the string name of the EOS model defined for this material.*
const char *	**QueryFlowStressModelType** () const *Returns the string name of the flow stress model defined for this material.*
const char *	**QueryDamageModelType** () const *Returns the string name of the damage model defined for this material.*
const char *	**QueryShearModelType** () const *Returns the string name of the shear modulus model defined for this material.*
const char *	**QueryYieldSurfaceModelType** () const

Returns the string name of the yield surface model defined for this material.

MatProp::ModelBase Class Reference

Base class for supporting a number of methods.
Inheritance diagram for MatProp::ModelBase:



Public Member Functions

const char *	**GetModelTypeName** () const *Returns the string that defines the model type.*
void	**QueryParameterList** (char **&parameters, int &numParameters) *Return a list of the parameters supplied by this model.*
void	**GetParameterCount** (int &numParameters) *Return the count of the parameter contained in the model.*
void	**FreeParameterList** (char **&parameters) const *Free resources (memory) associated with the parameter list query.*
void	**QueryPolynomialList** (char **&polynomials, int &numPolynomials) *Returns an array of strings representing the available <polynomial>s to query on this material.*
void	**FreePolynomialList** (char **&polynomials) const *Free resources (memory) associated with the parameter list query.*
void	**GetPolynomialArray** (const char *name, double *&array, char **&names, int &size) const *Get a <polynomial> - coeficient names and values.*
void	**FreePolynomialArray** (double *&array, char **&names, const int size) *Free resources allocated by GetPolynomialArray.*
double	**GetParameter** (const char *name) const *Return a scalar <parameter> value.*
int	**GetIntParameter** (const char *name) const *Return an integer scalar <parameter> value.*
void	**GetParameter** (const char *name, double &value) const *Get a scalar <parameter> value (call by reference style).*
void	**GetIntParameter** (const char *name, int &value) const *Get an integer scalar <parameter> value (call by reference style).*
double	**GetPolynomialArrayEntry** (const char *name, const char *entryName) const *Get a specific entry of a <polynomial> by name.*
double	**GetPolynomialArrayEntry** (const char *name, const int exponent) const *Get a specific entry of a <polynomial> by subscript.*
bool	**SetUnits** (**Units** units) *Set the unit based on the Group(1.1 Milestone).*
bool	**SetUnit** (UnitType type, char *val) *Set a specific unit type to a specific value(1.1 Milestone).*

Appendix C: Current Materials and Model Combinations.

Current Materials and Models									
	Models								
	eos		flow_stress					shear_mod	damage
Material Name	Linear-Poly	Mie-Gruniesen	PTW	MTS	Steinberg-Lund	Steinberg-Guinan	Johnson-Cook	Steinberg-Guinan	Steinberg-Guinan
Aluminum 1100-O	*	*		*	*	*	*	*	*
Aluminum 2024-T4		*			*	*		*	*
Aluminum 6061-T6	*	*			*	*	*	*	*
Beryllium	*	*	*	*	*	*		*	*
Copper OFHC Half Hard	*	*			*	*	*	*	*
Gold		*			*	*		*	*
Graphite		*			*	*		*	*
Kel-F C2F3Cl		*			*	*		*	*
Lead		*			*	*		*	*
Lexan		*			*	*		*	*
Lithium		*			*	*		*	*
Lithium Fluoride		*			*	*		*	*
Lucite		*			*	*		*	*
Mercury		*			*	*		*	*
Magnesium AZ31B	*	*			*	*		*	*
Micarta		*			*	*		*	*
Molybdenum	*	*	*	*	*	*	*	*	*
Monel 66%Ni-29%Cu-3%Al-1/2%Ti		*			*	*		*	*
Fansteel 85 - 61%Nb-28%Ta-10%W -1%Zr		*			*	*		*	*
Nickel	*	*			*	*	*	*	*
Niobium		*			*	*		*	*
Platinum		*			*	*		*	*
Polypentene		*			*	*		*	*
Silastic		*			*	*		*	*
Tantalum - 10% Tungsten	*	*			*	*		*	*
Tantalum	*	*	*	*	*	*	*	*	*
Teflon		*			*	*		*	*
Thorium		*			*	*		*	*
Thulium		*			*	*		*	*
Titanium -6%Aluminum -4% Vanadium	*	*			*	*	*	*	*

Current Materials and Models(continued)									
	eos		flow_stress					shear_mod	damage
Material Name	Linear-Poly	Mie-Gruniesen	PTW	MTS	Steinberg-Lund	Steinberg-Guinan	Johnson-Cook	Steinberg-Guinan	Steinberg-Guinan
Tin		*			*	*		*	
Titanium		*			*	*	*	*	*
Tungsten	*	*		*	*	*	*	*	*
Tungsten Carbide		*			*	*		*	*
Uranium - .75% Titanium		*			*	*	*	*	*
Uranium - 5% Molybdenum		*			*	*		*	*
Uranium - 6% Noibium							*		
Uranium	*	*	*		*	*		*	*
Vanadium	*	*	*		*	*		*	*
Tungsten - 3.5% Nickel - 1.5% Iron		*			*	*	*	*	*
Steel 4340	*						*		
Stainless Steel 304	*	*	*		*	*		*	*
Steel Vascomax 250		*			*	*		*	*
Stainless Steel 21-6-9		*	*		*	*	*	*	*

Appendix D: Current Material Properties and Model Parameters.

Material Properties		
Parameter	**Description**	**Units***
rho0	Reference density	g/cm^3
v0	Reference (specific) volume	cm^3/g
e0	Reference (internal) energy	eu/g
pmin	Minimum pressure at which material fails	Mbar
bhe	HE parameter related to "volume" at the C-J condition	-
ayz	Atomic weight	g/mole

Material Models			
Parameter	**Equation variable**	**Description**	**Units***
Model: Mie-Gruneisen Equation Of State			
c0	C_0	Bulk sound speed	cm/μs
s1	S_1	Hugoniot linear slope coefficient	-
s2	S_2	Hugoniot quadratic slope coefficient	-
s3	S_3	Hugoniot cubic slope coefficient	-
gamma0	γ_0	Initial value of Gruneisen gamma	-
b	b	Compression dependence of γ	-
ecxfit(ec0 ...ec9)	E^c_i	Cold curve polynomial coefficients (energy per reference volume)	eu/cm^3
emxfit(em0 ... em9)	E^m_i	Melt curve polynomial coefficients (energy per reference volume)	eu/cm^3
etamin		Minimum η used in cold and melt curve calcs	-
etamax		Maximum η used in cold and melt curve calcs	-
Model: Steinberg-Guinan Shear Modulus			
cmu	G_0	Reference shear modulus at 300K	Mbar
au	a	Pressure dependence of the shear modulus.	$Mbar^{-1}$
bu	b	Temperature dependence of the shear modulus.	K^{-1}
yp	y_p	Parameter in melt transition.	-
begr		Minimum η used in shear modulus calculation	-
endr		Maximum η used in shear modulus calculation	-
Model: Steinberg-Guinan Flow Stress			
y	Y_0	Yield strength at Hugoniot elastic limit	Mbar
yb	β	Work hardening parameter	-
yc	n	Work hardening exponent	-
gam0	ε_0	Initial plastic strain	-
ywhmx	Y_{max}	Work hardening maximum	Mbar
Model: Steinberg-Lund Flow Stress			

Material Models			
Parameter	**Equation variable**	**Description**	**Units***
y	Y_0	Yield strength at Hugoniot elastic limit	Mbar
yb	β	Work hardening parameter	-
yc	n	Work hardening exponent	-
gam0	ε_0	Initial plastic strain	-
ywhmx	Y_{max}	Work hardening maximum	Mbar
c1	C_1	Reference strain rate in thermal activation regime	μs^{-1}
c2	C_2	Coefficient in the drag regime	Mbar-μs
uk	U_k	Activation energy/over Boltzmann constant	eV
ypeierls	Y_p	Peierls stress	Mbar
ya	Y_A	Athermal yield strength	Mbar
ystrmx	Y^*_{max}	Work hardening max for rate-dependent term	Mbar
Model: Johnson-Cook Flow Stress			
jc_a	A	Constant contribution to strength	Mbar
jc_b	B	Coefficient for strain hardening	Mbar
jc_c	C	Multiplier on the log strain rate term	-
jc_m	m	Exponent on temperature term	-
jc_n	n	Hardening exponent	-
jc_p	C_p	Linear pressure dependence multiplier	-
tmelt	T_{melt}	Melt temperature	K
troom	T_{room}	Ambient room temperature	K
epsd0	$\acute{\varepsilon}_0$	Strain-rate normalization factor	μs^{-1}
epsd_cut	-	Minimum strain rate used in evaluations	μs^{-1}
Model: Steinberg-Guinan Damage			
spall		Specifies that the material can experience spall. When set to one (or non-zero), when the maximum principal stress exceeds |**pmin**| the material is defined to have spalled.	-
dam0		Damage length ?	cm
pcrush		Crush strength	Mbar
Model: Preston-Tonks-Wallace (PTW) Flow Stress			
molar_weight	M	Molar weight	g/mole
kappa	κ	Scale factor for dimensionless temperature	-
gamma	γ	Scale factor for strain rate	-
theta	θ	Initial hardening rate	-
p	p	Hardening rate parameter	-
s0	s_0	Saturation stress at 0K	-
s_inf	s_∞	Saturation stress at high temperature	-
y0	y_0	Yield strength at 0K	-
y_inf	y_∞	Yield strength at high temperature	-
y1	y_1	Stress multiplier for yield model in drag regime	-
y2	y_2	Exponent for yield model in drag regime	-
beta	β	Exponent for saturation term in drag regime	-

Material Models			
Parameter	**Equation variable**	**Description**	**Units***
g0	G_0	Shear modulus at 0K	Mbar
alpha	α	Temperature dependent shear modulus factor	-
tmelt	T_{melt}	Melt temperature	K
epsd_cut	-	Minimum strain rate used in evaluations	μs^{-1}
Model: Mechanical Threshold Stress (MTS) Flow Stress			
kob3	k/b^3	b= burgers vector , k=Boltzmann's constant	Mbar/K
sig_a	$\hat{\sigma}_a$	Athermal contribution to yield stress	Mbar
sig_s	$\hat{\sigma}_s$	Threshold stress contribution from solute	Mbar
g0_s	$g0_s$	Normalized activation energy: solute	-
epsd_s	$\acute{\varepsilon}_s$	Base strain rate: solute	μs^{-1}
q_s	q_s	Exponential in rate expression: solute	
p_s	p_s	Exponential in rate expression: solute	
sig_i	$\hat{\sigma}_i$	Threshold stress contribution from interstitial	Mbar
g0_i	$g0_i$	Normalized activation energy: interstitial	-
epsd_i	$\acute{\varepsilon}_i$	Base strain rate: interstitial	μs^{-1}
q_i	q_i	Exponential in rate expression: interstitial	-
p_i	p_i	Exponential in rate expression: interstitial	-
g0_eps	g_{0d}	Normalized activation energy: dislocation	-
epsd_eps	$\acute{\varepsilon}_s$	Base strain rate: dislocation	μs^{-1}
q_eps	q_d	Exponential in rate expression: dislocation	-
p_eps	p_d	Exponential in rate expression: dislocation	-
sighat0	$\hat{\sigma}_s$	Initial vale of the evolving threshold stress	Mbar
sig_s0	$\hat{\sigma}_{s0}$	Base value for saturation calculation	Mbar
g0_eps_s	$g0_{ds}$ or A	Normalized activation energy for saturation	μs^{-1}
epsd_eps_s	$\acute{\varepsilon}_{ds}$	Base strain rate for saturation expression	μs^{-1}
hard_form	-	Specify form for hardening evolution, F.	-
alpha	α	Coefficient in threshold stress evolution model	-
hard_power	ξ		-
c1	C_1	Constant hardening rate coefficient	Mbar
c2	C_2	Coefficient on the logarithmic hardening rate	Mbar
c3	C_3	Coefficient on the parabolic hardening rate	Mbar
c4	C_4	Coefficient on the linear hardening rate	Mbar
c5	C_5	Coefficient on the rate/temperature dependent hardening	Mbar
c6	C_6	Temperature coefficient in exponent of rate/temperature hardening	K^{-1}
csat	C_{sat}	Coefficient for reducing saturation stress	-
timeunit	τ	Time unit normalization ; 1.0 for s, 1.e6 for μs	μs
epsd_cut	-	Minimum strain rate used in evaluations	μs^{-1}
Model: Linear Polynomial EOS			
a0	a_0	Pressure at zero compression and energy	Mbar

Material Models			
Parameter	**Equation variable**	**Description**	**Units***
a1	a_1	Linear dependence of pressure on compression	Mbar
a2	a_2	Quadratic compression term	Mbar
a3	a_3	Cubic compression term	Mbar
b0	b_0	Base Gruneisen coefficient, Γ	-
b1	b_1	Linear dependence of Γ on compression	-
b2	b_2	Quadratic dependence of Γ on compression	-

*** The Units are the units stored specifically in the database. We will allow users the ability to retrieve the units from the database in different units if desired. The desired units must be compatible with the existing units (i.e. cm -> feet, or grams-> kilograms)**

Model References

1. Maudlin, Davidson and Henninger, *Implementation and Assessment of the Mechanical-Threshold-Stress Model Using the EPIC2 and PINON Computer Codes*, LA-11895-MS, Sept. 1990
2. Follasbee, P.S. and Kocks, U.F., *A constitutive description of the deformation of copper based on the used of the mechanical threshold stress as an internal state variable*, Acta Metall., 36(1) 81-93 (1988)
3. Steinberg, D.J., Cochran, S.G. and Guinan, M.W., *A constitutive model for metals applicable at high-strain rate*, J. Appl. Phys. 51(3) 1498-1504 (1980)
4. Steinberg, D.J. and Lund, C.M., *A constitutive model for strain rates from 10^{-4} to 10^{6} s^{-1}*, J. Appl. Phys. 65(4) 1528-1533
5. Zerilli, F.J. and Armstrong, R.W., *Dislocation-mechanics-based constitutive relations for material dynamics calculations*, J. Appl. Phys. 61(5) 1816-1825 (1987)
6. Zerilli, F.J. and Armstrong, R.W., *Description of tantalum deformation behavior by dislocation mechanics based constitutive relations*, J. Appl. Phys. 68(4) 1580-1591 (1990)
7. Johnson, G.R. and Cook, W.H., *A constitutive model and data for metals subjected to large strain, high strain rates and high temperatures*, Proceedings of the Seventh International Symposium on Ballistics, The Hague, The Netherlands, pp. 541-547, April 1983
8. Preston, D.L., Tonks, D.L. and Wallace, D.C., *Model of plastic deformation for extreme loading conditions*, J. Appl. Phys. 93(1) 211-220 (2003)
9. Steinberg, D.J., *Equation of state and strength properties of selected materials*, UCRL-MA-106439 (1996)

Appendix E: Abbreviated XML Database example

We are currently working on units and unit conversion for the 1.1 release of the code. Many of the current parameters do not have their unit assigned as yet.

```
<?xml version="1.0" encoding="UTF-8"?>
<matprops type="data" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="matprops.xsd">
<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
File: Al6061-T6.xml
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
  <material file_type="local" name="Alumninum 6061-T6">
    <alias name="Alumninum 6061-T6"/>
    <alias name="6061-T6"/>
    <legacy_id name=”sgeos” legacy_id=”25”/>
    <material_specifier>Al 6061-T6</material_specifier>
    <classification>Export Controlled</classification>
    <maintained_by>Dana Goto</ maintained_by>
    <revision_date>2007-04-10</revision_date>
    <shortcut name="sgeos" version="default">
      <shortcut_model version="default" type="eos" name="mie_gruniesen"/>
     <shortcut_model version="default" type="shear_mod" name="steinberg-guinan"/>
     <shortcut_model version="default" type="flow_stress" name="steinberg-guinan"/>
     <shortcut_model version="default" type="damage" name="steinberg-guinan"/>
    </shortcut>
    <properties>
      <parameter name="rho0">
        <value date="2007-04-10" units="" value="2.703000e+00" version="default"/>
      </parameter>
      <parameter name="v0">
        <value date="2007-04-10" units="" value="1.000000e+00" version="default"/>
      </parameter>
        ....
    </properties>
    <models>
      <model name="mie-gruniesen" type="eos">
        <polynomial name="ecxfit" terms="10">
          <variable exponent="0" name="ec0">
            <value date="2007-04-10" value="-7.48204639e-03" version="default"/>
          </variable>
          <variable exponent="1" name="ec1">
            <value date="2007-04-10" value="-1.49144219e-02" version="default"/>
          </variable>
                ....
        </polynomial>
        <parameter name="c0">
          <value date="2007-04-10" value="5.24000000e-01" version="default"/>
        </parameter>
        <parameter name="s1">
          <value date="2007-04-10" value="1.40000000e+00" version="default"/>
        </parameter>
              ...
      </model>
      <model name="steinberg-guinan" type="shear_mod">
        <parameter name="cmu">
          <value date="2007-04-10" value="2.760000e-01" version="default"/>
        </parameter>
        <parameter name="au">
          <value date="2007-04-10" units="" value="6.5200e+00" version="default"/>
        </parameter>
              ....
      </model>
              ...
    </models>
  </material>
</matprops>
```