Lecture 6

Writing a UMAT or VUMAT

Overview

- Motivation
- Steps Required in Writing a UMAT or VUMAT
- UMAT Interface
- Examples
- VUMAT Interface
- Examples

Overview

- ABAQUS/Standard and ABAQUS/Explicit have interfaces that allow the user to implement general constitutive equations.
  - In ABAQUS/Standard the user-defined material model is implemented in user subroutine **UMAT**.
  - In ABAQUS/Explicit the user-defined material model is implemented in user subroutine **VUMAT**.
- Use **UMAT** and **VUMAT** when none of the existing material models included in the ABAQUS material library accurately represents the behavior of the material to be modeled.

材料模型

- These interfaces make it possible to define any (proprietary) constitutive model of arbitrary complexity.
- User-defined material models can be used with any ABAQUS structural element type.
- Multiple user materials can be implemented in a single **UMAT** or **VUMAT** routine and can be used together.

In this lecture the implementation of material models in **UMAT** or **VUMAT** will be discussed and illustrated with a number of examples.

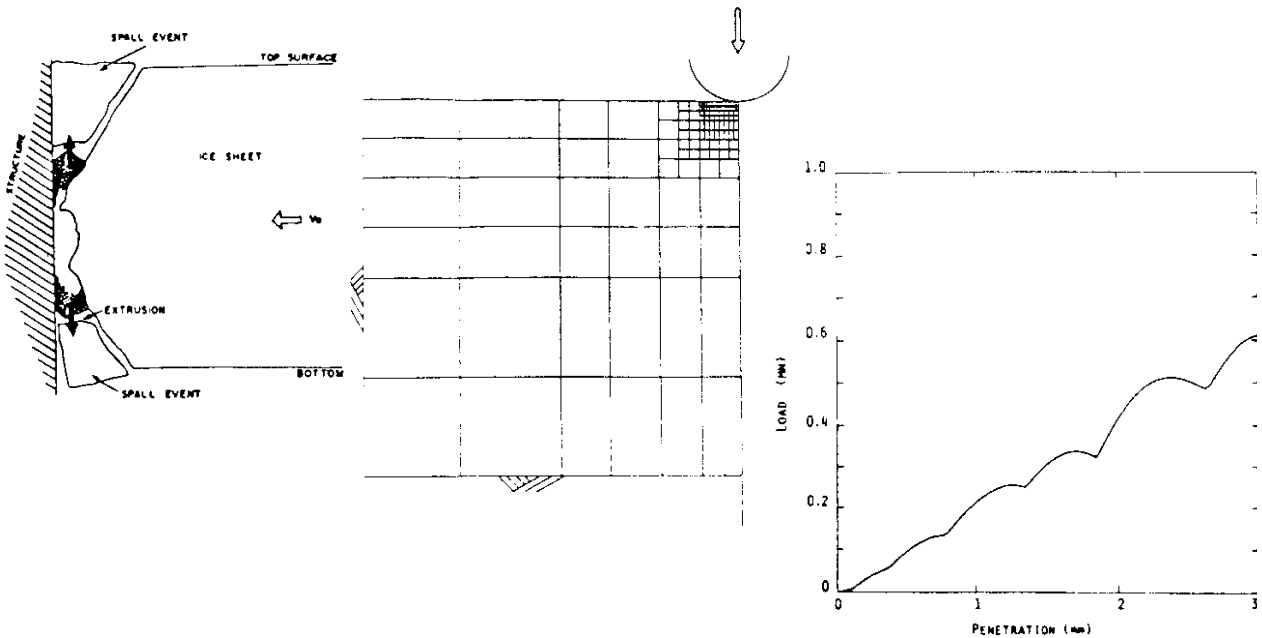
Motivation

- Proper testing of advanced constitutive models to simulate experimental results often requires complex finite element models.
  - Advanced structural elements
  - Complex loading conditions
  - Thermomechanical loading 热载荷
  - Contact and friction conditions
  - Static and dynamic analysis

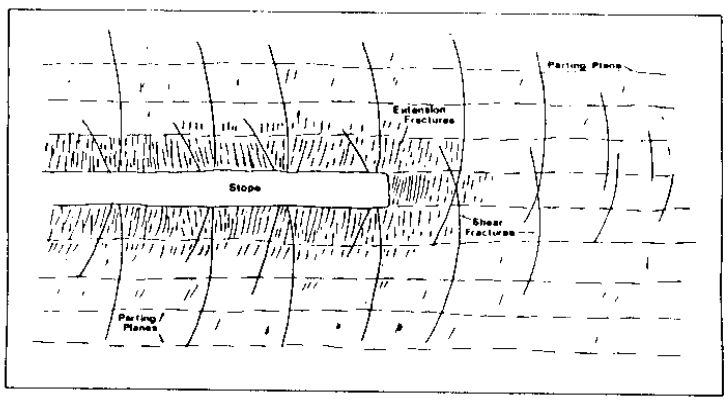
- Special analysis problems occur if the constitutive model simulates material instabilities and localization phenomena.
  - Special solution techniques are required for quasi-static analysis.
  - Robust element formulations should be available.
  - Explicit dynamic solution algorithms with robust, vectorized contact algorithms are desired.
- In addition, robust features are required to present and visualize the results.
  - Contour and path plots of state variables.
  - *X*–*Y* plots.
  - Tabulated results.

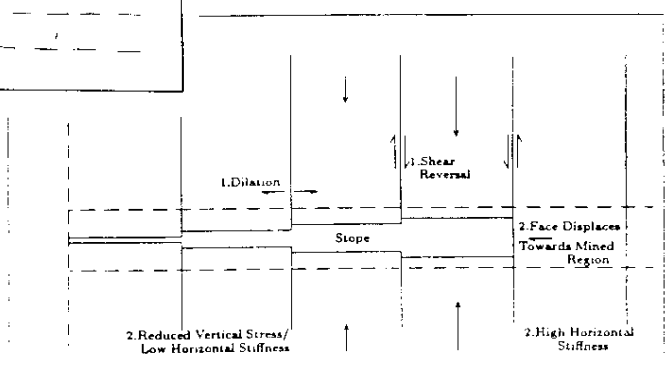
- The material model developer should be concerned only with the development of the material model and not the development and maintenance of the FE software.
  - Developments unrelated to material modeling
  - Porting problems with new systems
  - Long-term program maintenance of user-developed code

- "*Finite Element Modelling of the Damage Process in Ice*," R. F. McKenna, I. J. Jordaan, and J. Xiao, ABAQUS Users' Conference Proceedings, 1990.

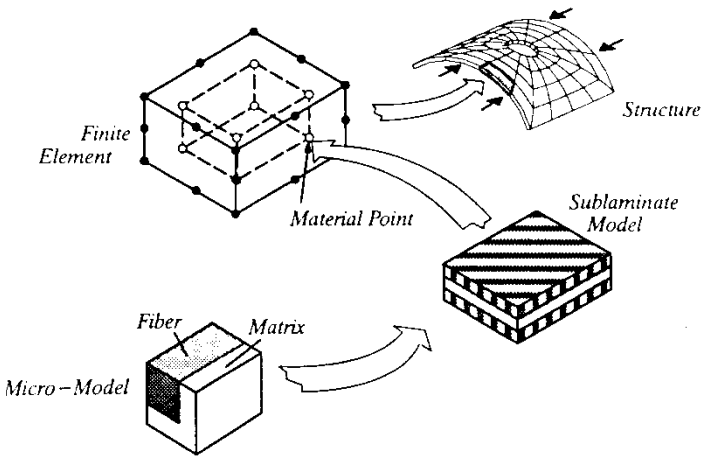


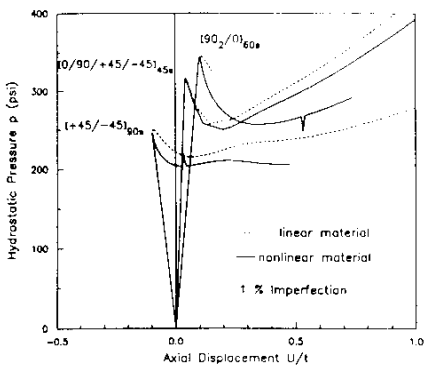
- "*The Numerical Simulation of Excavations in Deep Level Mining*," M. F. Snyman, G. P. Mitchell, and J. B. Martin, ABAQUS Users' Conference Proceedings, 1991.

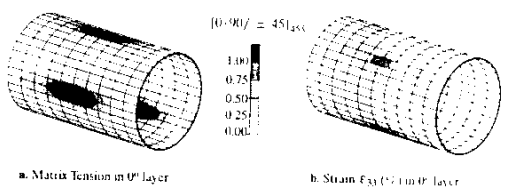




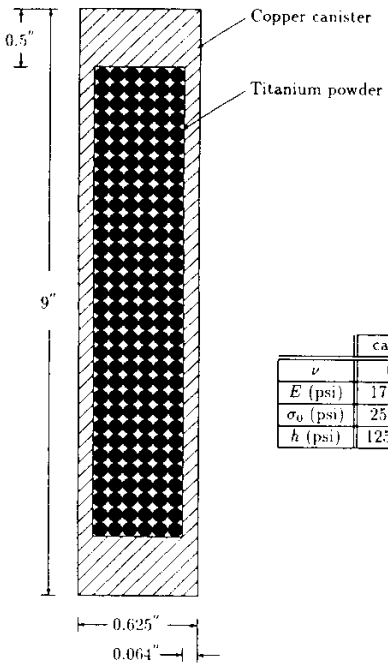
- "*Combined Micromechanical and Structural Finite Element Analysis of Laminated Composites*," R. M. HajAli, D. A. Pecknold, and M. F. Ahmad, ABAQUS Users' Conference Proceedings, 1993.



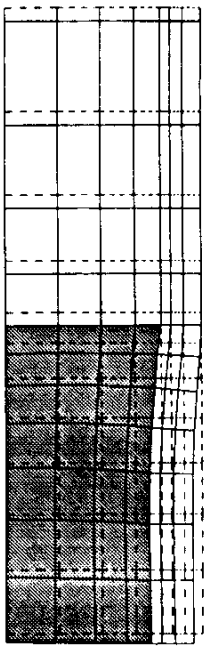




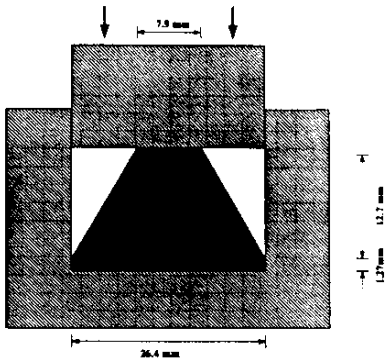
- "*Deformation Processing of Metal Powders: Cold and Hot Isostatic Pressing*," R. M. Govindarajan and N. Aravas, private communication, 1993.

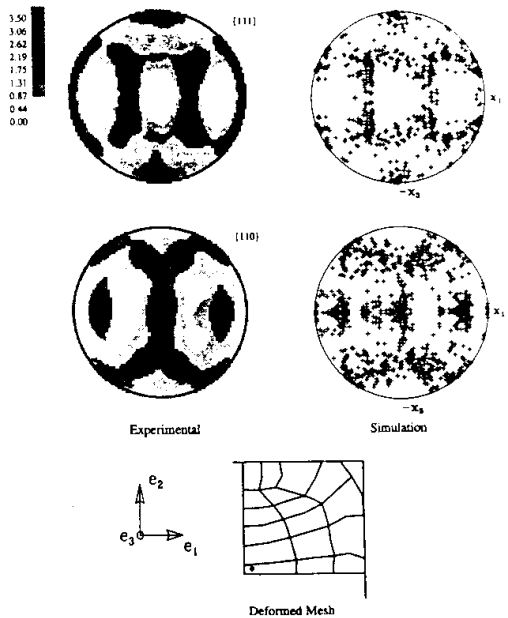


	canister	powder
ν	0.33	0.327
E (psi)	17×10^6	16.1×10^6
σ_0 (psi)	25×10^3	62×10^3
h (psi)	125×10^3	688×10^3



- "*Macroscopic Shape Change and Evolution of Crystallographic Texture in Pre-textured FCC Metals*," S. R. Kalidindi and Anand, Acta Metallurgica, 1993.





Steps Required in Writing a UMAT or VUMAT

- Proper definition of the constitutive equation, which requires one of the following:
  - Explicit definition of stress (Cauchy stress for large-strain applications)
  - Definition of the stress rate only (in corotational framework)
- Furthermore, it is likely to require:
  - Definition of dependence on time, temperature, or field variables
  - Definition of internal state variables, either explicitly or in rate form

- Transformation of the constitutive rate equation into an incremental equation using a suitable integration procedure:
  - Forward Euler (explicit integration)
  - Backward Euler (implicit integration)
  - Midpoint method

This is the hard part! Forward Euler (explicit) integration methods are simple but have a *stability limit*,

$$|\Delta\varepsilon| < \Delta\varepsilon_{stab},$$

where $\Delta\varepsilon_{stab}$ is usually less than the elastic strain magnitude.

- For explicit integration the time increment must be controlled.
- For implicit or midpoint integration the algorithm is more complicated and often requires local iteration. However, there is usually no stability limit.
- An incremental expression for the internal state variables must also be obtained.

- Calculation of the (consistent) Jacobian (required for ABAQUS/Standard **`UMAT`** only).
  - For small-deformation problems (e.g., linear elasticity) or large-deformation problems with small volume changes (e.g., metal plasticity), the consistent Jacobian is

$$C = \frac{\partial \Delta\sigma}{\partial \Delta\varepsilon},$$

    where $\Delta\sigma$ is the increment in (Cauchy) stress and $\Delta\varepsilon$ is the increment in strain. (In finite-strain problems, ε is an approximation to the logarithmic strain.)

    – This matrix may be nonsymmetric as a result of the constitutive equation or integration procedure.
    – The Jacobian is often approximated such that a loss of quadratic convergence may occur.

- It is easily calculated for forward integration methods (usually the elasticity matrix).
- If large deformations with large volume changes are considered (e.g., pressure-dependent plasticity), the exact form of the consistent Jacobian

$$C = \frac{1}{J}\frac{\partial \Delta(J\sigma)}{\partial \Delta\varepsilon}$$

should be used to ensure rapid convergence. Here, J is the determinant of the deformation gradient.

- Hyperelastic constitutive equations
  - Total-form constitutive equations relating the Cauchy stress σ and the deformation gradient F are commonly used to model, for example, rubber elasticity.
  - In this case, the consistent Jacobian is defined through:

$$\delta(J\sigma) = JC{:}\delta D,$$

    where $J = |F|$, C is the material Jacobian, and δD is the virtual rate of deformation, defined as

$$\delta D = sym(\delta F \cdot F^{-1}).$$

编码要求UMAT

- Coding the **UMAT** or **VUMAT**:
  - Follow FORTRAN 77 or C conventions.
  - Make sure that the code can be vectorized (for **VUMAT** only, to be discussed later).
  - Make sure that all variables are defined and initialized properly.
  - Use ABAQUS utility routines as required.
  - Assign enough storage space for state variables with the *DEPVAR option.

用一个小输入文件与检验 UMAT

- Verifying the **UMAT** or **VUMAT** with a small (one element) input file.
  1. Run tests with all displacements prescribed to verify the integration algorithm for stresses and state variables. Suggested tests include:
     - Uniaxial
     - Uniaxial in oblique direction
     - Uniaxial with finite rotation
     - Finite shear
  2. Run similar tests with load prescribed to verify the accuracy of the Jacobian.
  3. Compare test results with analytical solutions or standard ABAQUS material models, if possible. If the above verification is successful, apply to more complicated problems.

UMAT Interface 界面

- These input lines act as the interface to a **UMAT** in which isotropic hardening plasticity is defined.

```
*MATERIAL, NAME=ISOPLAS
*USER MATERIAL, CONSTANTS=8, (UNSYMM)
30.E6, 0.3, 30.E3, 0., 40.E3, 0.1, 50.E3, 0.5
*DEPVAR
13
*INITIAL CONDITIONS, TYPE=SOLUTION
```

Data line to specify initial solution-dependent variables

USER SUBROUTINES,(INPUT=file_name***)**

- The *USER MATERIAL option is used to input material constants for the **UMAT**. The unsymmetric equation solution technique will be used if the UNSYMM parameter is used.

输入材料常数

非对称方程

- The *DEPVAR option is used to allocate space at each material point for solution-dependent state variables (SDVs).
- The *INITIAL CONDITIONS, TYPE=SOLUTION option is used to initialize SDVs if they are starting at a nonzero value.
- Coding for the **UMAT** is supplied in a separate file. The **UMAT** is invoked with the ABAQUS execution procedure, as follows:

```
abaqus job=... user=....
```

  - The user subroutine must be invoked in a restarted analysis because user subroutines are not saved on the restart file.

- Additional notes:
  - If a constant material Jacobian is used and no other nonlinearity is present, reassembly can be avoided by invoking the quasi-Newton method with the input line

```
*SOLUTION TECHNIQUE, REFORM KERNEL=n
```

  - *n* is the number of iterations done without reassembly.
  - This does not offer advantages if other nonlinearities (such as contact changes) are present.

- Solution-dependent state variables can be output with identifiers SDV1, SDV2, etc. Contour, path, and *X*–*Y* plots of SDVs can be plotted in ABAQUS/Viewer.
- Include only a single **`UMAT`** subroutine in the analysis. If more than one material must be defined, test on the material name in **`UMAT`** and branch.

- The **UMAT** subroutine header is shown below:

```
      SUBROUTINE UMAT(STRESS, STATEV, DDSDDE, SSE, SPD, SCD, RPL,
     1 DDSDDT, DRPLDE, DRPLDT, STRAN, DSTRAN, TIME, DTIME, TEMP, DTEMP,
     2 PREDEF, DPRED, CMNAME, NDI, NSHR, NTENS, NSTATV, PROPS, NPROPS,
     3 COORDS, DROT, PNEWDT, CELENT, DFGRD0, DFGRD1, NOEL, NPT, LAYER,
     4 KSPT, KSTEP, KINC)

C
      INCLUDE 'ABA_PARAM.INC'
C
      CHARACTER*8 CMNAME
C
      DIMENSION STRESS(NTENS), STATEV(NSTATV), DDSDDE(NTENS, NTENS),
     1 DDSDDT(NTENS), DRPLDE(NTENS), STRAN(NTENS), DSTRAN(NTENS),
     2 PREDEF(1), DPRED(1), PROPS(NPROPS), COORDS(3), DROT(3, 3),
     3 DFGRD0(3, 3), DFGRD1(3, 3)
```

  - The include statement sets the proper precision for floating point variables (**REAL*8** on most machines).
  - The material name, **CMNAME**, is an 8-byte character variable.

UMAT Variables

- The following quantities are available in **`UMAT`**:
  - Stress, strain, and SDVs at the start of the increment
  - Strain increment, rotation increment, and deformation gradient at the start and end of the increment
  - Total and incremental values of time, temperature, and user-defined field variables
  - Material constants, material point position, and a characteristic element length
  - Element, integration point, and composite layer number (for shells and layered solids)
  - Current step and increment numbers

- The following quantities must be defined:
  - Stress, SDVs, and material Jacobian
- The following variables may be defined:
  - Strain energy, plastic dissipation, and “creep” dissipation
  - Suggested new (reduced) time increment

Complete descriptions of all parameters are provided in the **`UMAT`** section in Chapter 24 of the ABAQUS/Standard User’s Manual.

- The header is usually followed by dimensioning of local arrays. It is good practice to define constants via parameters and to include comments.

```
      DIMENSION EELAS(6), EPLAS(6), FLOW(6)
C
      PARAMETER(ZERO=0.D0, ONE=1.D0, TWO=2.D0, THREE=3.D0, SIX=6.D0,
     1          ENUMAX=.4999D0, NEWTON=10, TOLER=1.0D-6)
C
C -------------------------------------------------------------
C    UMAT FOR ISOTROPIC ELASTICITY AND ISOTROPIC MISES PLASTICITY
C    CANNOT BE USED FOR PLANE STRESS
C -------------------------------------------------------------
C    PROPS(1) - E
C    PROPS(2) - NU
C    PROPS(3..) - YIELD AND HARDENING DATA
C    CALLS UHARD FOR CURVE OF YIELD STRESS VS. PLASTIC STRAIN
C -------------------------------------------------------------
```

  - The **PARAMETER** assignments yield accurate floating point constant definitions on any platform.

UMAT Utilities

- Utility routines **SINV**, **SPRINC**, **SPRIND**, and **ROTSIG** can be called to assist in coding **UMAT**.
  - **SINV** will return the first and second invariants of a tensor.
  - **SPRINC** will return the principal values of a tensor.
  - **SPRIND** will return the principal values and directions of a tensor.
  - **ROTSIG** will rotate a tensor with an orientation matrix.
  - **XIT** will terminate an analysis and close all files associated with the analysis properly.
- For details regarding the arguments required in making these calls, refer to the **UMAT** section in Chapter 24 of the ABAQUS/Standard User's Manual and the examples in this lecture.

UMAT Conventions

- Stresses and strains are stored as vectors.
  - For plane stress elements: σ_{11}, σ_{22}, σ_{12}.
  - For (generalized) plane strain and axisymmetric elements: σ_{11}, σ_{22}, σ_{33}, σ_{12}.
  - For three-dimensional elements: σ_{11}, σ_{22}, σ_{33}, σ_{12}, σ_{13}, σ_{23}.
- The shear strain is stored as engineering shear strain,

$$\gamma_{12} = 2\varepsilon_{12}.$$

- The deformation gradient, F_{ij}, is always stored as a three-dimensional matrix.

UMAT Formulation Aspects

- For geometrically nonlinear analysis the strain increment and incremental rotation passed into the routine are based on the Hughes-Winget formulae.
  - Linearized strain and rotation increments are calculated in the mid-increment configuration.
  - Approximations are made, particularly if rotation increments are large: more accurate measures can be obtained from the deformation gradient if desired.
- The user must define the Cauchy stress: when this stress reappears during the next increment, it will have been rotated with the incremental rotation, **`DROT`**, passed into the subroutine.
  - The stress tensor can be rotated back using the utility routine **`ROTSIG`** if this is not desired.

- If the *ORIENTATION option is used in conjunction with **UMAT**, stress and strain components will be in the local system (again, this basis system rotates with the material in finite-strain analysis).
- Tensor state variables must be rotated in the subroutine (use **ROTSIG**).
- If **UMAT** is used with reduced-integration elements or shear flexible shell or beam elements, the hourglass stiffness and the transverse shear stiffness must be specified with the *HOURGLASS STIFFNESS and *TRANSVERSE SHEAR STIFFNESS options, respectively.

Usage Hints

- At the start of a new increment, the strain increment is extrapolated from the previous increment.
  - This extrapolation, which may sometimes cause trouble, is turned off with ∗STEP, EXTRAPOLATION=NO.
- If the strain increment is too large, the variable **`PNEWDT`** can be used to suggest a reduced time increment.
  - The code will abandon the current time increment in favor of a time increment given by **`PNEWDT*DTIME`**.
- The characteristic element length can be used to define softening behavior based on fracture energy concepts.

各向同性等温弹性

Example 1: Isotropic Isothermal Elasticity

Governing Equations

- Isothermal elasticity equation (with Lamé's constants):

$$\sigma_{ij} = \lambda\delta_{ij}\varepsilon_{kk} + 2\mu\varepsilon_{ij},$$

  or in a Jaumann (corotational) rate form:

$$\dot{\sigma}^{J}_{ij} = \lambda\delta_{ij}\dot{\varepsilon}_{kk} + 2\mu\dot{\varepsilon}_{ij}.$$

- The Jaumann rate equation is integrated in a corotational framework:

$$\Delta\sigma^{J}_{ij} = \lambda\delta_{ij}\Delta\varepsilon_{kk} + 2\mu\Delta\varepsilon_{ij}.$$

  The appropriate coding is shown on the following pages.

Coding for Isotropic Isothermal Elasticity

```
C -------------------------------------------------------------------
C    UMAT FOR ISOTROPIC ELASTICITY
C    CANNOT BE USED FOR PLANE STRESS
C -------------------------------------------------------------------
C    PROPS(1) - E
C    PROPS(2) - NU
C -------------------------------------------------------------------
C
      IF (NDI.NE.3) THEN
         WRITE (7, *) 'THIS UMAT MAY ONLY BE USED FOR ELEMENTS
     1   WITH THREE DIRECT STRESS COMPONENTS'
         CALL XIT
      ENDIF
C
C    ELASTIC PROPERTIES
      EMOD=PROPS(1)
      ENU=PROPS(2)
      EBULK3=EMOD/(ONE-TWO*ENU)
      EG2=EMOD/(ONE+ENU)
      EG=EG2/TWO
      EG3=THREE*EG
      ELAM=(EBULK3-EG2)/THREE
```

EMOD = E

ENU = ν

EBULK3 = $\frac{E}{1-2\nu}$

EG2 = $\frac{E}{1+\nu}$ ELAM = $\frac{1}{3}\left[\frac{E}{1-2\nu} - \frac{E}{1+\nu}\right]$

EG3 = $\frac{3E}{1+\nu}$

EG = $\frac{E}{2(1+\nu)}$

```
C
C     ELASTIC STIFFNESS
C
      DO K1=1, NDI
        DO K2=1, NDI
          DDSDDE(K2, K1)=ELAM
        END DO
        DDSDDE(K1, K1)=EG2+ELAM
      END DO
      DO K1=NDI+1, NTENS
        DDSDDE(K1 ,K1)=EG
      END DO
C
C     CALCULATE STRESS
C
      DO K1=1, NTENS
        DO K2=1, NTENS
          STRESS(K2)=STRESS(K2)+DDSDDE(K2, K1)*DSTRAN(K1)
        END DO
      END DO
C
      RETURN
      END
```

Remarks

- This very simple **`UMAT`** yields exactly the same results as the ABAQUS *ELASTIC option.
  - This is true even for large-strain calculations: all necessary large-strain contributions are generated by ABAQUS.
- The routine can be used with and without the *ORIENTATION option.
- It is usually straightforward to write a single routine that handles (generalized) plane strain, axisymmetric, and three-dimensional geometries.
  - Generally, plane stress must be treated as a separate case because the stiffness coefficients are different.
- The routine is written in incremental form as a preparation for subsequent elastic-plastic examples.

- Even for linear analysis, **UMAT** is called twice for the first iteration of each increment: once for assembly and once for recovery. Subsequently, it is called once per iteration: assembly and recovery are combined.
- A check is performed on the number of direct stress components, and the analysis is terminated by calling the subroutine, **XIT**.
  - A message is written to the message file (unit=7).

Example 2: Non-Isothermal Elasticity

Governing Equations

- Non-isothermal elasticity equation:

$$\sigma_{ij} = \lambda(T)\delta_{ij}\varepsilon^{el}_{kk} + 2\mu(T)\varepsilon^{el}_{ij}, \qquad \varepsilon^{el}_{ij} = \varepsilon_{ij} - \alpha T\delta_{ij},$$

  or in a Jaumann (corotational) rate form:

$$\dot{\sigma}^{J}_{ij} = \lambda\delta_{ij}\dot{\varepsilon}^{el}_{kk} + 2\mu\dot{\varepsilon}^{el}_{ij} + \dot{\lambda}\delta_{ij}\varepsilon^{el}_{kk} + 2\dot{\mu}\varepsilon^{el}_{ij}, \qquad \dot{\varepsilon}^{el}_{ij} = \dot{\varepsilon}_{ij} - \alpha\dot{T}\delta_{ij}.$$

- The Jaumann rate equation is integrated in a corotational framework:

$$\Delta\sigma^{J}_{ij} = \lambda\delta_{ij}\Delta\varepsilon^{el}_{kk} + 2\mu\Delta\varepsilon^{el}_{ij} + \Delta\lambda\delta_{ij}\varepsilon^{el}_{kk} + 2\Delta\mu\varepsilon^{el}_{ij}, \quad \Delta\varepsilon^{el}_{ij} = \Delta\varepsilon_{ij} - \alpha\Delta T\delta_{ij}.$$

The appropriate coding is shown on the following pages.

Coding for Non-Isothermal Elasticity

```
C    LOCAL ARRAYS
C ------------------------------------------------------------------
C    EELAS  - ELASTIC STRAINS
C    ETHERM - THERMAL STRAINS
C    DTHERM - INCREMENTAL THERMAL STRAINS
C    DELDSE - CHANGE IN STIFFNESS DUE TO TEMPERATURE CHANGE
C ------------------------------------------------------------------
      DIMENSION EELAS(6), ETHERM(6), DTHERM(6), DELDSE(6,6)
C
      PARAMETER(ZERO=0.D0, ONE=1.D0, TWO=2.D0, THREE=3.D0, SIX=6.D0)
C ------------------------------------------------------------------
C    UMAT FOR ISOTROPIC THERMO-ELASTICITY WITH LINEARLY VARYING
C    MODULI - CANNOT BE USED FOR PLANE STRESS
C ------------------------------------------------------------------
C    PROPS(1) - E(T0)
C    PROPS(2) - NU(T0)
C    PROPS(3) - T0
C    PROPS(4) - E(T1)
C    PROPS(5) - NU(T1)
C    PROPS(6) - T1
C    PROPS(7) - ALPHA
C    PROPS(8) - T_INITIAL
```

```
C     ELASTIC PROPERTIES AT START OF INCREMENT
C
      FAC1=(TEMP-PROPS(3))/(PROPS(6)-PROPS(3))
      IF (FAC1 .LT. ZERO) FAC1=ZERO
      IF (FAC1 .GT. ONE) FAC1=ONE
      FAC0=ONE-FAC1
      EMOD=FAC0*PROPS(1)+FAC1*PROPS(4)
      ENU=FAC0*PROPS(2)+FAC1*PROPS(5)
      EBULK3=EMOD/(ONE-TWO*ENU)
      EG20=EMOD/(ONE+ENU)
      EG0=EG20/TWO
      ELAM0=(EBULK3-EG20)/THREE
C
C     ELASTIC PROPERTIES AT END OF INCREMENT
C
      FAC1=(TEMP+DTEMP-PROPS(3))/(PROPS(6)-PROPS(3))
      IF (FAC1 .LT. ZERO) FAC1=ZERO
      IF (FAC1 .GT. ONE) FAC1=ONE
      FAC0=ONE-FAC1
      EMOD=FAC0*PROPS(1)+FAC1*PROPS(4)
      ENU=FAC0*PROPS(2)+FAC1*PROPS(5)
      EBULK3=EMOD/(ONE-TWO*ENU)
      EG2=EMOD/(ONE+ENU)
      EG=EG2/TWO
      ELAM=(EBULK3-EG2)/THREE
```

```
C
C     ELASTIC STIFFNESS AT END OF INCREMENT AND STIFFNESS CHANGE
C
       DO K1=1,NDI
         DO K2=1,NDI
           DDSDDE(K2,K1)=ELAM
           DELDSE(K2,K1)=ELAM-ELAM0
         END DO
         DDSDDE(K1,K1)=EG2+ELAM
         DELDSE(K1,K1)=EG2+ELAM-EG20-ELAM0
       END DO
       DO K1=NDI+1,NTENS
         DDSDDE(K1,K1)=EG
         DELDSE(K1,K1)=EG-EG0
       END DO
C
C     CALCULATE THERMAL EXPANSION
C
       DO K1=1,NDI
         ETHERM(K1)=PROPS(7)*(TEMP-PROPS(8))
         DTHERM(K1)=PROPS(7)*DTEMP
       END DO
```

```
      DO K1=NDI+1,NTENS
        ETHERM(K1)=ZERO
        DTHERM(K1)=ZERO
      END DO
C
C     CALCULATE STRESS, ELASTIC STRAIN AND THERMAL STRAIN
C
      DO K1=1, NTENS
        DO K2=1, NTENS
          STRESS(K2)=STRESS(K2)+DDSDDE(K2,K1)*(DSTRAN(K1)-DTHERM(K1))
     1                         +DELDSE(K2,K1)*( STRAN(K1)-ETHERM(K1))
        END DO
        ETHERM(K1)=ETHERM(K1)+DTHERM(K1)
        EELAS(K1)=STRAN(K1)+DSTRAN(K1)-ETHERM(K1)
      END DO
C
C     STORE ELASTIC AND THERMAL STRAINS IN STATE VARIABLE ARRAY
C
      DO K1=1, NTENS
        STATEV(K1)=EELAS(K1)
        STATEV(K1+NTENS)=ETHERM(K1)
      END DO
      RETURN
      END
```

Remarks

- This **UMAT** yields exactly the same results as the *ELASTIC option with temperature dependence.
- The routine is written in incremental form, which allows generalization to more complex temperature dependence.

Example 3: Neo-Hookean Hyperelasticity

Governing Equations

- The *ELASTIC option does not work well for finite elastic strains because a proper finite-strain energy function is not defined.
- Hence, we define a proper strain energy density function:

$$U = U(I_1, I_2, J) = C_{10}(I_1 - 3) + \frac{1}{D_1}(J-1)^2 \ .$$

  - Here I_1, I_2, and J are the three strain invariants, expressed in terms of the left Cauchy-Green tensor, $\underline{B}$:

$$I_1 = tr(\underline{B}), \quad I_2 = \frac{1}{2}(I_1^2 - tr(\underline{B} \cdot \underline{B})), \quad \underline{B} = \underline{F} \cdot \underline{F}^T, \quad J = det(\underline{F}) .$$

- In actuality, we use the deviatoric invariants $\bar{I}_1$ and $\bar{I}_2$ (see Section 4.6.1 of the ABAQUS Theory Manual for more information).
  - The constitutive equation can be written directly in terms of the deformation gradient:

$$\sigma_{ij} = \frac{2}{J} C_{10}\left(\bar{B}_{ij} - \frac{1}{3}\delta_{ij}\bar{B}_{kk}\right) + \frac{2}{D_1}(J-1)\delta_{ij}, \quad \bar{B}_{ij} = B_{ij}/J^{2/3}.$$

- We define the virtual rate of deformation as

$$\delta D_{ij} = \frac{1}{2}(\delta F_{im} F_{mj}^{-1} + F_{mi}^{-1}\delta F_{jm}).$$

- The Kirchhoff stress is defined through

$$\tau_{ij} = J\sigma_{ij}.$$

- The material Jacobian derives from the variation in Kirchhoff stress:

$$\delta\tau_{ij} = JC_{ijkl}\delta D_{kl} \quad ,$$

where C_{ijkl} are the components of the Jacobian. Using the Neo-Hookean model,

$$C_{ijkl} = \frac{2}{J}C_{10}\left(\frac{1}{2}(\delta_{ik}\bar{B}_{jl} + \bar{B}_{ik}\delta_{jl} + \delta_{il}\bar{B}_{jk} + \bar{B}_{il}\delta_{jk})\right.$$

$$\left. -\frac{2}{3}\delta_{ij}\bar{B}_{kl} - \frac{2}{3}\bar{B}_{ij}\delta_{kl} + \frac{2}{9}\delta_{ij}\delta_{kl}\bar{B}_{mm}\right) + \frac{2}{D_1}(2J-1)\delta_{ij}\delta_{kl} \quad .$$

  - The expression is fairly complex, but it is straightforward to implement.
  - For details of the derivation see Section 4.6.1 of the ABAQUS Theory Manual.

The appropriate coding is shown on the following pages.

Coding for Neo-Hookean Hyperelasticity

```
C    LOCAL ARRAYS
C ----------------------------------------------------------------
C    EELAS  - LOGARITHMIC ELASTIC STRAINS
C    EELASP - PRINCIPAL ELASTIC STRAINS
C    BBAR   - DEVIATORIC RIGHT CAUCHY-GREEN TENSOR
C    BBARP  - PRINCIPAL VALUES OF BBAR
C    BBARN  - PRINCIPAL DIRECTION OF BBAR (AND EELAS)
C    DISTGR - DEVIATORIC DEFORMATION GRADIENT (DISTORTION TENSOR)
C ----------------------------------------------------------------
C
      DIMENSION EELAS(6), EELASP(3), BBAR(6), BBARP(3), BBARN(3, 3),
     1          DISTGR(3,3)
C
      PARAMETER(ZERO=0.D0, ONE=1.D0, TWO=2.D0, THREE=3.D0, FOUR=4.D0,
     1          SIX=6.D0)
C
C ----------------------------------------------------------------
C    UMAT FOR COMPRESSIBLE NEO-HOOKEAN HYPERELASTICITY
C    CANNOT BE USED FOR PLANE STRESS
C ----------------------------------------------------------------
C    PROPS(1) - E
C    PROPS(2) - NU
```

```
C ------------------------------------------------------------------
C
C    ELASTIC PROPERTIES
C
      EMOD=PROPS(1)
      ENU=PROPS(2)
      C10=EMOD/(FOUR*(ONE+ENU))
      D1=SIX*(ONE-TWO*ENU)/EMOD
C
C    JACOBIAN AND DISTORTION TENSOR
C
      DET=DFGRD1(1, 1)*DFGRD1(2, 2)*DFGRD1(3, 3)
     1   -DFGRD1(1, 2)*DFGRD1(2, 1)*DFGRD1(3, 3)
      IF(NSHR.EQ.3) THEN
        DET=DET+DFGRD1(1, 2)*DFGRD1(2, 3)*DFGRD1(3, 1)
     1         +DFGRD1(1, 3)*DFGRD1(3, 2)*DFGRD1(2, 1)
     2         -DFGRD1(1, 3)*DFGRD1(3,1)*DFGRD1(2, 2)
     3         -DFGRD1(2, 3)*DFGRD1(3, 2)*DFGRD1(1, 1)
      END IF
      SCALE=DET**(-ONE/THREE)
      DO K1=1, 3
        DO K2=1, 3
          DISTGR(K2, K1)=SCALE*DFGRD1(K2, K1)
        END DO
```

```
      END DO
C     CALCULATE DEVIATORIC LEFT CAUCHY-GREEN DEFORMATION TENSOR
C
      BBAR(1)=DISTGR(1, 1)**2+DISTGR(1, 2)**2+DISTGR(1, 3)**2
      BBAR(2)=DISTGR(2, 1)**2+DISTGR(2, 2)**2+DISTGR(2, 3)**2
      BBAR(3)=DISTGR(3, 3)**2+DISTGR(3, 1)**2+DISTGR(3, 2)**2
      BBAR(4)=DISTGR(1, 1)*DISTGR(2, 1)+DISTGR(1, 2)*DISTGR(2, 2)
     1       +DISTGR(1, 3)*DISTGR(2, 3)
      IF(NSHR.EQ.3) THEN
        BBAR(5)=DISTGR(1, 1)*DISTGR(3, 1)+DISTGR(1, 2)*DISTGR(3, 2)
     1         +DISTGR(1, 3)*DISTGR(3, 3)
        BBAR(6)=DISTGR(2, 1)*DISTGR(3, 1)+DISTGR(2, 2)*DISTGR(3, 2)
     1         +DISTGR(2, 3)*DISTGR(3, 3)
      END IF
C
C     CALCULATE THE STRESS
C
      TRBBAR=(BBAR(1)+BBAR(2)+BBAR(3))/THREE
      EG=TWO*C10/DET
      EK=TWO/D1*(TWO*DET-ONE)
      PR=TWO/D1*(DET-ONE)
      DO K1=1,NDI
        STRESS(K1)=EG*(BBAR(K1)-TRBBAR)+PR
      END DO
```

```
      DO K1=NDI+1,NDI+NSHR
        STRESS(K1)=EG*BBAR(K1)
      END DO
C     CALCULATE THE STIFFNESS
C
      EG23=EG*TWO/THREE
      DDSDDE(1, 1)= EG23*(BBAR(1)+TRBBAR)+EK
      DDSDDE(2, 2)= EG23*(BBAR(2)+TRBBAR)+EK
      DDSDDE(3, 3)= EG23*(BBAR(3)+TRBBAR)+EK
      DDSDDE(1, 2)=-EG23*(BBAR(1)+BBAR(2)-TRBBAR)+EK
      DDSDDE(1, 3)=-EG23*(BBAR(1)+BBAR(3)-TRBBAR)+EK
      DDSDDE(2, 3)=-EG23*(BBAR(2)+BBAR(3)-TRBBAR)+EK
      DDSDDE(1, 4)= EG23*BBAR(4)/TWO
      DDSDDE(2, 4)= EG23*BBAR(4)/TWO
      DDSDDE(3, 4)=-EG23*BBAR(4)
      DDSDDE(4, 4)= EG*(BBAR(1)+BBAR(2))/TWO
      IF(NSHR.EQ.3) THEN
        DDSDDE(1, 5)= EG23*BBAR(5)/TWO
        DDSDDE(2, 5)=-EG23*BBAR(5)
        DDSDDE(3, 5)= EG23*BBAR(5)/TWO
        DDSDDE(1, 6)=-EG23*BBAR(6)
        DDSDDE(2, 6)= EG23*BBAR(6)/TWO
        DDSDDE(3, 6)= EG23*BBAR(6)/TWO
        DDSDDE(5, 5)= EG*(BBAR(1)+BBAR(3))/TWO
```

```
          DDSDDE(6, 6)= EG*(BBAR(2)+BBAR(3))/TWO
          DDSDDE(4,5)= EG*BBAR(6)/TWO
          DDSDDE(4,6)= EG*BBAR(5)/TWO
          DDSDDE(5,6)= EG*BBAR(4)/TWO
        END IF
        DO K1=1, NTENS
          DO K2=1, K1-1
            DDSDDE(K1, K2)=DDSDDE(K2, K1)
          END DO
        END DO
C
C     CALCULATE LOGARITHMIC ELASTIC STRAINS (OPTIONAL)
C
        CALL SPRIND(BBAR, BBARP, BBARN, 1, NDI, NSHR)
        EELASP(1)=LOG(SQRT(BBARP(1))/SCALE)
        EELASP(2)=LOG(SQRT(BBARP(2))/SCALE)
        EELASP(3)=LOG(SQRT(BBARP(3))/SCALE)
        EELAS(1)=EELASP(1)*BBARN(1,1)**2+EELASP(2)*BBARN(2, 1)**2
       1         +EELASP(3)*BBARN(3, 1)**2
        EELAS(2)=EELASP(1)*BBARN(1, 2)**2+EELASP(2)*BBARN(2, 2)**2
       1         +EELASP(3)*BBARN(3, 2)**2
        EELAS(3)=EELASP(1)*BBARN(1, 3)**2+EELASP(2)*BBARN(2, 3)**2
       1         +EELASP(3)*BBARN(3, 3)**2
        EELAS(4)=TWO*(EELASP(1)*BBARN(1, 1)*BBARN(1, 2)
```

Call to SPRIND

```
     1             +EELASP(2)*BBARN(2, 1)*BBARN(2, 2)
     2             +EELASP(3)*BBARN(3, 1)*BBARN(3, 2))
      IF(NSHR.EQ.3) THEN
        EELAS(5)=TWO*(EELASP(1)*BBARN(1, 1)*BBARN(1, 3)
     1               +EELASP(2)*BBARN(2, 1)*BBARN(2, 3)
     2               +EELASP(3)*BBARN(3, 1)*BBARN(3, 3))
        EELAS(6)=TWO*(EELASP(1)*BBARN(1, 2)*BBARN(1, 3)
     1               +EELASP(2)*BBARN(2, 2)*BBARN(2, 3)
     2               +EELASP(3)*BBARN(3, 2)*BBARN(3, 3))
      END IF
C
C     STORE ELASTIC STRAINS IN STATE VARIABLE ARRAY
C
      DO K1=1, NTENS
        STATEV(K1)=EELAS(K1)
      END DO
C
      RETURN
      END
```

Remarks

- This **UMAT** yields exactly the same results as the *HYPERELASTIC option with $N = 1$ and $C_{01} = 0$.
- Note the use of the utility **SPRIND**.

```
CALL SPRIND(BBAR, BBARP, BBARN, 1, NDI, NSHR)
```

  - Tensor **BBAR** consists of **NDI** direct components and **NSHR** shear components.
  - **SPRIND** returns the principal values and direction cosines of the principal directions of **BBAR** in **BBARP** and **BBARN**, respectively.
  - A value of 1 is used as the fourth argument to indicate that **BBAR** contains stresses. (A value of 2 is used for strains.)
- Hyperelastic materials are often implemented more easily in user subroutine **UHYPER**.

Example 4: Kinematic Hardening Plasticity

Governing Equations

- Elasticity:

$$\sigma_{ij} = \lambda\delta_{ij}\varepsilon^{el}_{kk} + 2\mu\varepsilon^{el}_{ij},$$

or in a Jaumann (corotational) rate form:

$$\dot{\sigma}^{J}_{ij} = \lambda\delta_{ij}\dot{\varepsilon}^{el}_{kk} + 2\mu\dot{\varepsilon}^{el}_{ij}.$$

  - The Jaumann rate equation is integrated in a corotational framework:

$$\Delta\sigma^{J}_{ij} = \lambda\delta_{ij}\Delta\varepsilon^{el}_{kk} + 2\mu\Delta\varepsilon^{el}_{ij}.$$

- Plasticity:
  - Yield function:

$$\sqrt{\frac{3}{2}(S_{ij}-\alpha_{ij})(S_{ij}-\alpha_{ij})}-\sigma_y = 0 \ .$$

  - Equivalent plastic strain rate:

$$\dot{\bar{\varepsilon}}^{pl} = \sqrt{\frac{2}{3}\dot{\varepsilon}_{ij}^{pl}\dot{\varepsilon}_{ij}^{pl}}.$$

  - Plastic flow law:

$$\dot{\varepsilon}_{ij}^{pl} = \frac{3}{2}(S_{ij}-\alpha_{ij})\dot{\bar{\varepsilon}}^{pl}/\sigma_y \ .$$

  - Prager-Ziegler (linear) kinematic hardening:

$$\dot{\alpha}_{ij} = \frac{2}{3}h\dot{\varepsilon}_{ij}^{pl}.$$

Integration Procedure

- We first calculate the equivalent stress based on purely elastic behavior (elastic predictor):

$$\bar{\sigma}^{pr} = \sqrt{\frac{3}{2}(S_{ij}^{pr} - \alpha_{ij}^{o})(S_{ij}^{pr} - \alpha_{ij}^{o})}\ , \qquad S_{ij}^{pr} = S_{ij}^{o} + 2\mu\Delta e_{ij}.$$

- Plastic flow occurs if the elastic predictor is larger than the yield stress. The backward Euler method is used to integrate the equations:

$$\Delta\varepsilon_{ij}^{pl} = \frac{3}{2}(S_{ij}^{pr} - \alpha_{ij}^{o})\Delta\bar{\varepsilon}^{pl}/\bar{\sigma}^{pr}.$$

- After some manipulation we obtain a closed form expression for the equivalent plastic strain increment:

$$\Delta\bar{\varepsilon}^{pl} = (\bar{\sigma}^{pr} - \sigma_y)/(h + 3\mu)\ .$$

- This leads to the following update equations for the shift tensor, the stress, and the plastic strain:

$$\Delta\alpha_{ij} = \eta_{ij} h \Delta\bar{\varepsilon}^{pl}, \quad \Delta\varepsilon_{ij}^{pl} = \frac{3}{2}\eta_{ij}\Delta\bar{\varepsilon}^{pl}$$

$$\sigma_{ij} = \alpha_{ij}^{o} + \Delta\alpha_{ij} + \eta_{ij}\sigma_y + \frac{1}{3}\delta_{ij}\sigma_{kk}^{pr}, \quad \eta_{ij} = (S_{ij}^{pr} - \alpha_{ij}^{o})/\bar{\sigma}^{pr}.$$

- In addition, you can readily obtain the consistent Jacobian:

$$\Delta\dot{\sigma}_{ij} = \lambda^{*}\delta_{ij}\Delta\dot{\varepsilon}_{kk} + 2\mu^{*}\Delta\dot{\varepsilon}_{ij} + \left(\frac{h}{1 + h/3\mu} - 3\mu^{*}\right)\eta_{ij}\eta_{kl}\Delta\dot{\varepsilon}_{kl}$$

$$\mu^{*} = \mu(\sigma_y + h\Delta\bar{\varepsilon}^{pl})/\bar{\sigma}^{pr}, \quad \lambda^{*} = k - \frac{2}{3}\mu^{*}.$$

  – The integration procedure for kinematic hardening is described in Section 21 of the ABAQUS/Explicit User's Manual.

The appropriate coding is shown on the following pages.

Coding for Kinematic Hardening Plasticity

```
C    LOCAL ARRAYS
C ------------------------------------------------------------------
C    EELAS  - ELASTIC STRAINS
C    EPLAS  - PLASTIC STRAINS
C    ALPHA  - SHIFT TENSOR
C    FLOW   - PLASTIC FLOW DIRECTIONS
C    OLDS   - STRESS AT START OF INCREMENT
C    OLDPL  - PLASTIC STRAINS AT START OF INCREMENT
C
      DIMENSION EELAS(6), EPLAS(6), ALPHA(6), FLOW(6), OLDS(6), OLDPL(6)
C
      PARAMETER(ZERO=0.D0, ONE=1.D0, TWO=2.D0, THREE=3.D0, SIX=6.D0,
     1          ENUMAX=.4999D0, TOLER=1.0D-6)
C
C ------------------------------------------------------------------
C    UMAT FOR ISOTROPIC ELASTICITY AND MISES PLASTICITY
C    WITH KINEMATIC HARDENING - CANNOT BE USED FOR PLANE STRESS
C ------------------------------------------------------------------
C    PROPS(1) - E
C    PROPS(2) - NU
C    PROPS(3) - SYIELD
C    PROPS(4) - HARD
```

```
C -------------------------------------------------------------------
C
C    ELASTIC PROPERTIES
C
      EMOD=PROPS(1)
      ENU=MIN(PROPS(2), ENUMAX)
      EBULK3=EMOD/(ONE-TWO*ENU)
      EG2=EMOD/(ONE+ENU)
      EG=EG2/TWO
      EG3=THREE*EG
      ELAM=(EBULK3-EG2)/THREE
C
C    ELASTIC STIFFNESS
C

      DO K1=1, NDI
        DO K2=1, NDI
          DDSDDE(K2, K1)=ELAM
        END DO
        DDSDDE(K1, K1)=EG2+ELAM
      END DO
      DO K1=NDI+1, NTENS
        DDSDDE(K1, K1)=EG
      END DO
```

```
C
C    RECOVER ELASTIC STRAIN, PLASTIC STRAIN AND SHIFT TENSOR AND ROTATE
C    NOTE: USE CODE 1 FOR (TENSOR) STRESS, CODE 2 FOR (ENGINEERING) STRAIN
C
      CALL ROTSIG(STATEV(        1), DROT, EELAS, 2, NDI, NSHR)
      CALL ROTSIG(STATEV(  NTENS+1), DROT, EPLAS, 2, NDI, NSHR)
      CALL ROTSIG(STATEV(2*NTENS+1), DROT, ALPHA, 1, NDI, NSHR)

C
C    SAVE STRESS AND PLASTIC STRAINS AND
C    CALCULATE PREDICTOR STRESS AND ELASTIC STRAIN
C
      DO K1=1, NTENS
        OLDS(K1)=STRESS(K1)
        OLDPL(K1)=EPLAS(K1)
        EELAS(K1)=EELAS(K1)+DSTRAN(K1)
        DO K2=1, NTENS
          STRESS(K2)=STRESS(K2)+DDSDDE(K2, K1)*DSTRAN(K1)
        END DO
      END DO
```

Calls to ROTSIG

```
C
C       CALCULATE EQUIVALENT VON MISES STRESS
C
       SMISES=(STRESS(1)-ALPHA(1)-STRESS(2)+ALPHA(2))**2
      1       +(STRESS(2)-ALPHA(2)-STRESS(3)+ALPHA(3))**2
      2       +(STRESS(3)-ALPHA(3)-STRESS(1)+ALPHA(1))**2
       DO K1=NDI+1,NTENS
         SMISES=SMISES+SIX*(STRESS(K1)-ALPHA(K1))**2
       END DO
       SMISES=SQRT(SMISES/TWO)
C
C     GET YIELD STRESS AND HARDENING MODULUS
C
       SYIELD=PROPS(3)
       HARD=PROPS(4)
C
C     DETERMINE IF ACTIVELY YIELDING
C
       IF(SMISES.GT.(ONE+TOLER)*SYIELD) THEN
C
C       ACTIVELY YIELDING
```

```
C       SEPARATE THE HYDROSTATIC FROM THE DEVIATORIC STRESS
C       CALCULATE THE FLOW DIRECTION
C
         SHYDRO=(STRESS(1)+STRESS(2)+STRESS(3))/THREE
         DO K1=1,NDI
           FLOW(K1)=(STRESS(K1)-ALPHA(K1)-SHYDRO)/SMISES
         END DO
         DO K1=NDI+1,NTENS
           FLOW(K1)=(STRESS(K1)-ALPHA(K1))/SMISES
         END DO
C
C       SOLVE FOR EQUIVALENT PLASTIC STRAIN INCREMENT
C
         DEQPL=(SMISES-SYIELD)/(EG3+HARD)
```

```
C
C       UPDATE SHIFT TENSOR, ELASTIC AND PLASTIC STRAINS AND STRESS
C
         DO K1=1,NDI
           ALPHA(K1)=ALPHA(K1)+HARD*FLOW(K1)*DEQPL
           EPLAS(K1)=EPLAS(K1)+THREE/TWO*FLOW(K1)*DEQPL
           EELAS(K1)=EELAS(K1)-THREE/TWO*FLOW(K1)*DEQPL
           STRESS(K1)=ALPHA(K1)+FLOW(K1)*SYIELD+SHYDRO
         END DO
         DO K1=NDI+1,NTENS
           ALPHA(K1)=ALPHA(K1)+HARD*FLOW(K1)*DEQPL
           EPLAS(K1)=EPLAS(K1)+THREE*FLOW(K1)*DEQPL
           EELAS(K1)=EELAS(K1)-THREE*FLOW(K1)*DEQPL
           STRESS(K1)=ALPHA(K1)+FLOW(K1)*SYIELD
         END DO
C       CALCULATE PLASTIC DISSIPATION
C
         SPD=ZERO
         DO K1=1,NTENS
           SPD=SPD+(STRESS(K1)+OLDS(K1))*(EPLAS(K1)-OLDPL(K1))/TWO
         END DO
```

```
C
C       FORMULATE THE JACOBIAN (MATERIAL TANGENT)
C       FIRST CALCULATE EFFECTIVE MODULI
C
         EFFG=EG*(SYIELD+HARD*DEQPL)/SMISES
         EFFG2=TWO*EFFG
         EFFG3=THREE*EFFG
         EFFLAM=(EBULK3-EFFG2)/THREE
         EFFHRD=EG3*HARD/(EG3+HARD)-EFFG3
         DO K1=1, NDI
           DO K2=1, NDI
             DDSDDE(K2, K1)=EFFLAM
           END DO
           DDSDDE(K1, K1)=EFFG2+EFFLAM
         END DO
         DO K1=NDI+1, NTENS
           DDSDDE(K1, K1)=EFFG
         END DO
         DO K1=1, NTENS
           DO K2=1, NTENS
             DDSDDE(K2, K1)=DDSDDE(K2, K1)+EFFHRD*FLOW(K2)*FLOW(K1)
           END DO
         END DO
       ENDIF
```

```
C
C     STORE ELASTIC STRAINS, PLASTIC STRAINS AND SHIFT TENSOR
C     IN STATE VARIABLE ARRAY
C
       DO K1=1,NTENS
         STATEV(K1)=EELAS(K1)
         STATEV(K1+NTENS)=EPLAS(K1)
         STATEV(K1+2*NTENS)=ALPHA(K1)
       END DO
C
       RETURN
       END
```

Remarks

- This **`UMAT`** yields exactly the same results as the *PLASTIC option with KINEMATIC hardening.
  - This is also true for large-strain calculations. The necessary rotations of stress and strain are taken care of by ABAQUS.

- Rotation of the shift tensor and the elastic and plastic strains is accomplished by the calls to **ROTSIG**. The call

```
CALL ROTSIG(STATEV(1), DROT, EELAS, 2, NDI, NSHR)
```

  applies the incremental rotation, **DROT**, to **STATEV** and stores the result in **ELAS**.

  - **STATEV** consists of **NDI** direct components and **NSHR** shear components.
  - A value of 1 is used as the fourth argument to indicate that the transformed array contains tensor shear components such as α_{ij}. A value of 2 indicates that the array contains engineering shear components, such as ε_{ij}^{pl}.

- The rotation should be applied prior to the integration procedure.

- The routine is written for linear hardening because the classical Prager-Ziegler theory is limited to this case.
  - More complex nonlinear kinematic hardening models are much more difficult to integrate.
  - However, once a suitable integration procedure is obtained, the implementation in **`UMAT`** is straightforward and follows the examples discussed here.

Example 5: Isotropic Hardening Plasticity

Governing Equations

- Elasticity:

$$\sigma_{ij} = \lambda\delta_{ij}\varepsilon^{el}_{kk} + 2\mu\varepsilon^{el}_{ij},$$

or in a Jaumann (corotational) rate form:

$$\dot{\sigma}^{J}_{ij} = \lambda\delta_{ij}\dot{\varepsilon}^{el}_{kk} + 2\mu\dot{\varepsilon}^{el}_{ij}.$$

  - The Jaumann rate equation is integrated in a corotational framework:

$$\Delta\sigma^{J}_{ij} = \lambda\delta_{ij}\Delta\varepsilon^{el}_{kk} + 2\mu\Delta\varepsilon^{el}_{ij}.$$

- Plasticity:
  - Yield function:

$$\sqrt{\frac{3}{2}S_{ij}S_{ij}} - \sigma_y(\bar{\varepsilon}^{pl}) = 0\,, \qquad S_{ij} = \sigma_{ij} - \frac{1}{3}\delta_{ij}\sigma_{kk}\,.$$

  - Equivalent plastic strain:

$$\bar{\varepsilon}^{pl} = \int_0^t \dot{\bar{\varepsilon}}^{pl}\,dt, \qquad \dot{\bar{\varepsilon}}^{pl} = \sqrt{\frac{2}{3}\dot{\varepsilon}_{ij}^{pl}\dot{\varepsilon}_{ij}^{pl}}.$$

  - Plastic flow law:

$$\dot{\varepsilon}_{ij}^{pl} = \frac{3}{2}\frac{S_{ij}}{\sigma_y}\dot{\bar{\varepsilon}}^{pl}.$$

Integration Procedure

- We first calculate the von Mises stress based on purely elastic behavior (elastic predictor):

$$\bar{\sigma}^{pr} = \sqrt{\frac{3}{2} S_{ij}^{pr} S_{ij}^{pr}}, \qquad S_{ij}^{pr} = S_{ij}^{o} + 2\mu \Delta e_{ij}.$$

- If the elastic predictor is larger than the current yield stress, plastic flow occurs. The backward Euler method is used to integrate the equations.
  - After some manipulation we can reduce the problem to a single equation in terms of the incremental equivalent plastic strain:

$$\bar{\sigma}^{pr} - 3\mu \Delta \bar{\varepsilon}^{pl} = \sigma_y(\bar{\varepsilon}^{pl}).$$

  - This equation is solved with Newton's method.

- After the equation is solved, the following update equations for the stress and the plastic strain can be used:

$$\sigma_{ij} = \eta_{ij}\sigma_y + \frac{1}{3}\delta_{ij}\sigma_{kk}^{pr}, \qquad \Delta\varepsilon_{ij}^{pl} = \frac{3}{2}\eta_{ij}\Delta\bar{\varepsilon}^{pl}$$

$$\eta_{ij} = S_{ij}^{pr}/\bar{\sigma}^{pr}.$$

- In addition, you can readily obtain the consistent Jacobian:

$$\Delta\dot{\sigma}_{ij} = \lambda^*\delta_{ij}\Delta\dot{\varepsilon}_{kk} + 2\mu^*\Delta\dot{\varepsilon}_{ij} + \left(\frac{h}{1+h/3\mu} - 3\mu^*\right)\eta_{ij}\eta_{kl}\Delta\dot{\varepsilon}_{kl}$$

$$\mu^* = \mu\sigma_y/\bar{\sigma}^{pr}, \quad \lambda^* = k - \frac{2}{3}\mu^*, \quad h = d\sigma_y/d\bar{\varepsilon}^{pl}.$$

  - A detailed discussion about the isotropic plasticity integration algorithm can be found in Section 4.2.2 of the ABAQUS Theory Manual.

The appropriate coding is shown on the following pages.

Coding for Isotropic Mises Plasticity

```
C    LOCAL ARRAYS
C ----------------------------------------------------------------
C    EELAS  - ELASTIC STRAINS
C    EPLAS  - PLASTIC STRAINS
C    FLOW   - DIRECTION OF PLASTIC FLOW
C ----------------------------------------------------------------
C
      DIMENSION EELAS(6),EPLAS(6),FLOW(6), HARD(3)
C
      PARAMETER(ZERO=0.D0, ONE=1.D0, TWO=2.D0, THREE=3.D0, SIX=6.D0,
     1          ENUMAX=.4999D0, NEWTON=10, TOLER=1.0D-6)
C
C ----------------------------------------------------------------
C    UMAT FOR ISOTROPIC ELASTICITY AND ISOTROPIC MISES PLASTICITY
C    CANNOT BE USED FOR PLANE STRESS
C ----------------------------------------------------------------
C    PROPS(1) - E
C    PROPS(2) - NU
C    PROPS(3..) - SYIELD AN HARDENING DATA
C    CALLS UHARD FOR CURVE OF YIELD STRESS VS. PLASTIC STRAIN
C ----------------------------------------------------------------
```

```
C
C     ELASTIC PROPERTIES
C
       EMOD=PROPS(1)
       ENU=MIN(PROPS(2), ENUMAX)
       EBULK3=EMOD/(ONE-TWO*ENU)
       EG2=EMOD/(ONE+ENU)
       EG=EG2/TWO
       EG3=THREE*EG
       ELAM=(EBULK3-EG2)/THREE
C
C     ELASTIC STIFFNESS
C

       DO K1=1, NDI
         DO K2=1, NDI
           DDSDDE(K2, K1)=ELAM
         END DO
         DDSDDE(K1, K1)=EG2+ELAM
       END DO
       DO K1=NDI+1, NTENS
         DDSDDE(K1, K1)=EG
       END DO
```

```
C     RECOVER ELASTIC AND PLASTIC STRAINS AND ROTATE FORWARD
C     ALSO RECOVER EQUIVALENT PLASTIC STRAIN
C
      CALL ROTSIG(STATEV(      1), DROT, EELAS, 2, NDI, NSHR)
      CALL ROTSIG(STATEV(NTENS+1), DROT, EPLAS, 2, NDI, NSHR)
      EQPLAS=STATEV(1+2*NTENS)
C
C     CALCULATE PREDICTOR STRESS AND ELASTIC STRAIN
C
      DO K1=1, NTENS
        DO K2=1, NTENS
          STRESS(K2)=STRESS(K2)+DDSDDE(K2, K1)*DSTRAN(K1)
        END DO
        EELAS(K1)=EELAS(K1)+DSTRAN(K1)
      END DO
C
C     CALCULATE EQUIVALENT VON MISES STRESS
C
      SMISES=(STRESS(1)-STRESS(2))**2+(STRESS(2)-STRESS(3))**2
     1                               +(STRESS(3)-STRESS(1))**2
      DO K1=NDI+1,NTENS
        SMISES=SMISES+SIX*STRESS(K1)**2
      END DO
      SMISES=SQRT(SMISES/TWO)
```

```
C
C     GET YIELD STRESS FROM THE SPECIFIED HARDENING CURVE
C
       NVALUE=NPROPS/2-1
       CALL UHARD(SYIEL0, HARD, EQPLAS, EQPLASRT,TIME,DTIME,TEMP,
      1     DTEMP,NOEL,NPT,LAYER,KSPT,KSTEP,KINC,CMNAME,NSTATV,
      2     STATEV,NUMFIELDV,PREDEF,DPRED,NVALUE,PROPS(3))
C
C     DETERMINE IF ACTIVELY YIELDING
C
       IF (SMISES.GT.(ONE+TOLER)*SYIEL0) THEN
C
C       ACTIVELY YIELDING
C       SEPARATE THE HYDROSTATIC FROM THE DEVIATORIC STRESS
C       CALCULATE THE FLOW DIRECTION
C
          SHYDRO=(STRESS(1)+STRESS(2)+STRESS(3))/THREE
          DO K1=1,NDI
            FLOW(K1)=(STRESS(K1)-SHYDRO)/SMISES
          END DO
          DO K1=NDI+1, NTENS
            FLOW(K1)=STRESS(K1)/SMISES
          END DO
```

```
C
C       SOLVE FOR EQUIVALENT VON MISES STRESS
C       AND EQUIVALENT PLASTIC STRAIN INCREMENT USING NEWTON ITERATION
C
         SYIELD=SYIEL0
         DEQPL=ZERO
         DO KEWTON=1, NEWTON
           RHS=SMISES-EG3*DEQPL-SYIELD
           DEQPL=DEQPL+RHS/(EG3+HARD(1))
            CALL UHARD(SYIELD,HARD,EQPLAS+DEQPL,EQPLASRT,TIME,DTIME,TEMP,
     1      DTEMP,NOEL,NPT,LAYER,KSPT,KSTEP,KINC,CMNAME,NSTATV,
     2      STATEV,NUMFIELDV,PREDEF,DPRED,NVALUE,PROPS(3))
           IF(ABS(RHS).LT.TOLER*SYIEL0) GOTO 10
         END DO
C
C       WRITE WARNING MESSAGE TO THE .MSG FILE
C
         WRITE(7,2) NEWTON
    2      FORMAT(//,30X,'***WARNING - PLASTICITY ALGORITHM DID NOT ',
     1                  'CONVERGE AFTER ',I3,' ITERATIONS')
   10    CONTINUE
```

```
C
C       UPDATE STRESS, ELASTIC AND PLASTIC STRAINS AND
C       EQUIVALENT PLASTIC STRAIN
C
         DO K1=1,NDI
           STRESS(K1)=FLOW(K1)*SYIELD+SHYDRO
           EPLAS(K1)=EPLAS(K1)+THREE/TWO*FLOW(K1)*DEQPL
           EELAS(K1)=EELAS(K1)-THREE/TWO*FLOW(K1)*DEQPL
         END DO
         DO K1=NDI+1,NTENS
           STRESS(K1)=FLOW(K1)*SYIELD
           EPLAS(K1)=EPLAS(K1)+THREE*FLOW(K1)*DEQPL
           EELAS(K1)=EELAS(K1)-THREE*FLOW(K1)*DEQPL
         END DO
         EQPLAS=EQPLAS+DEQPL
C
C       CALCULATE PLASTIC DISSIPATION
C
         SPD=DEQPL*(SYIEL0+SYIELD)/TWO
```

```
C
C       FORMULATE THE JACOBIAN (MATERIAL TANGENT)
C       FIRST CALCULATE EFFECTIVE MODULI
C
         EFFG=EG*SYIELD/SMISES
         EFFG2=TWO*EFFG
         EFFG3=THREE/TWO*EFFG2
         EFFLAM=(EBULK3-EFFG2)/THREE
         EFFHRD=EG3*HARD(1)/(EG3+HARD(1))-EFFG3
         DO K1=1, NDI
           DO K2=1, NDI
             DDSDDE(K2, K1)=EFFLAM
           END DO
           DDSDDE(K1, K1)=EFFG2+EFFLAM
         END DO
         DO K1=NDI+1, NTENS
           DDSDDE(K1, K1)=EFFG
         END DO
         DO K1=1, NTENS
           DO K2=1, NTENS
             DDSDDE(K2, K1)=DDSDDE(K2, K1)+EFFHRD*FLOW(K2)*FLOW(K1)
           END DO
         END DO
       ENDIF
```

```
C
C     STORE ELASTIC AND (EQUIVALENT) PLASTIC STRAINS
C     IN STATE VARIABLE ARRAY
C
       DO K1=1, NTENS
         STATEV(K1)=EELAS(K1)
         STATEV(K1+NTENS)=EPLAS(K1)
       END DO
       STATEV(1+2*NTENS)=EQPLAS
C
       RETURN
       END


       SUBROUTINE UHARD(SYIELD,HARD,EQPLAS,EQPLASRT,TIME,DTIME,TEMP,
      1     DTEMP,NOEL,NPT,LAYER,KSPT,KSTEP,KINC,
      2     CMNAME,NSTATV,STATEV,NUMFIELDV,
      3     PREDEF,DPRED,NVALUE,TABLE)

       INCLUDE 'ABA_PARAM.INC'

       CHARACTER*80 CMNAME
       DIMENSION HARD(3),STATEV(NSTATV),TIME(*),
      1          PREDEF(NUMFIELDV),DPRED(*)
```

```
C
      DIMENSION TABLE(2, NVALUE)
C
      PARAMETER(ZERO=0.D0)
C
C    SET YIELD STRESS TO LAST VALUE OF TABLE, HARDENING TO ZERO
C
      SYIELD=TABLE(1, NVALUE)
      HARD(1)=ZERO
C    IF MORE THAN ONE ENTRY, SEARCH TABLE
C
      IF(NVALUE.GT.1) THEN
        DO K1=1, NVALUE-1
          EQPL1=TABLE(2,K1+1)
          IF(EQPLAS.LT.EQPL1) THEN
            EQPL0=TABLE(2, K1)
            IF(EQPL1.LE.EQPL0) THEN
              WRITE(7, 1)
    1         FORMAT(//, 30X, '***ERROR - PLASTIC STRAIN MUST BE ',
     1                        'ENTERED IN ASCENDING ORDER')
              CALL XIT
            ENDIF
```

```
C
C           CURRENT YIELD STRESS AND HARDENING
C
             DEQPL=EQPL1-EQPL0
             SYIEL0=TABLE(1, K1)
             SYIEL1=TABLE(1, K1+1)
             DSYIEL=SYIEL1-SYIEL0
             HARD(1)=DSYIEL/DEQPL
             SYIELD=SYIEL0+(EQPLAS-EQPL0)*HARD(1)
             GOTO 10
           ENDIF
         END DO
   10    CONTINUE
      ENDIF
      RETURN
      END
```

Remarks

- This **UMAT** yields exactly the same results as the *PLASTIC option with ISOTROPIC hardening.
  - This result is also true for large-strain calculations. The necessary rotations of stress and strain are taken care of by ABAQUS.
  - The rotation of elastic and plastic strain, prior to integration, is accomplished by the calls to **ROTSIG**.

- The routine calls user subroutine **UHARD** to recover a piecewise linear hardening curve.
  - It is straightforward to replace the piecewise linear curve by an analytic description.
  - A local Newton iteration is used to determine the current yield stress and hardening modulus.
  - If the data are not given in ascending order of strain, the routine **XIT** is called, which closes all files and terminates execution.

VUMAT Interface

- These input lines act as the interface to a **VUMAT** in which kinematic hardening plasticity is defined.

```
*MATERIAL, NAME=KINPLAS
*USER MATERIAL, CONSTANTS=4
30.E6, 0.3, 30.E3, 40.E3
*DEPVAR
5
*INITIAL CONDITIONS, TYPE=SOLUTION
```

Data line to specify initial solution-dependent variables

- The input lines are identical to those for the **UMAT** interface.
  - The user subroutine must be kept in a separate file, and is invoked with the ABAQUS execution procedure, as follows:

```
abaqus job=... user=....
```

  - The user subroutine must be invoked in a restarted analysis because user subroutines are not saved in the restart file.

- Additional notes:
  - Solution-dependent state variables can be output with identifiers SDV1, SDV2, etc. Contour, path, and *X*–*Y* plots of SDVs can be plotted in ABAQUS/Viewer.
  - Include only a single **`VUMAT`** subroutine in the analysis. If more than one material must be defined, test on the material name in the **`VUMAT`** routine and branch.

- The **VUMAT** subroutine header is shown below:

```
      SUBROUTINE VUMAT(
C Read only -
     1   NBLOCK, NDIR, NSHR, NSTATEV, NFIELDV, NPROPS, LANNEAL,
     2   STEPTIME, TOTALTIME, DT, CMNAME, COORDMP, CHARLENGTH,
     3   PROPS, DENSITY, STRAININC, RELSPININC,
     4   TEMPOLD, STRETCHOLD, DEFGRADOLD, FIELDOLD,
     5   STRESSOLD, STATEOLD, ENERINTERNOLD, ENERINELASOLD,
     6   TEMPNEW, STRETCHNEW, DEFGRADNEW, FIELDNEW,
C Write only -
     7   STRESSNEW, STATENEW, ENERINTERNNEW, ENERINELASNEW)
C
      INCLUDE 'VABA_PARAM.INC'
C
```

```
      DIMENSION PROPS(NPROPS), DENSITY(NBLOCK), COORDMP(NBLOCK),
     1  CHARLENGTH(NBLOCK), STRAININC(NBLOCK, NDIR+NSHR),
     2  RELSPININC(NBLOCK, NSHR), TEMPOLD(NBLOCK),
     3  STRETCHOLD(NBLOCK, NDIR+NSHR),DEFGRADOLD(NBLOCK,NDIR+NSHR+NSHR),
     4  FIELDOLD(NBLOCK, NFIELDV), STRESSOLD(NBLOCK, NDIR+NSHR),
     5  STATEOLD(NBLOCK, NSTATEV), ENERINTERNOLD(NBLOCK),
     6  ENERINELASOLD(NBLOCK), TEMPNEW(NBLOCK),
     7  STRETCHNEW(NBLOCK, NDIR+NSHR),DEFGRADNEW(NBLOCK,NDIR+NSHR+NSHR),
     8  FIELDNEW(NBLOCK, NFIELDV), STRESSNEW(NBLOCK,NDIR+NSHR),
     9  STATENEW(NBLOCK, NSTATEV), ENERINTERNNEW(NBLOCK),
     1  ENERINELASNEW(NBLOCK)
C
      CHARACTER*8 CMNAME
```

VUMAT Variables

- The following quantities are available in **`VUMAT`**, but they cannot be redefined:
  - Stress, stretch, and SDVs at the start of the increment
  - Relative rotation vector and deformation gradient at the start and end of an increment and strain increment
  - Total and incremental values of time, temperature, and user-defined field variables at the start and end of an increment
  - Material constants, density, material point position, and a characteristic element length
  - Internal and dissipated energies at the beginning of the increment
  - Number of material points to be processed in a call to the routine (**`NBLOCK`**)

  - A flag indicating whether the routine is being called during an annealing process

- The following quantities must be defined:
  - Stress and SDVs at the end of an increment
- The following variables may be defined:
  - Internal and dissipated energies at the end of the increment

Many of these variables are equivalent or similar to those in **`UMAT`**.

Complete descriptions of all parameters are provided in the **`VUMAT`** section in Chapter 21 of the ABAQUS/Explicit User's Manual.

Comparison of VUMAT and UMAT Interfaces

There are a number of significant differences between the **UMAT** and **VUMAT** interfaces.

- **VUMAT** uses a two-state architecture: the initial values are in the **OLD** arrays, the new values must be put in the **NEW** arrays.
- The **VUMAT** interface is written to take advantage of vector processing.
- The material Jacobian does not need to be defined.
- No information is provided about element numbers.
- The time increment cannot be redefined.
- Utility routines are not available because they would prevent vectorization.

- The header is usually followed by dimensioning of local arrays. It is good practice to define constants via parameters and to include comments.

```
      PARAMETER( ZERO = 0.D0, ONE = 1.D0, TWO = 2.D0, THREE = 3.D0,
     1   THIRD = 1.D0/3.D0, HALF = .5D0, TWO_THIRDS = 2.D0/3.D0,
     2   THREE_HALFS = 1.5D0 )
C J2 Mises Plasticity with kinematic hardening for plane strain case.
C The state variables are stored as:
C         STATE(*, 1) = back stress component 11
C         STATE(*, 2) = back stress component 22
C         STATE(*, 3) = back stress component 33
C         STATE(*, 4) = back stress component 12
C         STATE(*, 5) = equivalent plastic strain
```

  - The **PARAMETER** assignments yield accurate floating point constant definitions on any platform.

VUMAT Conventions

- Stresses and strains are stored as vectors.
  - For plane stress elements: σ_{11}, σ_{22}, σ_{12}.
  - For plane strain and axisymmetric elements: σ_{11}, σ_{22}, σ_{33}, σ_{12}.
  - For three-dimensional elements: σ_{11}, σ_{22}, σ_{33}, σ_{12}, σ_{23}, σ_{31}.

    For three-dimensional elements, this storage scheme is inconsistent with that for ABAQUS/Standard.
- The shear strain is stored as tensor shear strains:

$$\varepsilon_{12} = \frac{1}{2}\gamma_{12}.$$

- The deformation gradient is stored similar to the way in which symmetric tensors are stored.
  - For plane stress elements: $F_{11}, F_{22}, F_{12}, F_{21}$.
  - For plane strain and axisymmetric elements: $F_{11}, F_{22}, F_{33}, F_{12}, F_{21}$.
  - For three-dimensional elements:
    $F_{11}, F_{22}, F_{33}, F_{12}, F_{23}, F_{31}, F_{21}, F_{32}, F_{13}$.

VUMAT Formulation Aspects

Vectorized Interface

- In **VUMAT** the data are passed in and out in large blocks (dimension **NBLOCK**). **NBLOCK** typically is equal to 64 or 128.
  - Each entry in an array of length **NBLOCK** corresponds to a single material point. All material points in the same block have the same material name and belong to the same element type.
- This structure allows vectorization of the routine.
  - A vectorized **VUMAT** should make sure that all operations are done in vector mode with **NBLOCK** the vector length.
- In vectorized code, branching inside loops should be avoided.
  - Element type-based branching should be outside the **NBLOCK** loop.

Corotational Formulation

- The constitutive equation is formulated in a corotational framework, based on the Jaumann stress rate.
  - The strain increment is obtained with Hughes-Winget.
  - Other measures can be obtained from the deformation gradient.
- The user must define the Cauchy stress: this stress reappears during the next increment as the “old” stress.
- There is no need to rotate tensor state variables.

Example 6: VUMAT for Kinematic Hardening

The governing equations and integration procedure are the same as in **Example 4: Kinematic Hardening Plasticity** (p. L6.54).

The Jacobian is not required.

Coding for Kinematic Hardening Plasticity VUMAT

```
C
      E      = PROPS(1)
      XNU    = PROPS(2)
      YIELD  = PROPS(3)
      HARD   = PROPS(4)
C
C    ELASTIC CONSTANTS
C
      TWOMU  = E / ( ONE + XNU )
      THREMU = THREE_HALFS * TWOMU
      SIXMU  = THREE * TWOMU
      ALAMDA = TWOMU * ( E - TWOMU ) / ( SIXMU - TWO * E )
      TERM   = ONE / ( TWOMU * ( ONE + HARD/THREMU ) )
      CON1   = SQRT( TWO_THIRDS )
C
C
```

```
C
C If stepTime equals to zero, assume the material pure elastic and use
C initial elastic modulus
C
      IF( STEPTIME .EQ. ZERO ) THEN
C
      DO I = 1,NBLOCK
C
C  Trial Stress
          TRACE = STRAININC (I, 1) + STRAININC (I, 2) + STRAININC (I, 3)
          STRESSNEW(I, 1)=STRESSOLD(I, 1) + ALAMDA*TRACE
     1    +               TWOMU*STRAININC(I,1)
          STRESSNEW(I, 2)=STRESSOLD(I, 2) + ALAMDA*TRACE
     1    +               TWOMU*STRAININC(I, 2)
          STRESSNEW(I, 3)=STRESSOLD(I, 3) + ALAMDA*TRACE
     1    +               TWOMU*STRAININC(I,3)
          STRESSNEW(I, 4)=STRESSOLD(I, 4)
     1    +               TWOMU*STRAININC(I, 4)
       END DO
C
       ELSE
```

```
C
C    PLASTICITY CALCULATIONS IN BLOCK FORM
C
      DO I = 1, NBLOCK
C  Elastic predictor stress
         TRACE = STRAININC(I, 1) + STRAININC(I, 2) + STRAININC(I, 3)
         SIG1= STRESSOLD(I, 1) + ALAMDA*TRACE + TWOMU*STRAININC(I, 1)
         SIG2= STRESSOLD(I, 2) + ALAMDA*TRACE + TWOMU*STRAININC(I, 2)
         SIG3= STRESSOLD(I, 3) + ALAMDA*TRACE + TWOMU*STRAININC(I, 3)
         SIG4= STRESSOLD(I, 4)                + TWOMU*STRAININC(I, 4)
C  Elastic predictor stress measured from the back stress
         S1 = SIG1 - STATEOLD(I, 1)
         S2 = SIG2 - STATEOLD(I, 2)
         S3 = SIG3 - STATEOLD(I, 3)
         S4 = SIG4 - STATEOLD(I, 4)
C  Deviatoric part of predictor stress measured from the back stress
         SMEAN = THIRD * ( S1 + S2 + S3 )
         DS1 = S1 - SMEAN
         DS2 = S2 - SMEAN
         DS3 = S3 - SMEAN
C  Magnitude of the deviatoric predictor stress difference
         DSMAG = SQRT( DS1**2 + DS2**2 + DS3**2 + TWO*S4**2 )
```

```
C  Check for yield by determining the factor for plasticity, zero for
C  elastic, one for yield
         RADIUS = CON1 * YIELD
         FACYLD = ZERO
         IF(  DSMAG - RADIUS .GE. ZERO ) FACYLD = ONE
C  Add a protective addition factor to prevent a divide by zero when DSMAG
C  is zero. If DSMAG is zero, we will not have exceeded the yield stress
C  and FACYLD will be zero.
         DSMAG  = DSMAG + ( ONE - FACYLD )
C  Calculated increment in gamma ( this explicitly includes the time step)
         DIFF   = DSMAG - RADIUS
         DGAMMA = FACYLD * TERM * DIFF
C  Update equivalent plastic strain
         DEQPS  = CON1 * DGAMMA
         STATENEW(I, 5) = STATEOLD(I, 5) + DEQPS
C  Divide DGAMMA by DSMAG so that the deviatoric stresses are explicitly
C  converted to tensors of unit magnitude in the following calculations
         DGAMMA = DGAMMA / DSMAG
C  Update back stress
         FACTOR  = HARD * DGAMMA * TWO_THIRDS
         STATENEW(I, 1) = STATEOLD(I, 1) + FACTOR * DS1
         STATENEW(I, 2) = STATEOLD(I, 2) + FACTOR * DS2
         STATENEW(I, 3) = STATEOLD(I, 3) + FACTOR * DS3
         STATENEW(I, 4) = STATEOLD(I, 4) + FACTOR *  S4
```

```
C  Update stress
         FACTOR   = TWOMU * DGAMMA
         STRESSNEW(I, 1) = SIG1 - FACTOR * DS1
         STRESSNEW(I, 2) = SIG2 - FACTOR * DS2
         STRESSNEW(I, 3) = SIG3 - FACTOR * DS3
         STRESSNEW(I, 4) = SIG4 - FACTOR *  S4
C  Update the specific internal energy -
         STRESS_POWER = HALF * (
     1         ( STRESSOLD(I, 1)+STRESSNEW(I, 1) )*STRAININC(I, 1)
     2   +     ( STRESSOLD(I, 2)+STRESSNEW(I, 2) )*STRAININC(I, 2)
     3   +     ( STRESSOLD(I, 3)+STRESSNEW(I, 3) )*STRAININC(I, 3)
     4   + TWO*( STRESSOLD(I, 4)+STRESSNEW(I, 4) )*STRAININC(I, 4) )
         ENERINTERNNEW(I) = ENERINTERNOLD(I)
     1   +                     STRESS_POWER/DENSITY(I)
C  Update the dissipated inelastic specific energy -
         SMEAN = THIRD* (STRESSNEW(I,1)+STRESSNEW(I,2)
     1   +                STRESSNEW(I,3))
         EQUIV_STRESS = SQRT( THREE_HALFS
     1   *                 ( (STRESSNEW(I, 1)-SMEAN)**2
     2   +                   (STRESSNEW(I, 2)-SMEAN)**2
     3   +                   (STRESSNEW(I, 3)-SMEAN)**2
     4   +             TWO * STRESSNEW(I, 4)**2 ) )
C
```

```
          PLASTIC_WORK_INC = EQUIV_STRESS * DEQPS
          ENERINELASNEW(I) = ENERINELASOLD(I)
     1    +                    PLASTIC_WORK_INC / DENSITY(I)
C
       END DO
C
       END IF
       RETURN
       END
```

Remarks

- In the **datacheck** phase, **`VUMAT`** is called with a set of fictitious strains and a **`TOTALTIME`** and **`STEPTIME`** both equal to 0.0.
  - A check is done on the user's constitutive relation, and an initial stable time increment is determined based on calculated equivalent initial material properties.
  - Ensure that elastic properties are used in this call to **`VUMAT`**; otherwise, too large an initial time increment may be used, leading to instability.
  - A warning message is printed to the status (**`.sta`**) file informing the user that this check is being performed.

- Special coding techniques are used to obtain vectorized coding.
  - All small loops inside the material routine are “unrolled.”
  - The same code is executed regardless of whether the behavior is purely elastic or elastic plastic.
- Special care must be taken to avoid divides by zero.
  - No external subroutines are called inside the loop.
  - The use of *local* scalar variables inside the loop is allowed.
  - The compiler will automatically expand these local scalar variables to local vectors.
  - Iterations should be avoided.
- If iterations cannot be avoided, use a fixed number of iterations and do not test on convergence.

Example 7: VUMAT for Isotropic Hardening

The governing equations and integration procedure are the same as in **Example 5: Isotropic Hardening Plasticity** (p. L6.69).

The increment of equivalent plastic strain is obtained explicitly through

$$\Delta\bar{\varepsilon}^{pl} = \frac{\bar{\sigma}^{pr} - \sigma_y}{3\mu + h},$$

where σ_y is the yield stress and $h = d\sigma_y / d\bar{\varepsilon}^{pl}$ is the plastic hardening at the beginning of the increment.

The Jacobian is not required.

Coding for Isotropic Hardening Plasticity VUMAT

```
C
C
      parameter ( zero = 0.d0, one = 1.d0, two = 2.d0,
     *     third = 1.d0 / 3.d0, half = 0.5d0, op5 = 1.5d0)
C
C For plane strain, axisymmetric, and 3D cases using
C the J2 Mises Plasticity with piecewise-linear isotropic hardening.
C
C The state variable is stored as:
C
C            STATE(*,1) = equivalent plastic strain
C
C User needs to input
C     props(1)      Young's modulus
C     props(2)      Poisson's ratio
C     props(3..)    syield and hardening data
C     calls vuhard for curve of yield stress vs. plastic strain
C
```

```
      e      = props(1)
      xnu    = props(2)
      twomu  = e / ( one + xnu )
      alamda = xnu * twomu / ( one - two * xnu )
      thremu = op5 * twomu
      nvalue = nprops/2-1
C
      if ( stepTime .eq. zero ) then
        do k = 1, nblock
          trace = strainInc(k,1) + strainInc(k,2) + strainInc(k,3)
          stressNew(k,1) = stressOld(k,1)
     *         + twomu * strainInc(k,1) + alamda * trace
          stressNew(k,2) = stressOld(k,2)
     *         + twomu * strainInc(k,2) + alamda * trace
          stressNew(k,3) = stressOld(k,3)
     *         + twomu * strainInc(k,3) + alamda * trace
          stressNew(k,4)=stressOld(k,4) + twomu * strainInc(k,4)
          if ( nshr .gt. 1 ) then
            stressNew(k,5)=stressOld(k,5) + twomu * strainInc(k,5)
            stressNew(k,6)=stressOld(k,6) + twomu * strainInc(k,6)
          end if
        end do
      else
```

```
do k = 1, nblock

  peeqOld=stateOld(k,1)
  call vuhard(yieldOld, hard, peeqOld, props(3), nvalue)

  trace = strainInc(k,1) + strainInc(k,2) + strainInc(k,3)

   s11 = stressOld(k,1) + twomu * strainInc(k,1) + alamda * trace
   s22 = stressOld(k,2) + twomu * strainInc(k,2) + alamda * trace
  s33 = stressOld(k,3) + twomu * strainInc(k,3) + alamda * trace
  s12 = stressOld(k,4) + twomu * strainInc(k,4)

  if ( nshr .gt. 1 ) then
    s13 = stressOld(k,5) + twomu * strainInc(k,5)
    s23 = stressOld(k,6) + twomu * strainInc(k,6)
  end if
```

```
C
          smean = third * ( s11 + s22 + s33 )

          s11 = s11 - smean
          s22 = s22 - smean
          s33 = s33 - smean

          if ( nshr .eq. 1 ) then

            vmises = sqrt( op5*(s11*s11+s22*s22+s33*s33+two*s12*s12) )

          else

            vmises = sqrt( op5 * ( s11 * s11 + s22 * s22 + s33 * s33 +
     *           two * s12 * s12 + two * s13 * s13 + two * s23 * s23 ) )

          end if
C
          sigdif = vmises - yieldOld
          facyld = zero
          if ( sigdif .gt. zero ) facyld = one
          deqps = facyld * sigdif / ( thremu + hard )
```

```
C
C Update the stress
C
          yieldNew = yieldOld + hard * deqps
          factor = yieldNew / ( yieldNew + thremu * deqps )

          stressNew(k,1) = s11 * factor + smean
          stressNew(k,2) = s22 * factor + smean
          stressNew(k,3) = s33 * factor + smean
          stressNew(k,4) = s12 * factor

          if ( nshr .gt. 1 ) then

            stressNew(k,5) = s13 * factor
            stressNew(k,6) = s23 * factor

          end if
C
C Update the state variables
C
          stateNew(k,1) = stateOld(k,1) + deqps
```

```
C
C Update the specific internal energy -
C
          if ( nshr .eq. 1 ) then

            stressPower = half * (
     *        ( stressOld(k,1) + stressNew(k,1) ) * strainInc(k,1) +
     *       ( stressOld(k,2) + stressNew(k,2) ) * strainInc(k,2) +
     *       ( stressOld(k,3) + stressNew(k,3) ) * strainInc(k,3) ) +
     *        ( stressOld(k,4) + stressNew(k,4) ) * strainInc(k,4)

          else

            stressPower = half * (
     *       ( stressOld(k,1) + stressNew(k,1) ) * strainInc(k,1) +
     *       ( stressOld(k,2) + stressNew(k,2) ) * strainInc(k,2) +
     *        ( stressOld(k,3) + stressNew(k,3) ) * strainInc(k,3) ) +
     *        ( stressOld(k,4) + stressNew(k,4) ) * strainInc(k,4) +
     *        ( stressOld(k,5) + stressNew(k,5) ) * strainInc(k,5) +
     *        ( stressOld(k,6) + stressNew(k,6) ) * strainInc(k,6)

          end if

          enerInternNew(k) = enerInternOld(k) + stressPower / density(k)
```

```
C
C Update the dissipated inelastic specific energy -
C
          plasticWorkInc = half * ( yieldOld + yieldNew ) * deqps
          enerInelasNew(k) = enerInelasOld(k)
     *         + plasticWorkInc / density(k)

        end do

       end if
C
       return
       end
```

```
      subroutine vuhard(syield, hard, eqplas, table, nvalue)
      include 'vaba_param.inc'
c
      dimension table(2, nvalue)
c
      parameter(zero=0.d0)
c
c    set yield stress to last value of table, hardening to zero
c
      syield=table(1, nvalue)
      hard=zero
c
c    if more than one entry, search table
c
      if(nvalue.gt.1) then
        do k1=1, nvalue-1
          eqpl1=table(2,k1+1)
          if(eqplas.lt.eqpl1) then
            eqpl0=table(2, k1)
c
c           yield stress and hardening
c
            deqpl=eqpl1-eqpl0
            syiel0=table(1, k1)
```

```
            syiel1=table(1, k1+1)
            dsyiel=syiel1-syiel0
            hard=dsyiel/deqpl
            syield=syiel0+(eqplas-eqpl0)*hard
            goto 10
          endif
        end do
10    continue
      endif

      return
      end
```

Remarks

- This **VUMAT** yields the same results as the *PLASTIC option with ISOTROPIC hardening.
  - This result is also true for large-strain calculations. The necessary rotations of stress and strain are taken care of by ABAQUS.
- The routine calls user subroutine **VUHARD** to recover a piecewise linear hardening curve.
  - It is straightforward to replace the piecewise linear curve by an analytic description.