User-defined subroutines for constitutive modelling in ABAQUS

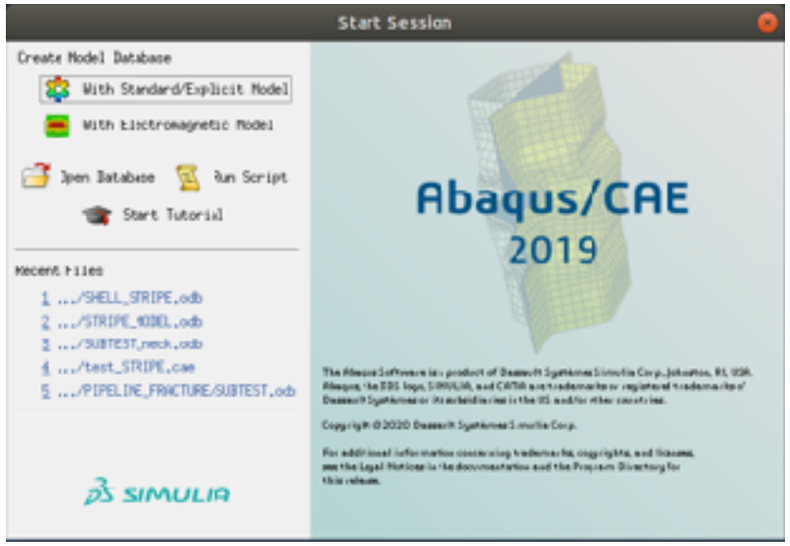
David Morin

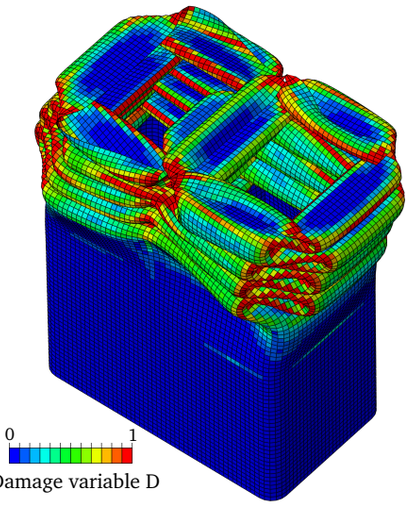
Norwegian University of Science and Technology

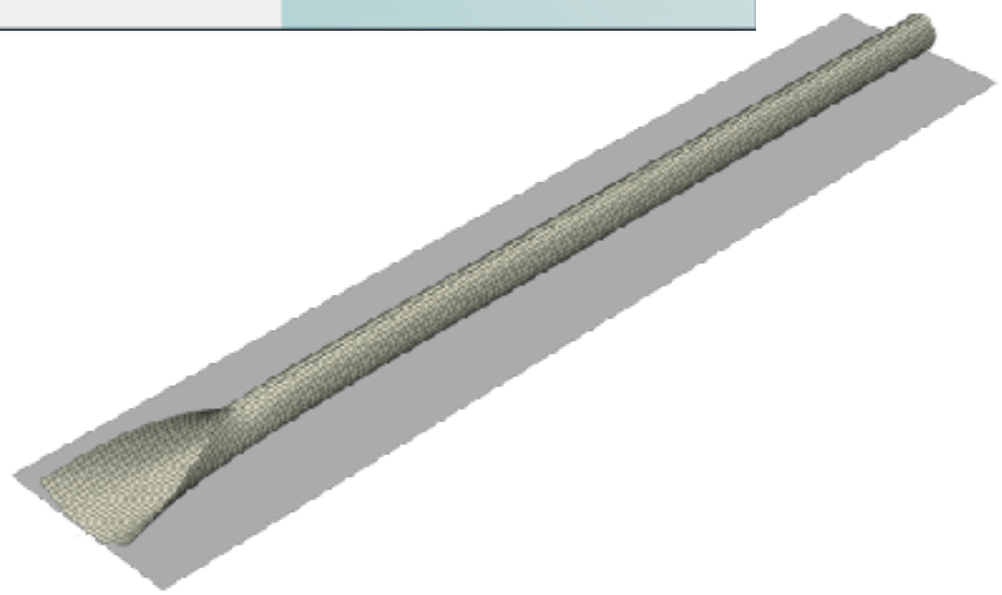
Context

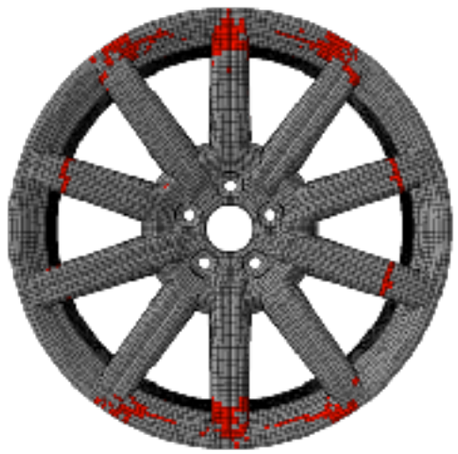
ABAQUS is a powerful Finite Element Analysis (FEA) software…

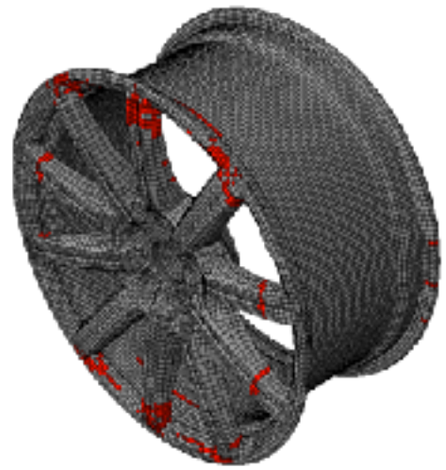
- Good pre/post-processor
- Many element formulations
- Many material models
- Powerful user coding capabilities





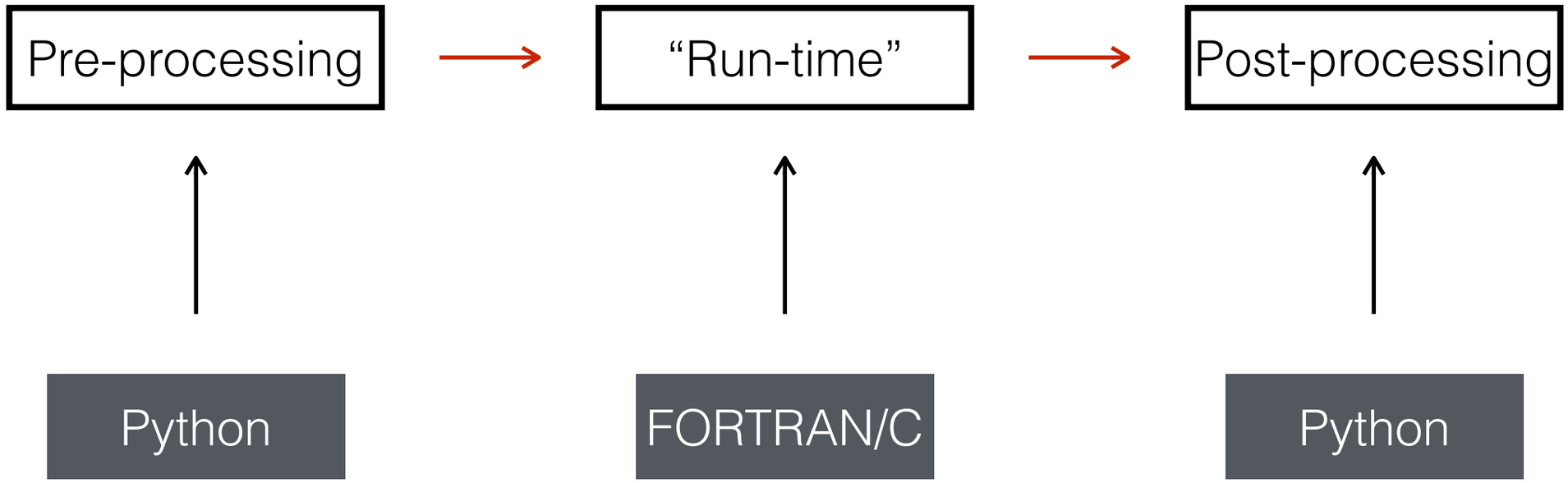






User-coding in ABAQUS

There are 3 *"big"* steps in ABAQUS and all of them offers some kind of user-coding:



User-subroutines in ABAQUS

There are many subroutines available in ABAQUS:

ABAQUS/standard

- CREEP
- DFLOW
- DFLUX
- DISP
- DLOAD
- FILM
- FLOW
- FRIC
- FRIC_COEF
- GAPCON
- GAPELECTR
- HARDINI
- HETVAL
- MPC
- ORIENT
- RSURFU
- SDVINI
- SIGINI
- UAMP
- UANISOHYPER_INV
- UANISOHYPER_STRAIN
- UCORR
- UCREEPNETWORK
- UDAMAGEVI
- UDECURRENT
- UDEMPOTENTIAL
- UDMGINI
- UDSECURRENT
- UEL
- UELMAT
- UEPACTIVATIONFACET
- UEPACTIVATIONSETUP
- UEPACTIVATIONVOL
- UEXPAN
- UEXTERNALDB
- UFIELD
- UFLUID
- UFLUIDCONNECTORLOSS
- UFLUIDCONNECTORVALVE
- UFLUIDLEAKOFF
- UHELIOPIPEFRICTION
- UGENS
- UHARD
- UHYPEL
- UHYPER
- UINTER
- UMASFL
- UMAT
- UMATHT
- UMDFLUX
- UMDFLUXSETUP
- UMESHMOTION
- UMIXMODEFATIGUE
- UMOTION
- UMULLIN3
- UPOREP
- UPRESS
- UPSD
- URDFIL
- USDFLD
- USUPERELASHARDMOD
- UTEMP
- UTRACLOAD
- UTRS
- UTRSNETWORK
- UVARM
- UWAVE
- UXFEMNONLOCALWEIGHT
- VOIDRI

ABAQUS/explicit

- VDFLUX
- VDISP
- VDLOAD
- VEXTERNALDB
- VFABRIC
- VFRIC
- VFRIC_COEF
- VFRICTION
- VHETVAL
- VUAMP
- VUANISOHYPER_INV
- VUANISOHYPER_STRAIN
- VUCHARLENGTH
- VUCREEPNETWORK
- VUEL
- VUEOS
- VUEXPAN
- VUFIELD
- VUFLUIDEXCH
- VUFLUIDEXCHEFFAREA
- VUHARD
- VUINTER
- VUINTERACTION
- VUMAT
- VUMATHT
- VUMULLINS
- VUSDFLD
- VUSUPERELASHARDMOD
- VUTRS
- VUVISCOSITY
- VWAVE

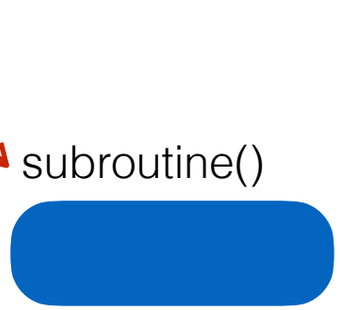
User-subroutines in ABAQUS

ABAQUS/standard and ABAQUS/explicit do not have the same subroutines:

- ABAQUS/explicit uses a vectorised system
- ABAQUS/standard uses a scalar system

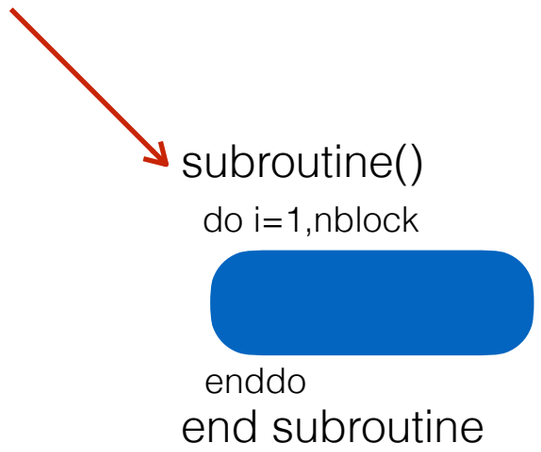
ABAQUS/standard

One integration point



ABAQUS/explicit

A block of integration point

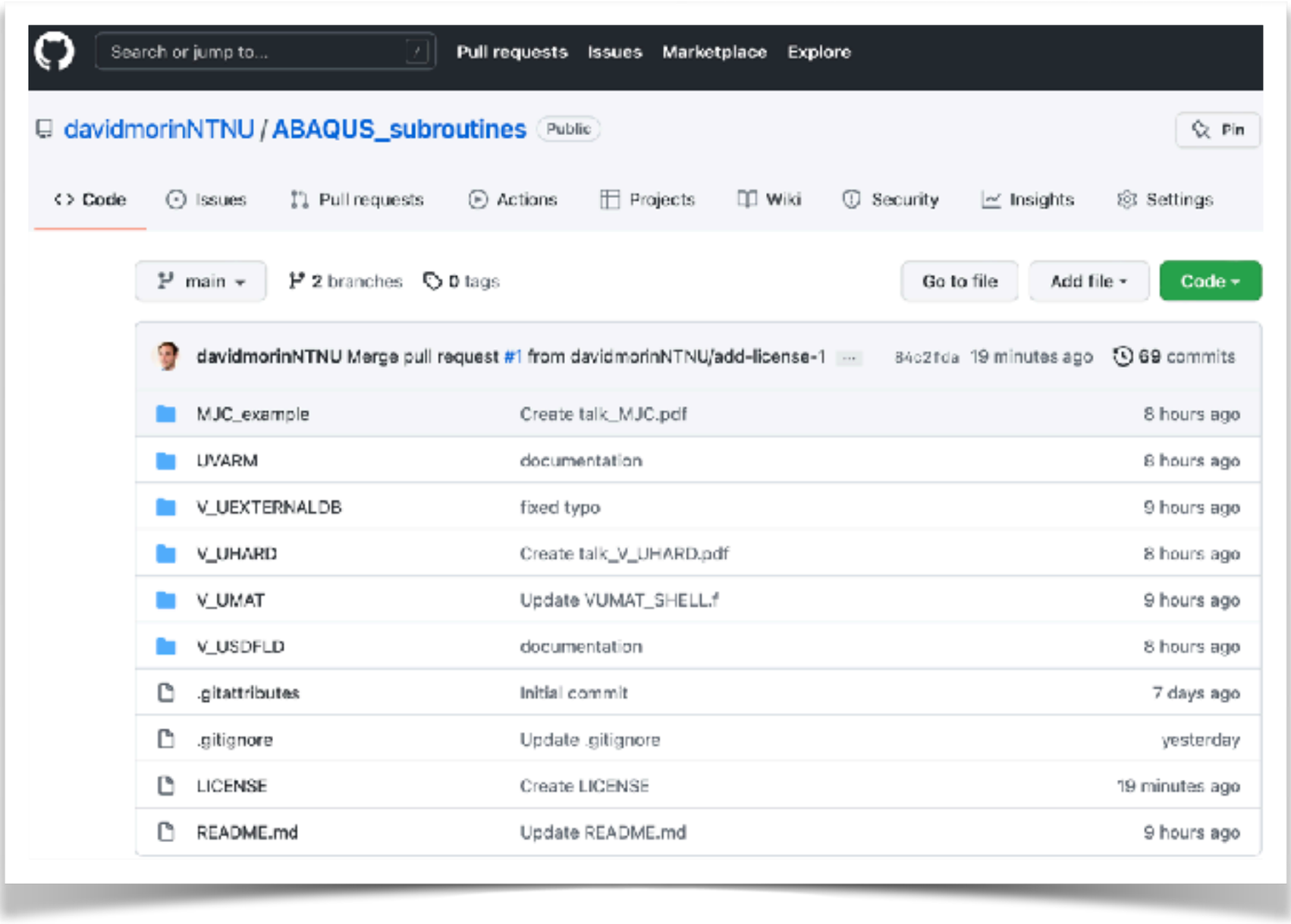


This lecture covers only some subroutines related to “material modelling”

User-subroutines in ABAQUS

To get started:

https://github.com/davidmorinNTNU/ABAQUS_subroutines



User-subroutines in ABAQUS

To get started:

https://github.com/davidmorinNTNU/ABAQUS_subroutines

main ▾ ABAQUS_subroutines / V_UHARD / Go to file Add file ▾ ···

davidmorinNTNU Create talk_V_UHARD.pdf ··· 4376af8 yesterday History

..

UHARD_V1.f	Update UHARD_V1.f	4 days ago
VUHARD_V1.f	Update VUHARD_V1.f	5 days ago
VUHARD_V2.f	Create VUHARD_V2.f	3 days ago
example_UHARD_V1.inp	Update boundary conditions V_UHARD_V1 example file	4 days ago
example_VUHARD_V1.inp	fixed typo	yesterday
example_VUHARD_V2.inp	fixed typo	yesterday
plot_two_elements_results_VUHARD_V2.py	added post-processing scripts for VUHARD_V2 example.	3 days ago
plot_two_elements_results_V_UHARD_V1.py	post-processing scripts V_UHARD	4 days ago
post_two_elements_V_UHARD_V1.py	post-processing scripts V_UHARD	4 days ago
post_two_elements_V_UHARD_V2.py	added post-processing scripts for VUHARD_V2 example.	3 days ago
talk_V_UHARD.pdf	Create talk_V_UHARD.pdf	yesterday

User-subroutines in ABAQUS

Why should we develop/implement something into ABAQUS?

- Lack of a particular model
- Improvement of an existing model

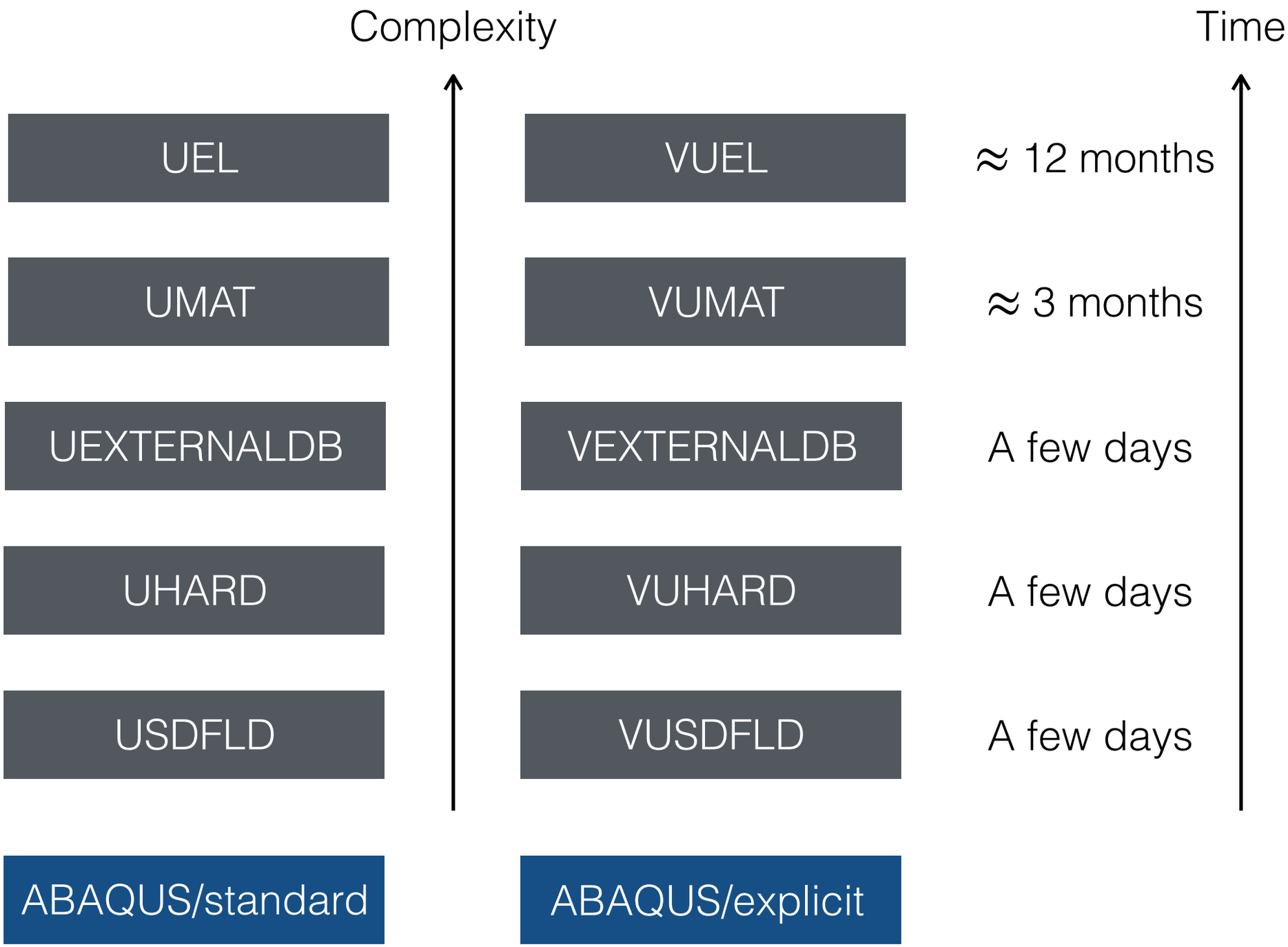
Things to consider before implementing any model:

- Can you work with an existing model?
- Will someone else use your code?

User-subroutines in ABAQUS

UEL/VUEL	**UEL:** User subroutine to define an element.
UMAT/VUMAT	**UMAT:** User subroutine to define a material's mechanical behavior.
V-UEXTERNALDB	**UEXTERNALDB:** User subroutine to manage user-defined external databases and calculate model-independent history information.
UHARD/VUHARD	**UHARD:** User subroutine to define the yield surface size and hardening parameters for isotropic plasticity or combined hardening models.
USDFLD/VUSDFLD	**USDFLD:** User subroutine to redefine field variables at a material point.
ABAQUS	

User-subroutines in ABAQUS



(V)USDFLD subroutine

USDFLD: **User subroutine to redefine field variables at a material point.**

VUSDFLD: **User subroutine to redefine field variables at a material point.**

(V)USDFLD subroutine

ABAQUS/standard

Overview

User subroutine **`USDFLD`**:

- allows you to define field variables at a material point as functions of time or of any of the available material point quantities listed in the Output Variable Identifiers table ("Abaqus/Standard output variable identifiers," Section 4.2.1 of the Abaqus Analysis User's Guide) except the user-defined output variables UVARM and UVARM*n*;
- can be used to introduce solution-dependent material properties since such properties can be easily defined as functions of field variables;
- will be called at all material points of elements for which the material definition includes user-defined field variables;
- must call utility routine **`GETVRM`** to access material point data;
- can use and update state variables; and
- can be used in conjunction with user subroutine **`UFIELD`** to prescribe predefined field variables.

Field variables

State dependent variables

ABAQUS/explicit

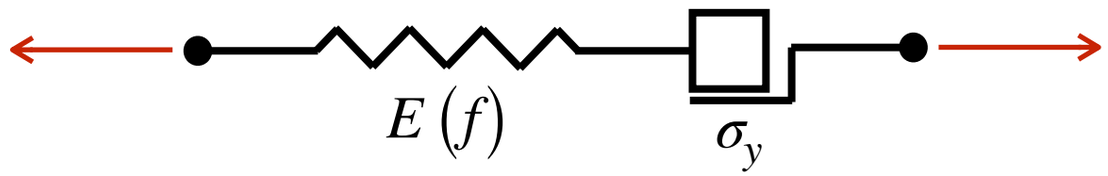
Overview

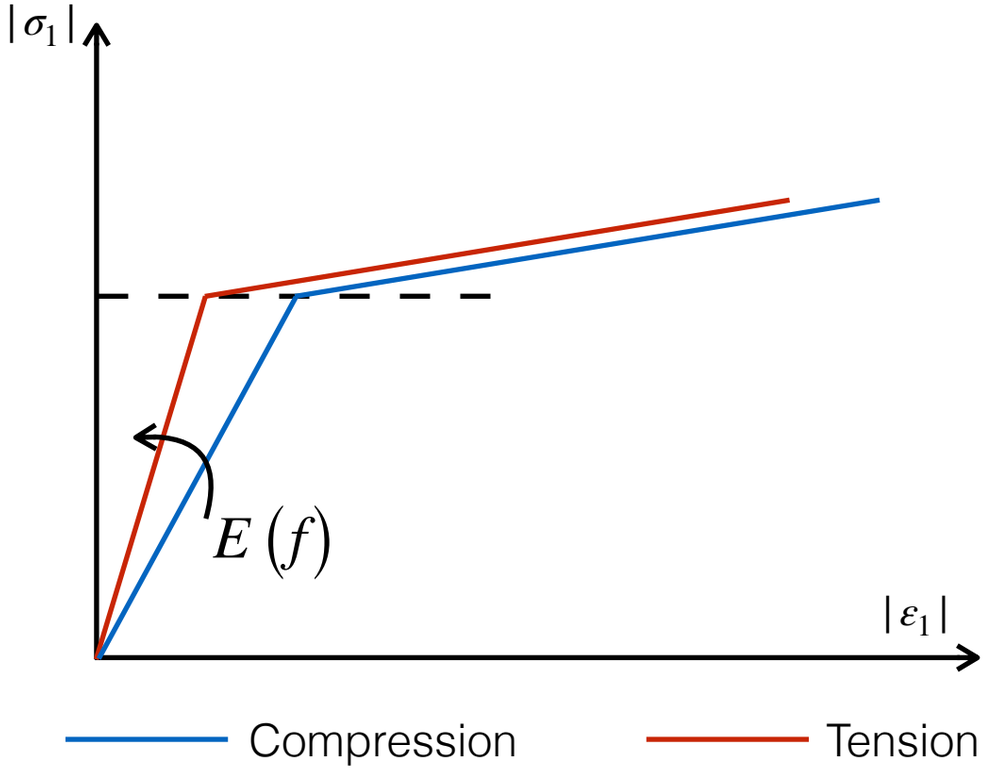
User subroutine **`VUSDFLD`**:

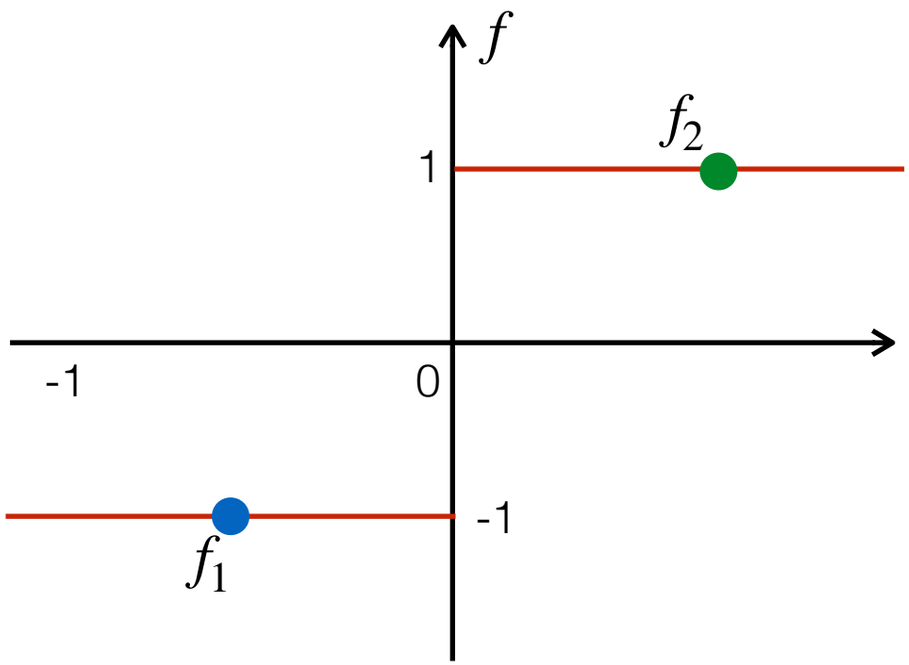
- allows the redefinition of field variables at a material point as functions of time or of any of the available material point quantities listed in "Available output variable keys" in "Obtaining material point information in an Abaqus/Explicit analysis," Section 2.1.7;
- can be used to introduce solution-dependent material properties since such properties can be easily defined as functions of field variables;
- will be called at all material points of elements for which the material definition includes user-defined field variables;
- can call utility routine **`VGETVRM`** to access material point data; and
- can use and update solution-dependent state variables.

(V)USDFLD subroutine

In this example, we want a material model where the Young's modulus E is different between tension and compression.







$$E\left(f_1\right) < E\left(f_2\right)$$

(V)USDFLD subroutine

Material cards for both solvers are identical.

ABAQUS/explicit

```
77   **----------------------------------------------------------------
78   ** MATERIALS
79   **----------------------------------------------------------------
80   *material,name=EXAMPLE_VUSDLFD_V1
81   *density
82   7.8e-9
83   *elastic,dependencies=1
84   **      E,  NU, TEMP,    F
85    21000.0, 0.3,  0.0, -1.0
86   210000.0, 0.3,  0.0,  1.0
87   *plastic
88   250.0, 0.0
89   350.0, 1.0
90   *User defined field
91   **----------------------------------------------------------------
```

Activate a field dependency

Tabulated data with the field variable f
Temperature must always be present

Call to the VUSDFLD subroutine

ABAQUS/standard

```
77   **----------------------------------------------------------------
78   ** MATERIALS
79   **----------------------------------------------------------------
80   *material,name=EXAMPLE_USDLFD_V1
81   *elastic,dependencies=1
82   **      E,  NU, TEMP,    F
83    21000.0, 0.3,  0.0, -1.0
84   210000.0, 0.3,  0.0,  1.0
85   *plastic
86   250.0, 0.0
87   350.0, 1.0
88   *User defined field
89   **----------------------------------------------------------------
```

(V)USDFLD subroutine

```
      SUBROUTINE USDFLD(FIELD,STATEV,PNEWDT,DIRECT,T,CELENT,
     + TIME,DTIME,CMNAME,ORNAME,NFIELD,NSTATV,NOEL,NPT,LAYER,
     + KSPT,KSTEP,KINC,NDI,NSHR,COORD,JMAC,JMATYP,MATLAYO,LACCFLA)
      INCLUDE 'ABA_PARAM.INC'
!-----------------------------------------------------------------
!-----Declaration ABAQUS variables
!-----------------------------------------------------------------
      CHARACTER*80 CMNAME,ORNAME
      DIMENSION FIELD(NFIELD),STATEV(NSTATV),DIRECT(3,3)
      DIMENSION T(3,3),TIME(2),COORD(*),JMAC(*),JMATYP(*)
!-----Data from ABAQUS
      DIMENSION ARRAY(15),JARRAY(15)
      CHARACTER*3 FLGRAY(15)
!-----------------------------------------------------------------
!-----Declaration internal variables
!-----------------------------------------------------------------
      real*8 SIGH,SMISES,TRIAX
!-----------------------------------------------------------------
!     Access stress invariants
!-----------------------------------------------------------------
      CALL GETVRM('SINV',ARRAY,JARRAY,FLGRAY,JRCD,
     +            JMAC,JMATYP,MATLAYO,LACCFLA)
c
      SIGH   = ARRAY(3)
      SMISES = ARRAY(1)
```

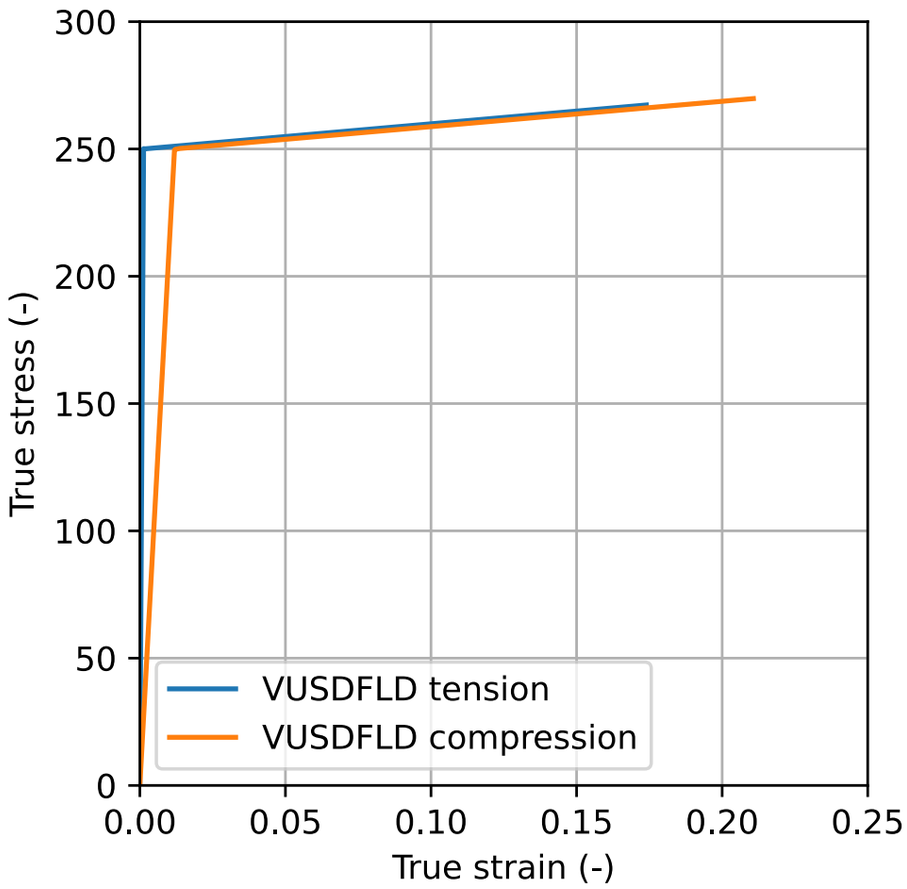
Access to ABAQUS variables

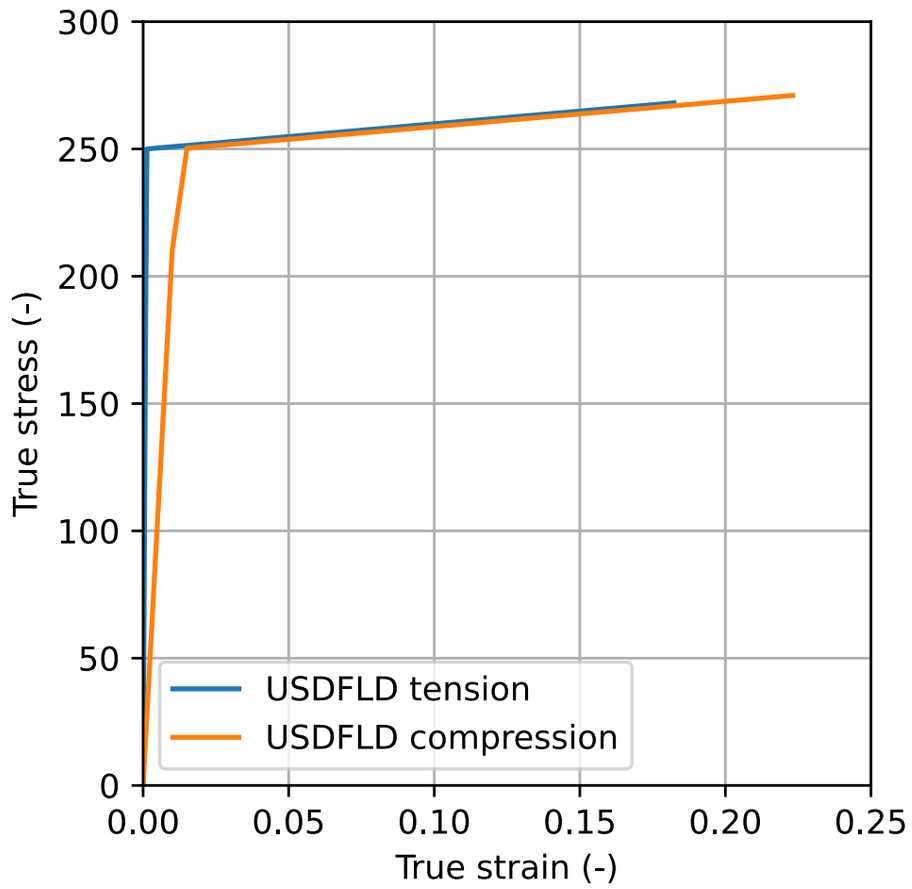
```
!-----------------------------------------------------------------
!     Compute the stress triaxiality
!-----------------------------------------------------------------
      if(SMISES.gt.0.0)then
         TRIAX = -SIGH/SMISES
      else
         TRIAX = 0.0
      endif
!-----------------------------------------------------------------
!     Update field variable
!-----------------------------------------------------------------
      if(TRIAX.ge.0.0)then
         FIELD(1) = 1.05
      else
         FIELD(1) =-1.05
      endif
!-----------------------------------------------------------------
!     End of subroutine
!-----------------------------------------------------------------
      RETURN
      END
```

Field variable f

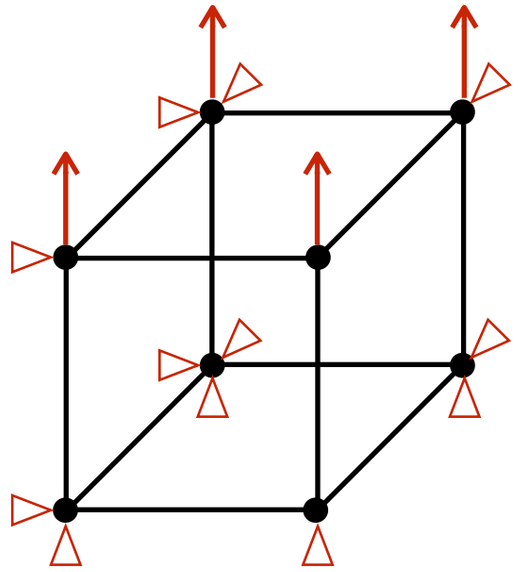
(V)USDFLD subroutine

Results from the USDFLD_V1 and VUSDFLD_V1 subroutines

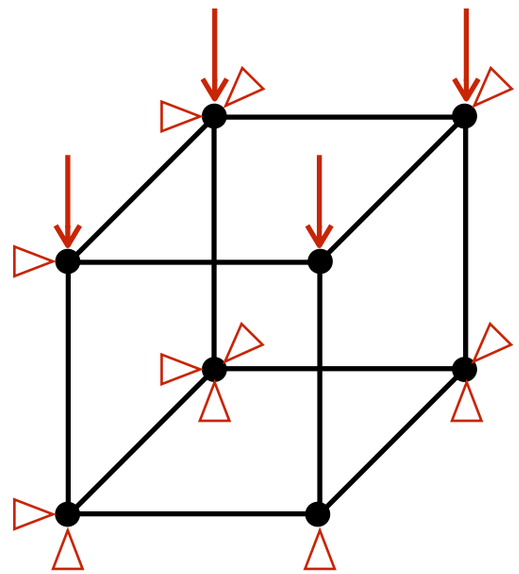




Single solid element
in uniaxial tension



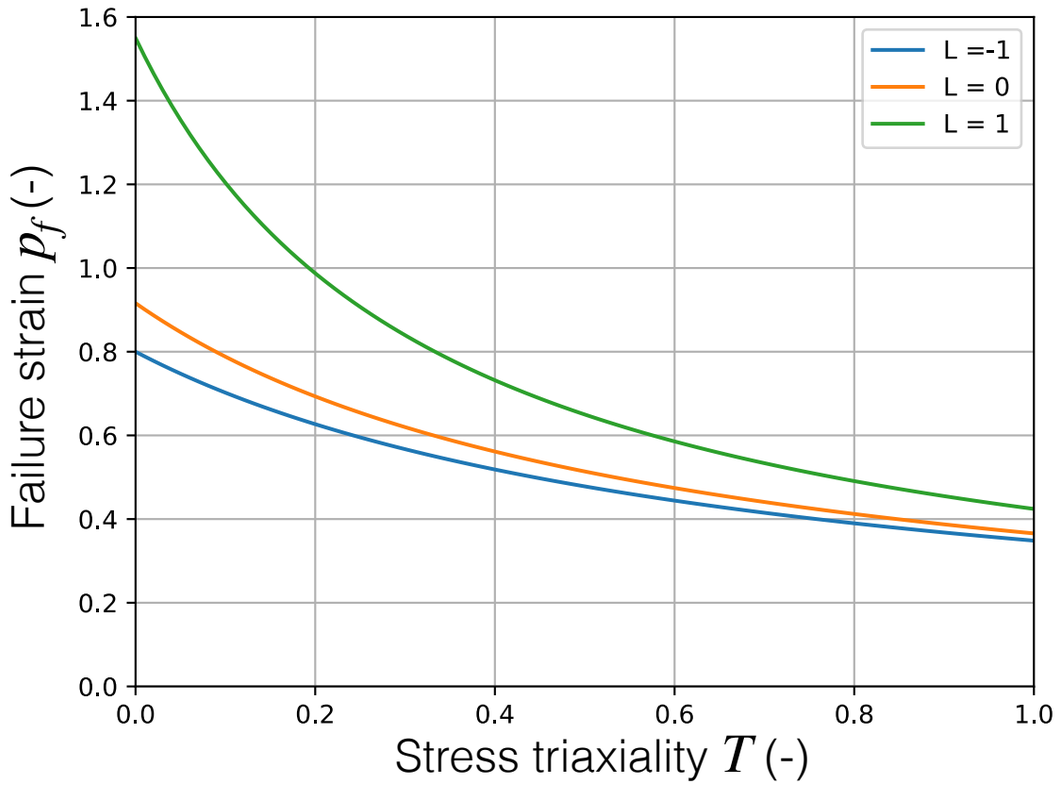
Single solid element in
uniaxial compression

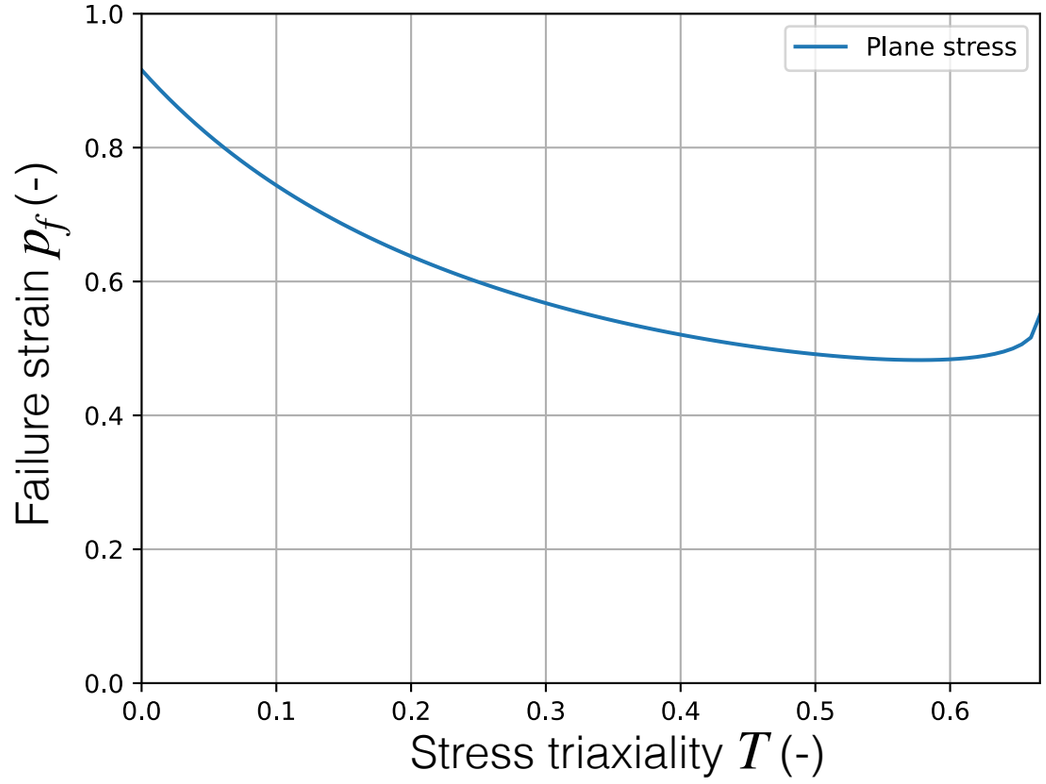


(V)USDFLD subroutine

In this example, we want to add a user-defined fracture model to a built-in plasticity model from ABAQUS.

Cockcroft-Latham criterion: $$D = \int_0^{p_f} \frac{\langle \sigma_1 \rangle}{W_c} dp \leq 1$$





(V)USDFLD subroutine

Cockcroft-Latham criterion:

$$D = \int_0^{p_f} \frac{\langle \sigma_1 \rangle}{W_c} dp \leq 1$$

Call to the VUSDFLD subroutine

W_c

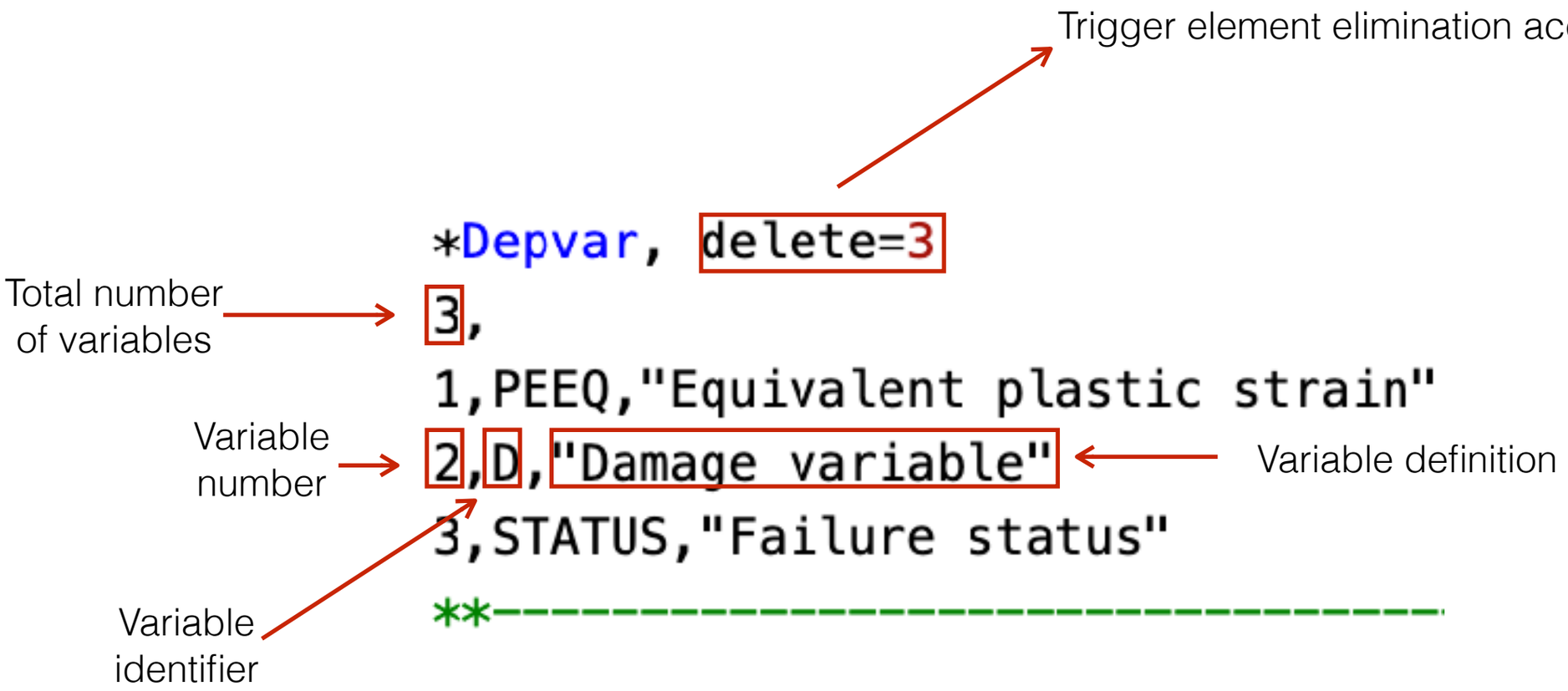
State dependent variables

```
**--------------------------------
** MATERIALS
**--------------------------------
*material,name=EXAMPLE_VUSDLFD_V2
*density
7.8e-9
*elastic
210000.0, 0.3
*plastic
250.0, 0.0
350.0, 1.0
*User defined field, properties=1
100.0
*Depvar, delete=3
3,
1,PEEQ,'Equivalent plastic strain'
2,D,'Damage variable'
3,STATUS,'Failure status'
**--------------------------------
```

Properties can not be given to the ABAQUS/standard USDFLD subroutine

(V)USDFLD subroutine

State Dependent Variables (SDV):



(V)USDFLD subroutine

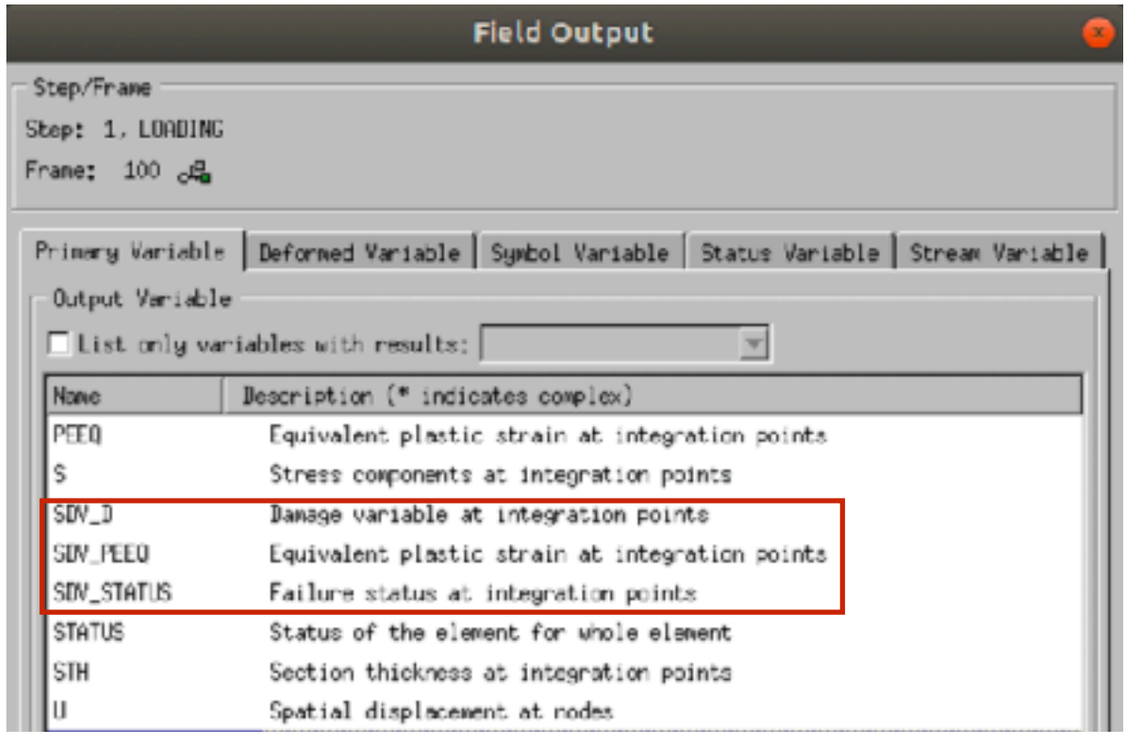
Input file structure:

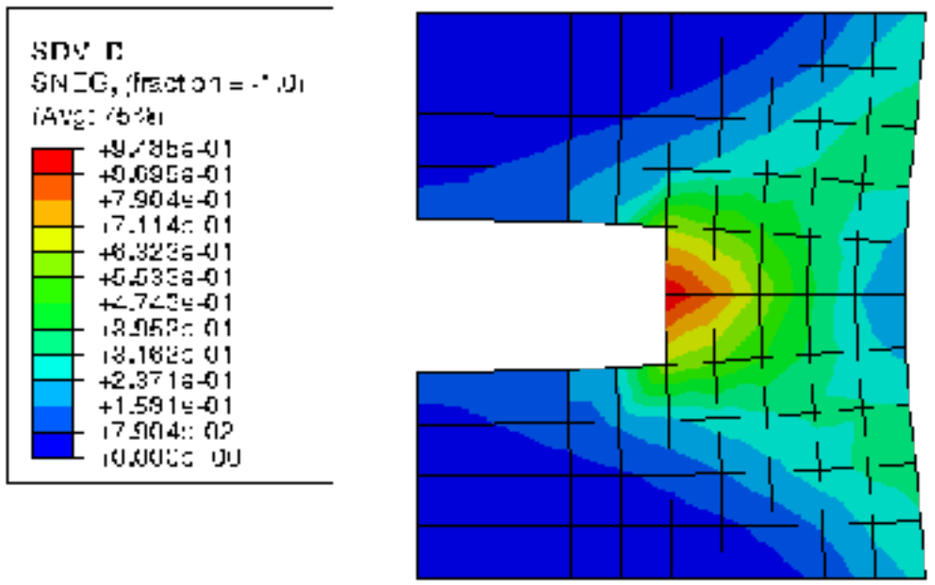
```
**----------------------------------------------------
** FIELD OUTPUT
**----------------------------------------------------
*Output, field, number interval=100, time marks=YES
*Element Output, directions=YES
PEEQ, S, SDV, STATUS
*Node Output
U
```

State dependent variables

Status for element elimination (from odb file)

Odb file structure:





(V)USDFLD subroutine

ABAQUS/standard

ABAQUS/explicit

Status = 0 means inactive (failed) integration point

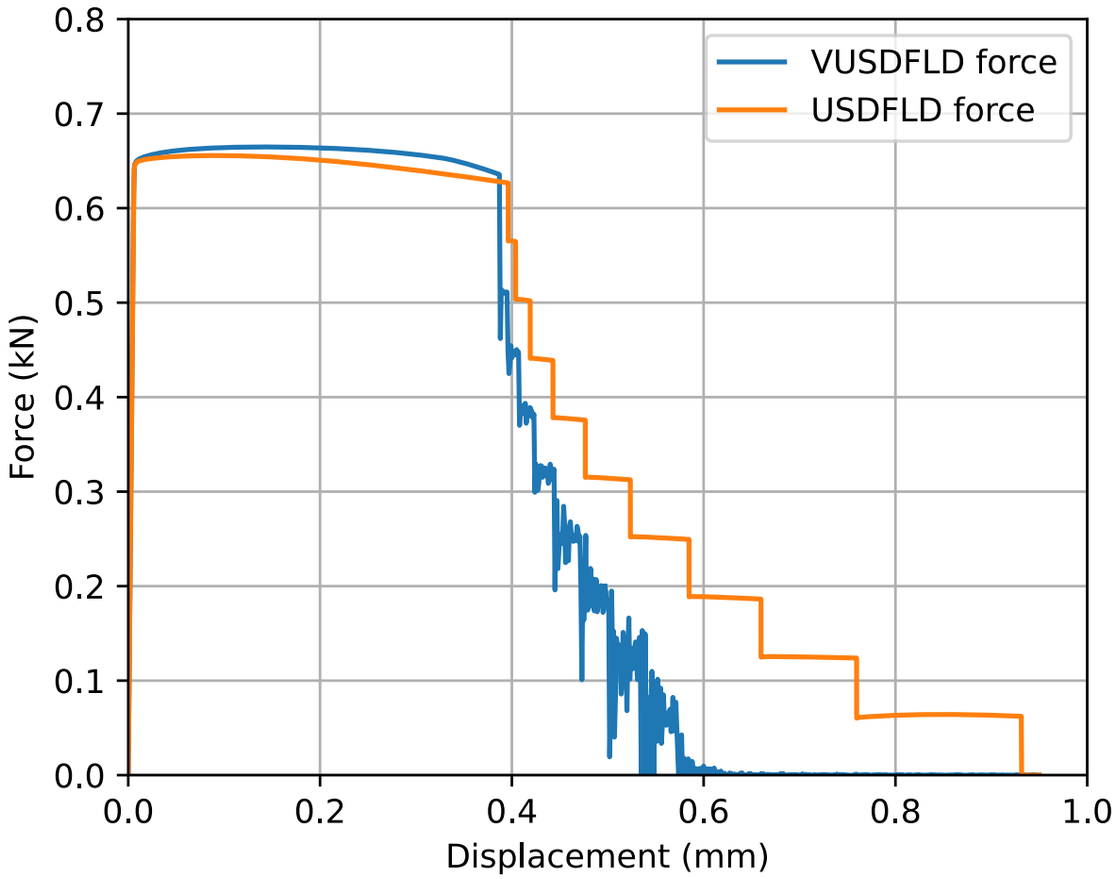
```
63    !-----------------------------------
64    !     Check for fracture
65    !-----------------------------------
66          if(damage.ge.1.0)then
67             STATEV(3) = 0.0
68          else
69             STATEV(3) = 1.0
70          endif
71    !-----------------------------------
```

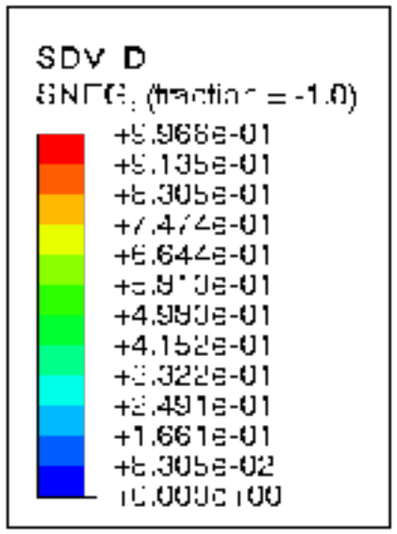
```
101   !-----------------------------------
102   !     Check for fracture
103   !-----------------------------------
104         do i=1,nblock
105            if(damage(i).ge.1.0)then
106               stateNew(i,3) = 0.0
107            else
108               stateNew(i,3) = 1.0
109            endif
110         enddo
```

Status = 1 means active integration point

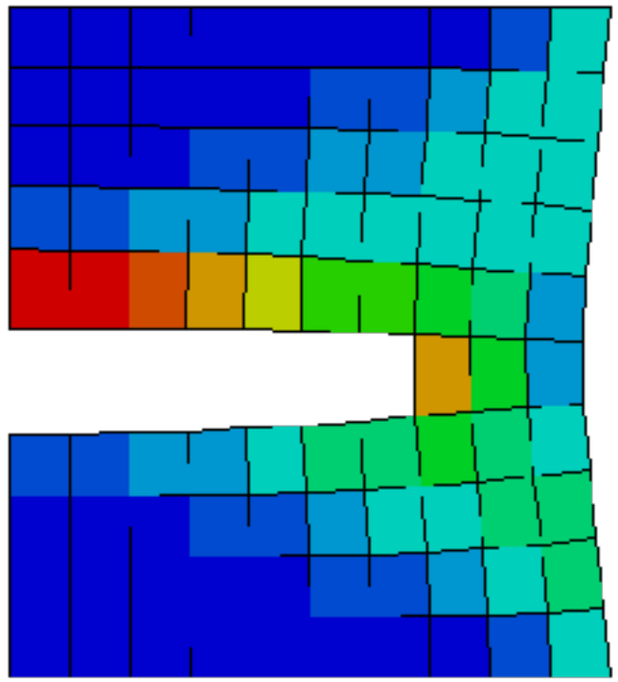
(V)USDFLD subroutine

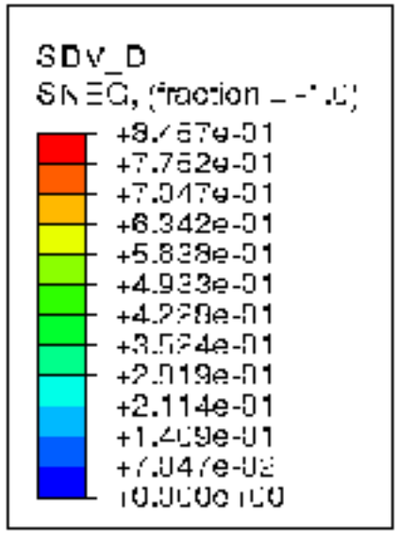
Results from the USDFLD_V2 and VUSDFLD_V2 subroutines



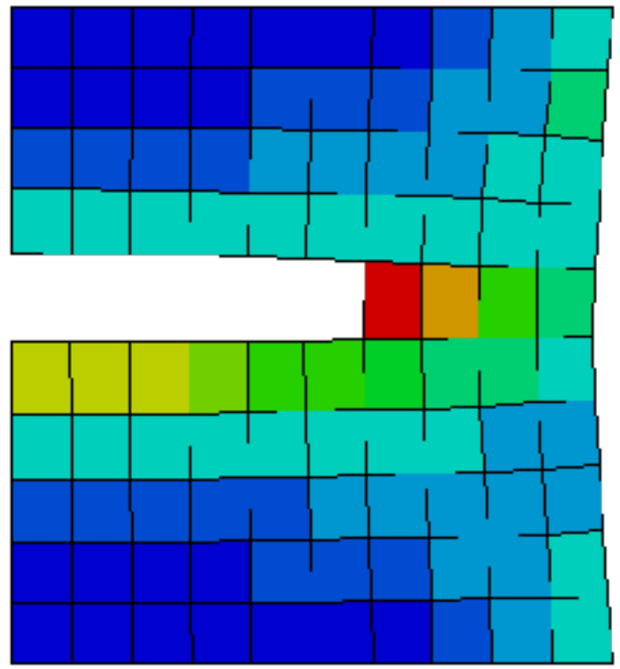


VUSDFLD_V2





USDFLD_V2



(V)USDFLD subroutine

These are quite flexible subroutine which can be used for various applications.

- No properties for the USDFLD subroutine (only for explicit)
- The explicit update of SDVs
- Limited number of variables in ABAQUS/explicit accessible through the vgetvrm subroutine.

(V)UMAT subroutine

UMAT: **User subroutine to define a material's mechanical behavior.**

WARNING: The use of this subroutine generally requires considerable expertise. You are cautioned that the implementation of any realistic constitutive model requires extensive development and testing. Initial testing on a single-element model with prescribed traction loading is strongly recommended.

VUMAT: **User subroutine to define material behavior.**

(V)UMAT subroutine

Overall principle:

Hypo-elastic(-plastic) material model: $\varepsilon = \varepsilon_e + \varepsilon_p$

Hyper-elastic(-plastic) material model: $\mathbf{F} = \mathbf{F}_e \cdot \mathbf{F}_p$

H are the State Dependent Variables

Deformation gradient tensor $\mathbf{F}$

(V)UMAT subroutine

ABAQUS/standard

$\boldsymbol{\sigma}(t+\Delta t), \boldsymbol{\sigma}(t)$ $H(t+\Delta t), H(t)$

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

$\Delta\boldsymbol{\varepsilon}(t+\Delta t)$

$\mathbf{F}$

ABAQUS/explicit

$H(t)$

```
      SUBROUTINE VUMAT(
     + NBLOCK, NDIR, NSHR, NSTATEV, NFIELDV, NPROPS, LANNEAL, STEPTIME,
     + TOTALTIME, DT, CMNAME, COORDMP, CHARLENGTH, PROPS, DENSITY,
     + STRAININC, RELSPININC, TEMPOLD, STRETCHOLD, DEFGRADOLD, FIELDOLD,
     + STRESSOLD, STATEOLD, ENERINTERNOLD, ENERINELASOLD, TEMPNEW,
     + STRETCHNEW, DEFGRADNEW, FIELDNEW,
     + STRESSNEW, STATENEW, ENERINTERNNEW, ENERINELASNEW)
```

$\Delta\boldsymbol{\varepsilon}(t+\Delta t)$

$\boldsymbol{\sigma}(t)$

$\mathbf{U}(t+\Delta t)$

$\mathbf{F}(t)$

$\mathbf{U}(t)$

$\boldsymbol{\sigma}(t+\Delta t)$

$H(t+\Delta t)$

$\mathbf{F}$

(V)UMAT subroutine

Some differences between UMAT and VUMAT

ABAQUS/standard

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{31} \\ \sigma_{23} \end{bmatrix} \quad \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ 2\varepsilon_{12} \\ 2\varepsilon_{31} \\ 2\varepsilon_{23} \end{bmatrix}$$

Jaumann:

$$\boldsymbol{\sigma}^{\nabla J} = \dot{\boldsymbol{\sigma}} - \mathbf{W} \cdot \boldsymbol{\sigma} + \boldsymbol{\sigma} \cdot \mathbf{W}$$

The user must ensure the objectivity of all tensor stored in the state dependent variables

Ordering tensor components

Stress rate (objectivity)

ABAQUS/explicit

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{31} \end{bmatrix} \quad \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \varepsilon_{12} \\ \varepsilon_{23} \\ \varepsilon_{31} \end{bmatrix}$$

Green-Naghdi:

$$\boldsymbol{\sigma}^{\nabla G} = \dot{\boldsymbol{\sigma}} - \boldsymbol{\Omega} \cdot \boldsymbol{\sigma} + \boldsymbol{\sigma} \cdot \boldsymbol{\Omega}$$

The objectivity of all tensor stored in the state dependent variables is handled by ABAQUS

(V)UMAT subroutine

The implicit solver requires the tangent operator $\frac{\partial \Delta \boldsymbol{\sigma}}{\partial \Delta \boldsymbol{\varepsilon}}$

to form the stiffness matrix of the model.

ABAQUS/standard

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

- The tangent operator (DDSDDE) is a common source of error leading to slow convergence and crashes
- By default the consistent tangent operator is expected (full newton solver)

- A workaround:
  - Supply the elasticity matrix into DDSDDE
  - Use the quasi-newton solution technique

(V)UMAT subroutine

ABAQUS/explicit

At the first time step:

- A purely elastic computation for time-step computation
- Be careful with visco-elasticity, anisotropic elasticity…

$$\Delta t \approx l_e \sqrt{\frac{\rho}{E}}$$

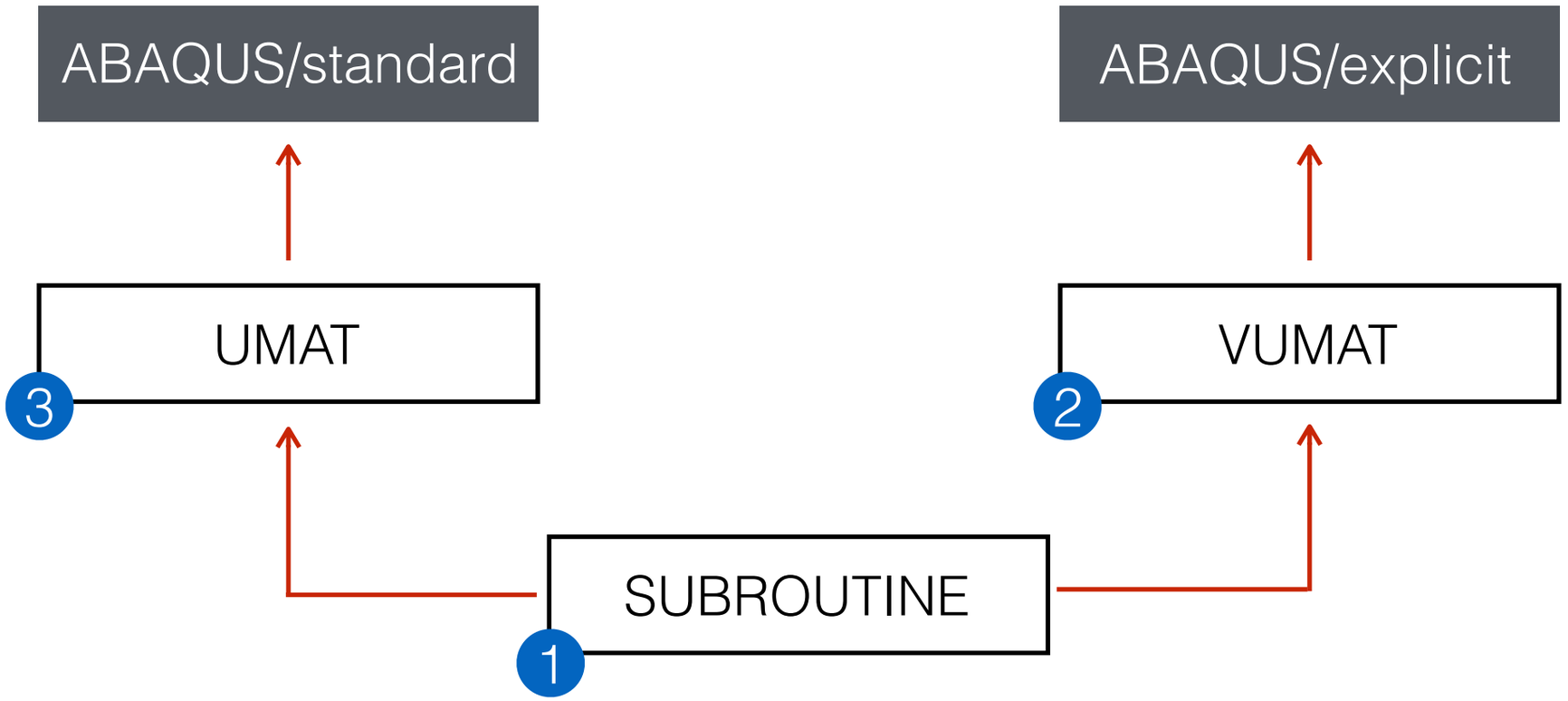
where:

- l_e is the element characteristic length
- ρ is the density of the material
- E is the Young's modulus

```
!-----------------------------------------------------------------
!     Initialization step (elastic)
!-----------------------------------------------------------------
      if((steptime.eq.totaltime).and.(steptime.eq.zero))then
         E0   = props(1)
         NU   = props(2)
         BULK = E0/(3.0*(1.0-2.0*NU))
         R2G  = E0/(1.0+NU)
         do i=1,NBLOCK
            if(NSHR.eq.3)then
               volstrain = STRAININC(i,1)+STRAININC(i,2)+STRAININC(i,3)
               STRESSNEW(i,1) = R2G*(STRAININC(i,1)-third*volstrain)
     +                          +BULK*volstrain
               STRESSNEW(i,2) = R2G*(STRAININC(i,2)-third*volstrain)
     +                          +BULK*volstrain
               STRESSNEW(i,3) = R2G*(STRAININC(i,3)-third*volstrain)
     +                          +BULK*volstrain
               STRESSNEW(i,4) = R2G*STRAININC(i,4)
               STRESSNEW(i,5) = R2G*STRAININC(i,5)
               STRESSNEW(i,6) = R2G*STRAININC(i,6)
            else
               volstrain = STRAININC(i,1)+STRAININC(i,2)+STRAININC(i,3)
               STRESSNEW(i,1) = R2G*(STRAININC(i,1)-third*volstrain)
     +                          +BULK*volstrain
               STRESSNEW(i,2) = R2G*(STRAININC(i,2)-third*volstrain)
     +                          +BULK*volstrain
               STRESSNEW(i,3) = R2G*(STRAININC(i,3)-third*volstrain)
     +                          +BULK*volstrain
               STRESSNEW(i,4) = R2G*STRAININC(i,4)
            endif
         enddo
!-----------------------------------------------------------------
```

(V)UMAT subroutine

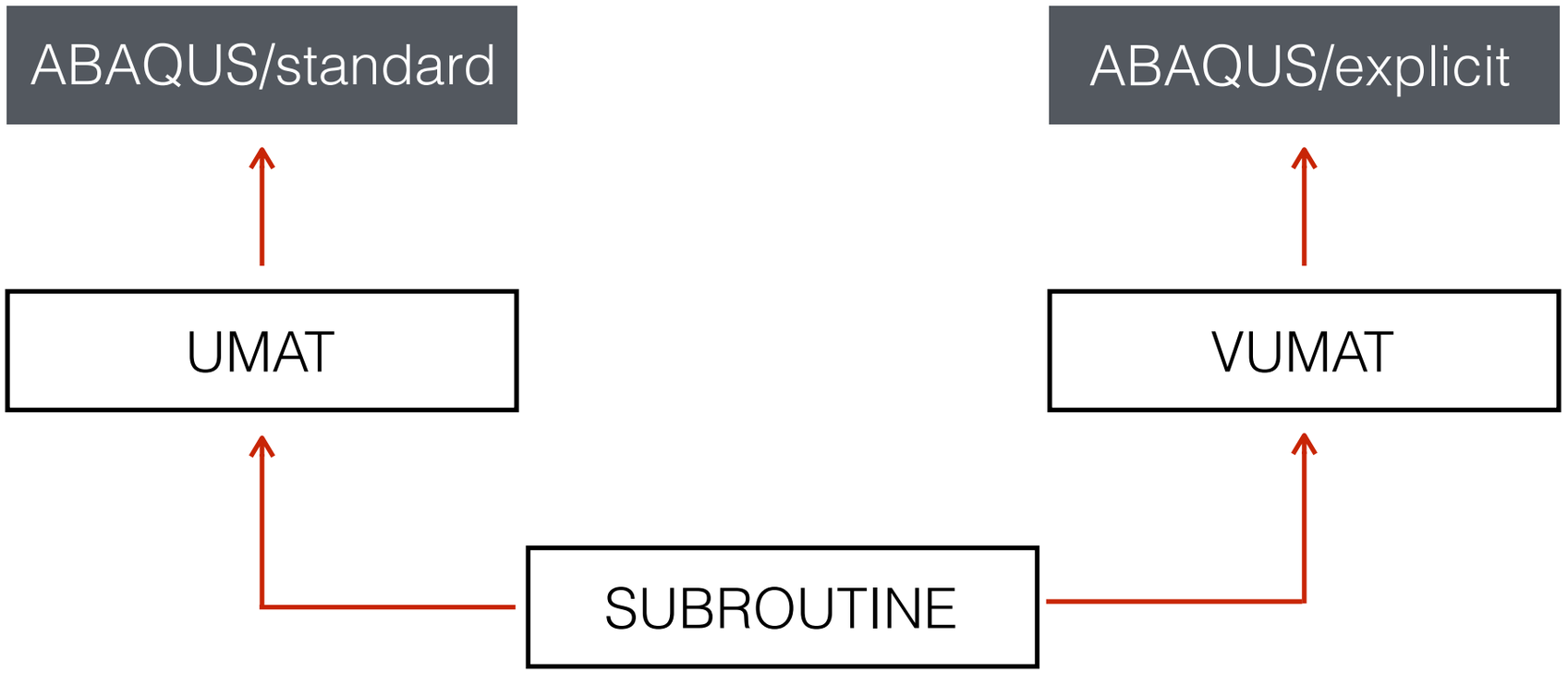
A simplified approach to UMAT/VUMAT coding:



1. Code and compile the material model in FORTRAN
2. Link the code to the VUMAT subroutine
3. Link the code to the UMAT subroutine

(V)UMAT subroutine

A simplified approach to UMAT/VUMAT coding:



- Easier to debug
- A model available in both ABAQUS/standard and explicit

- Less efficient code

(V)UMAT subroutine

ABAQUS/standard

- Include a subroutine with the code
- Send data to the subroutine

```
      INCLUDE './UMAT_MODEL.f'
!-----------------------------------------------------------------------
      SUBROUTINE UMAT(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
     +                RPL,DDSDDT,DRPLDE,DRPLDT,
     +                STRAN,DSTRAN,TIMEA,DTIMEA,TEMP,DTEMP,PREDEF,DPRED,
     +                CMNAME,NDI,NSHR,NTENS,NSTATV,PROPS,NPROPS,COORDS,
     +                DROT,PNEWDT,CELENT,DFGRD0,DFGRD1,NOEL,NPT,LAYER,
     +                KSPT,KSTEP,KINC)
      INCLUDE 'ABA_PARAM.INC'
!-----------------------------------------------------------------------
!-----Declaration ABAQUS variables
!-----------------------------------------------------------------------
      character*80 CMNAME
      DIMENSION STRESS(NTENS),STATEV(NSTATV),DDSDDE(NTENS,NTENS),
     +          DDSDDT(NTENS),DRPLDE(NTENS),STRAN(NTENS),DSTRAN(NTENS),
     +          TIMEA(2),PREDEF(1),DPRED(1),PROPS(NPROPS),COORDS(3),
     +          DROT(3,3),DFGRD0(3,3),DFGRD1(3,3)
!-----------------------------------------------------------------------
!     Call UMAT_MODEL
!-----------------------------------------------------------------------
      call UMAT_MODEL(DDSDDE,STRESS,STATEV,DSTRAN,PROPS,
     +                KINC,NTENS,NSTATV,NPROPS)
!-----------------------------------------------------------------------
!     End of subroutine
!-----------------------------------------------------------------------
      return
      end
```

(V)UMAT subroutine

ABAQUS/explicit

- Include a subroutine with the code
- Send data to the subroutine

```
      INCLUDE './UMAT_MODEL.f'
!-----------------------------------------------------------------------
       SUBROUTINE VUMAT(
     + NBLOCK, NDIR, NSHR, NSTATEV, NFIELDV, NPROPS, LANNEAL, STEPTIME,
     + TOTALTIME, DT, CMNAME, COORDMP, CHARLENGTH, PROPS, DENSITY,
     + STRAININC, RELSPININC, TEMPOLD, STRETCHOLD, DEFGRAdoLD, FIELdoLD,
     + STRESSOLD, STATEOLD, ENERINTERNOLD, ENERINELASOLD, TEMPNEW,
     + STRETCHNEW, DEFGRADNEW, FIELDNEW,
     + STRESSNEW, STATENEW, ENERINTERNNEW, ENERINELASNEW)
C                                    :
                                     :
                                     :
!-----------------------------------------------------------------------
c           GRAB HISTORY VARIABLES
!-----------------------------------------------------------------------
              do k=1,NSTATEV
                 STATEV(k) = STATEOLD(i,k)
              enddo
!-----------------------------------------------------------------------
c           CALL UMAT_MODEL
!-----------------------------------------------------------------------
              call UMAT_MODEL(DDSDDE,STRESS,STATEV,DSTRAN,PROPS2,
     +                        KINC,NTENS,NSTATV,NPROPS)
!-----------------------------------------------------------------------
c           UNPACK STRESSES
!-----------------------------------------------------------------------
              STRESSNEW(i,1) = STRESS(1)
              STRESSNEW(i,2) = STRESS(2)
              STRESSNEW(i,3) = STRESS(3)
              STRESSNEW(i,4) = STRESS(4)
              if(NSHR.eq.3)then
                 STRESSNEW(i,6) = STRESS(5)
                 STRESSNEW(i,5) = STRESS(6)
              endif
!-----------------------------------------------------------------------
```

- Vectorized to scalar
- Tensor ordering

```
!-----------------------------------
GRAB STRAININC
!-----------------------------------
DSTRAN(1) = STRAININC(i,1)
DSTRAN(2) = STRAININC(i,2)
DSTRAN(3) = STRAININC(i,3)
DSTRAN(4) = 2.0*STRAININC(i,4)
if(NSHR.eq.3)then
   DSTRAN(5) = 2.0*STRAININC(i,6)
   DSTRAN(6) = 2.0*STRAININC(i,5)
endif
!-----------------------------------
```

(V)UMAT subroutine

The same material card input can be used between ABAQUS/Standard and ABAQUS/Explicit

ABAQUS/explicit

```
**----------------------------------------------------------
** MATERIALS
**----------------------------------------------------------
*material,name=EXAMPLE_VUMAT
*density
7.8e-9
*user material, CONSTANTS=24
**       E,       NU, BLANK,   BLANK, BLANK, BLANK,   BLANK, BLANK
 210000.0,      0.3,   0.0,     0.0,   0.0,   0.0,     0.0,   0.0
** SIGMA0,       T1,    Q1,      T2,    Q2,    T3,      Q3, BLANK
    250.0, 10000.0,  10.0, 1000.0, 100.0, 100.0, 1000.0,   0.0
**      WC,    DCRIT, BLANK,   BLANK, BLANK, BLANK,   BLANK, BLANK
    100.0,      1.0
*depvar, delete=3
3
1, P, "Equivalent plastic strain"
2, D, "Damage"
3,STATUS, "Status variable"
**----------------------------------------------------------
```

ABAQUS/standard

```
**----------------------------------------------------------
** MATERIALS
**----------------------------------------------------------
*material,name=EXAMPLE_UMAT
*user material, CONSTANTS=24
**       E,       NU, BLANK,   BLANK, BLANK, BLANK,   BLANK, BLANK
 210000.0,      0.3,   0.0,     0.0,   0.0,   0.0,     0.0,   0.0
** SIGMA0,       T1,    Q1,      T2,    Q2,    T3,      Q3, BLANK
    250.0, 10000.0,  10.0, 1000.0, 100.0, 100.0, 1000.0,   0.0
**      WC,    DCRIT, BLANK,   BLANK, BLANK, BLANK,   BLANK, BLANK
    100.0,      1.0
*depvar, delete=3
3
1, P, "Equivalent plastic strain"
2, D, "Damage"
3,STATUS, "Status variable"
**----------------------------------------------------------
```

(V)UMAT subroutine

In this example, we want to develop an elasto-plastic model with a ductile fracture model.

Hypo-elasticity:

$$\hat{\mathbf{D}} = \hat{\mathbf{D}}_e + \hat{\mathbf{D}}_p$$

Isotropic elastic:

$$\dot{\hat{\boldsymbol{\sigma}}} = \frac{E}{1+\nu}\hat{\mathbf{D}}_e^{'} + \frac{E}{3\,(1-2\nu)}\mathrm{tr}\hat{\mathbf{D}}_e\mathbf{I}$$

Yield function:

$$f = \sigma_{eq} - \sigma_y$$

Associated plastic flow:

$$\hat{\mathbf{D}}_p = \dot{\lambda}\frac{\partial f}{\partial \hat{\boldsymbol{\sigma}}}$$

Equivalent stress:

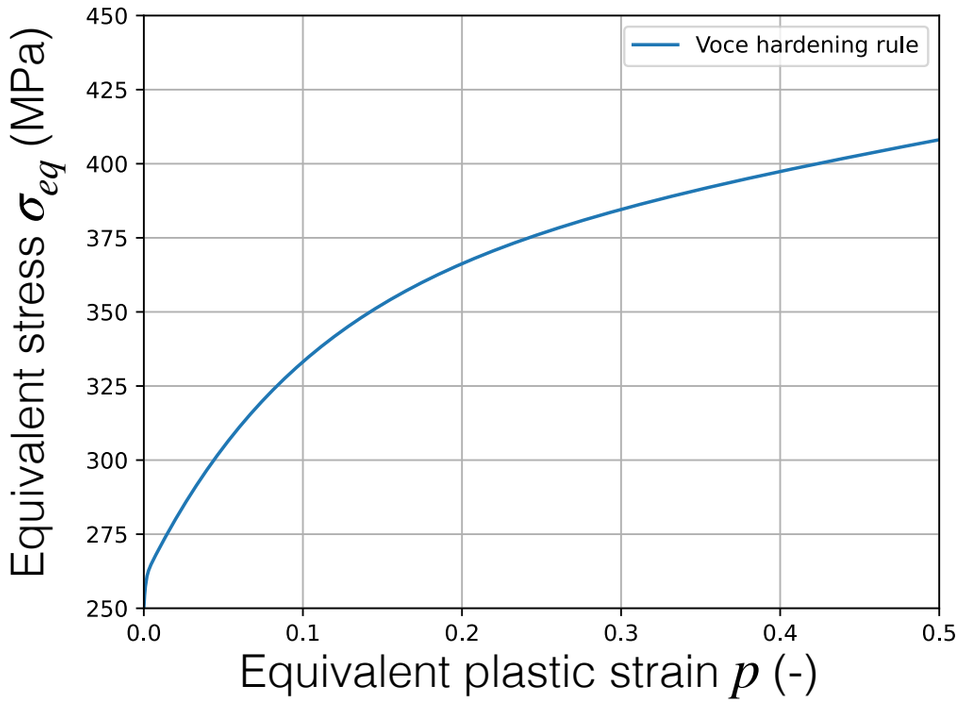
$$\sigma_{eq} = \sqrt{3J_2}$$

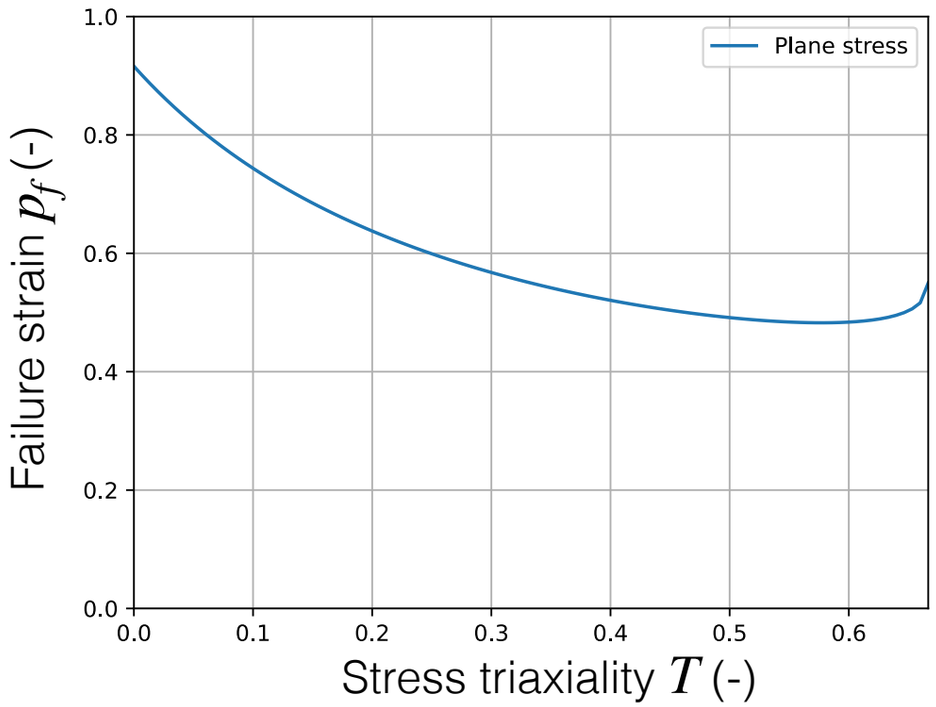
Isotropic hardening:

$$\sigma_y = \sigma_0 + \sum_{i=1}^{3} Q_i\left(1 - \exp\left(-\frac{\theta_i}{Q_i}p\right)\right)$$

Damage indicator model:

$$D = \int_0^{p_f} \frac{\langle\sigma_1\rangle}{W_c}dp \leq D_c$$





(V)UMAT subroutine

Computational plasticity:

1) Elastic prediction

$$\hat{\boldsymbol{\sigma}}^{TRIAL} = \hat{\boldsymbol{\sigma}}^{t} + \hat{\mathbf{C}}_e : \hat{\mathbf{D}}^{(t+\Delta t)} \Delta t$$

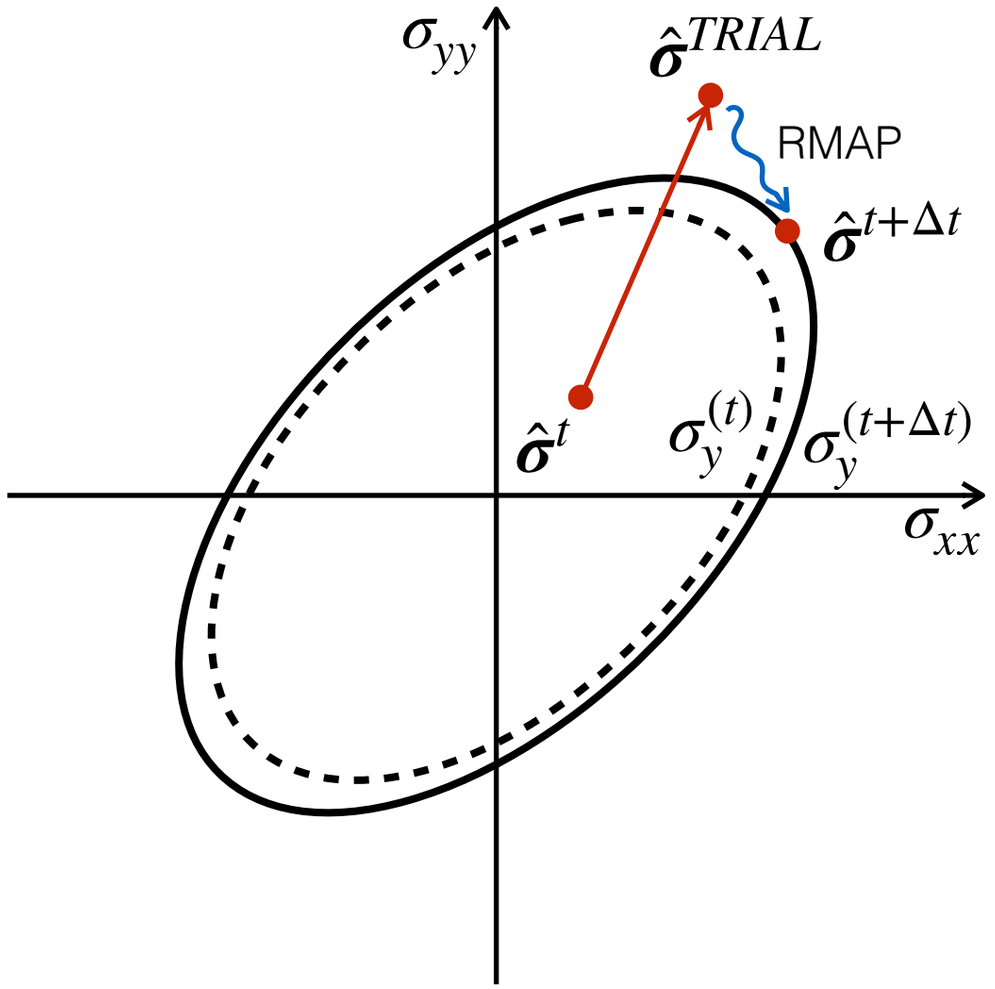
2) Compute yield function

$$f = \sigma_{eq}\left(\hat{\boldsymbol{\sigma}}^{TRIAL}\right) - \sigma_y^{(t)}$$

3) Check yield function

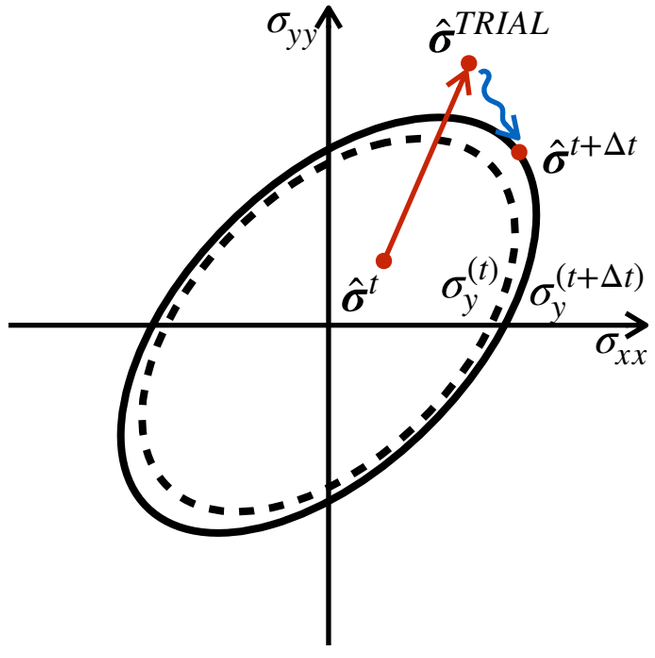
If $f < 0 \longrightarrow \hat{\boldsymbol{\sigma}}^{t+\Delta t} = \hat{\boldsymbol{\sigma}}^{TRIAL}$

If $f \geq 0 \longrightarrow$ Solve the non-linear system of equations (RMAP)

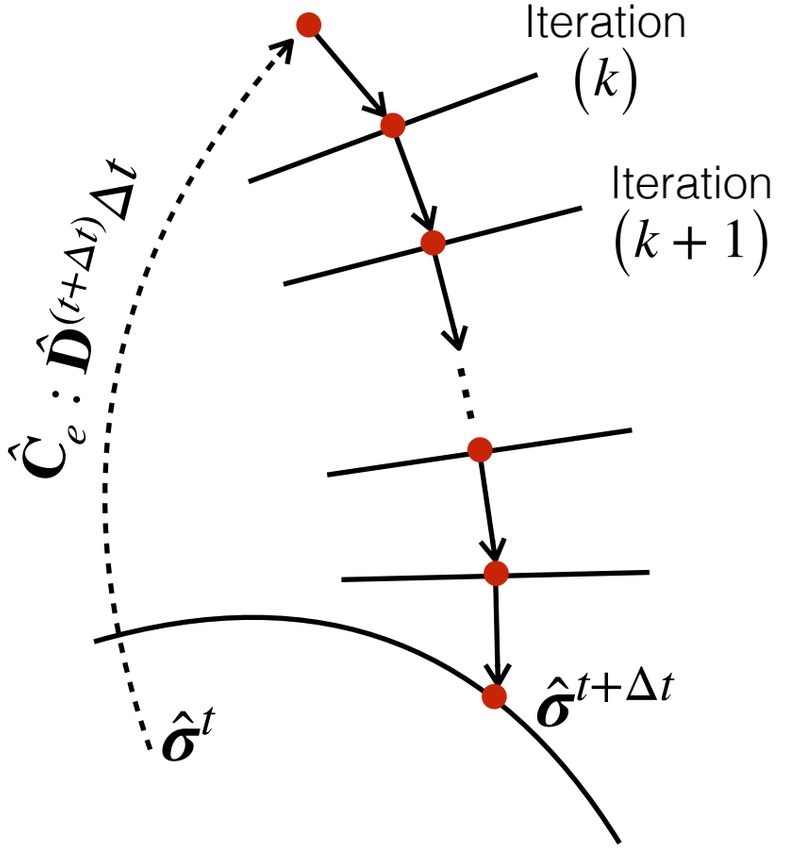


(V)UMAT subroutine

Return Map Algorithm (RMAP):



Cutting-plane algorithm:



Linearisation of the yield surface: $f\left(\hat{\boldsymbol{\sigma}}\right)|_{k+1} = 0$

$$\delta\lambda^{(k+1)} = \frac{f\left(\hat{\boldsymbol{\sigma}}\right)\Big|_k}{\frac{\partial f}{\partial \hat{\boldsymbol{\sigma}}}\Big|_k : \mathbf{C}_e : \frac{\partial f}{\partial \hat{\boldsymbol{\sigma}}}\Big|_k - \frac{\partial f}{\partial \sigma_y}\Big|_k \frac{\partial \sigma_y}{\partial \lambda}\Big|_k}$$

$$\hat{\boldsymbol{\sigma}}^{(k+1)} = \hat{\boldsymbol{\sigma}}^{(k)} - \delta\lambda \hat{\mathbf{C}}_e : \frac{\partial f}{\partial \hat{\boldsymbol{\sigma}}}$$

$$\boldsymbol{\sigma}_y^{(k+1)} = \boldsymbol{\sigma}_y^{(k)} - \delta\lambda \frac{\partial f}{\partial \lambda}$$

(V)UMAT subroutine

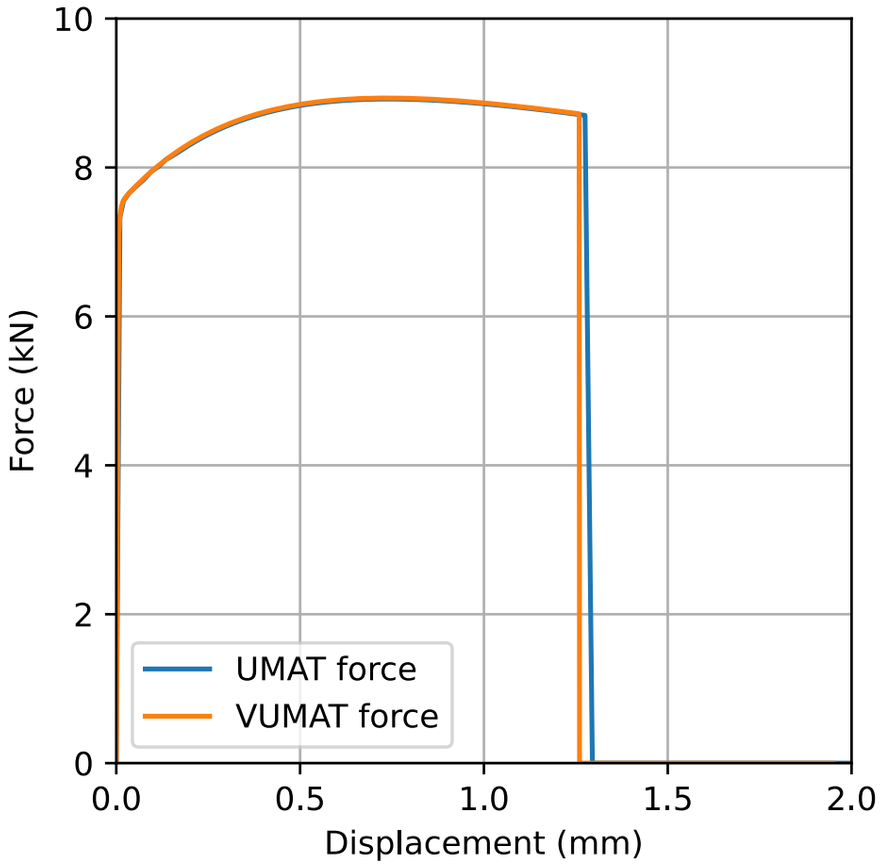
UMAT_MODEL.f quick description:

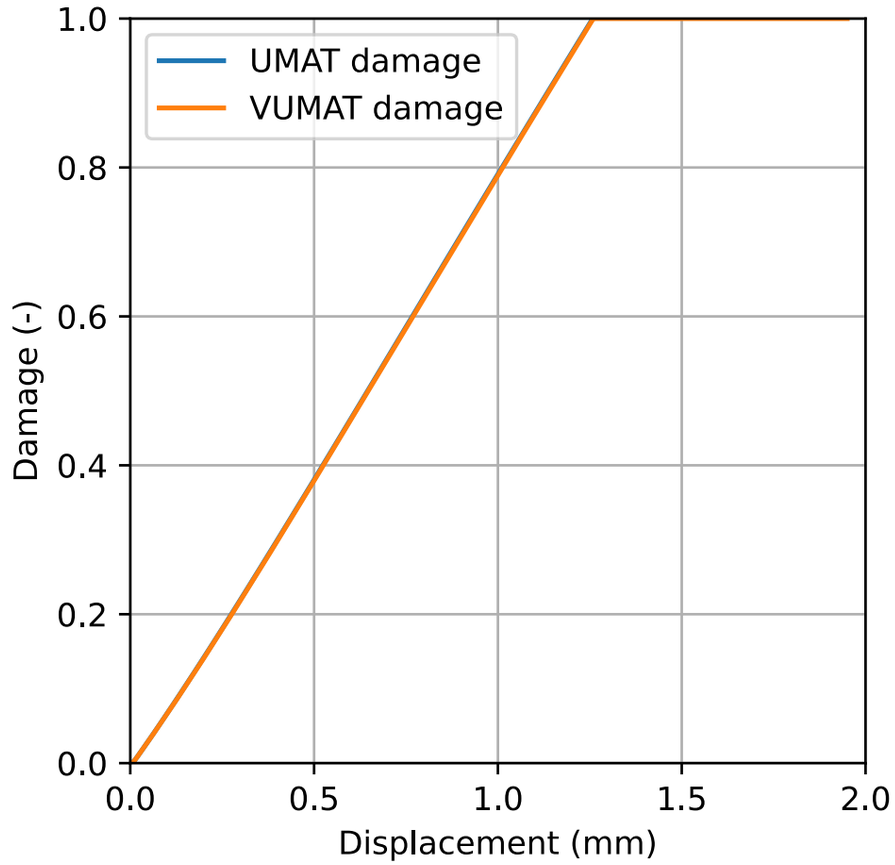
```
!-----------------------------------------------------
!-----Declaration internal variables
!-----------------------------------------------------
      integer i,j
!DECLARE -- Strain increments
      real*8 d1,d2,d3,d4,d5,d6
!DECLARE -- Stress tensor components
      real*8 s1,s2,s3,s4,s5,s6
!DECLARE -- Old stress tensor components
      real*8 s1old,s2old,s3old,s4old,s5old,s6old
!DECLARE -- Trial stress tensor components
      real*8 t1,t2,t3,t4,t5,t6
!DECLARE -- Equivalent plastic strain
      real*8 p,pold
!DECLARE -- Elasticity constants
      real*8 E0,NU
!DECLARE -- Elasticity matrix
      real*8 Ce(6,6)
!DECLARE -- Yield surface parameters
      real*8 SIGMAY,dSIGMAYdp
!DECLARE -- Yield function
      real*8 YF
!DECLARE -- Equivalent stress
      real*8 PHI
!DECLARE -- Yield function derivatives
      real*8 dfds1,dfds2,dfds3,dfds4,dfds5,dfds6,con
      real*8 dyfds,dyfdL,dfdSY
!DECLARE -- Work-hardening parameters
      real*8 SIGMA0,TR1,QR1,TR2,QR2,TR3,QR3
      real*8 T1oQ1,T2oQ2,T3oQ3
```

```
!-----------------------------------------------------------------
!     Extract history variables
!-----------------------------------------------------------------
      pold      = STATEV(1)
      damageold = STATEV(2)
!-----------------------------------------------------------------
!     Compute elastic prediction
!-----------------------------------------------------------------
      t1 = s1old+(Ce(1,1)*d1+Ce(1,2)*d2+Ce(1,3)*d3)
      t2 = s2old+(Ce(2,1)*d1+Ce(2,2)*d2+Ce(2,3)*d3)
      t3 = s3old+(Ce(3,1)*d1+Ce(3,2)*d2+Ce(3,3)*d3)
      t4 = s4old+(Ce(4,4)*d4)
      t5 = s5old+(Ce(5,5)*d5)
      t6 = s6old+(Ce(6,6)*d6)
!-----------------------------------------------------------------
!     Compute equivalent stress
!-----------------------------------------------------------------
      PHI = (t1*t1+t2*t2+t3*t3-t1*t2-t2*t3-t3*t1
     +     +3.0*(t4*t4+t5*t5+t6*t6))
      if(PHI.gt.0.0)then
         PHI = sqrt(PHI)
      else
         PHI = 0.0
      endif
!-----------------------------------------------------------------
!     Compute Yield stress
!-----------------------------------------------------------------
      SIGMAY = SIGMA0+QR1*(1.0-exp(-T1oQ1*pold))
     +                +QR2*(1.0-exp(-T2oQ2*pold))
     +                +QR3*(1.0-exp(-T3oQ3*pold))
!-----------------------------------------------------------------
!     Compute Yield function
!-----------------------------------------------------------------
      YF = PHI-SIGMAY
!-----------------------------------------------------------------
!     CHECK FOR PLASTICITY
!-----------------------------------------------------------------
      IF(YF.lt.0.0)THEN! ELASTICITY
```

(V)UMAT subroutine

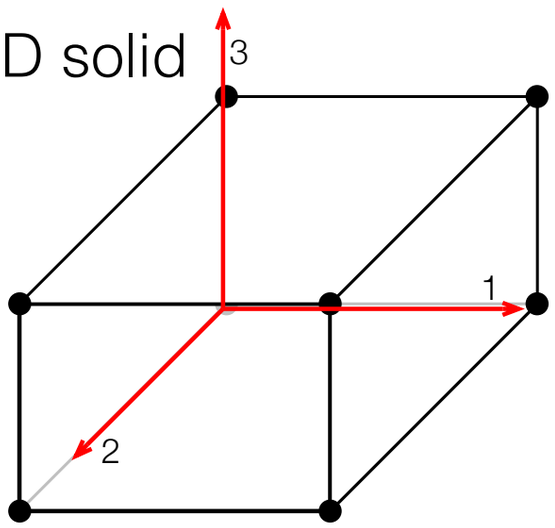
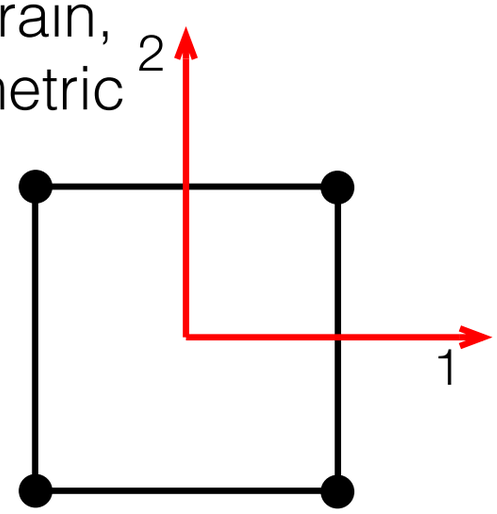
Results from the UMAT and VUMAT subroutines





(V)UMAT subroutine

The UMAT and VUMAT subroutines should work for:

<table>
<tr>
<td></td>
<td>3D solid
 
</td>
<td>2D plane strain,
2D axisymmetric
 
</td>
</tr>
<tr>
<td></td>
<td>Stress tensor:
$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{12} & \sigma_{22} & \sigma_{23} \\ \sigma_{23} & \sigma_{13} & \sigma_{33} \end{bmatrix}$$</td>
<td>Stress tensor:
$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22} \end{bmatrix}$$</td>
</tr>
<tr>
<td>VUMAT</td>
<td>

```
STRESS(1) = STRESSOLD(i,1)
STRESS(2) = STRESSOLD(i,2)
STRESS(3) = STRESSOLD(i,3)
STRESS(4) = STRESSOLD(i,4)
if(NSHR.eq.3)then
   STRESS(5) = STRESSOLD(i,6)
   STRESS(6) = STRESSOLD(i,5)
endif
```

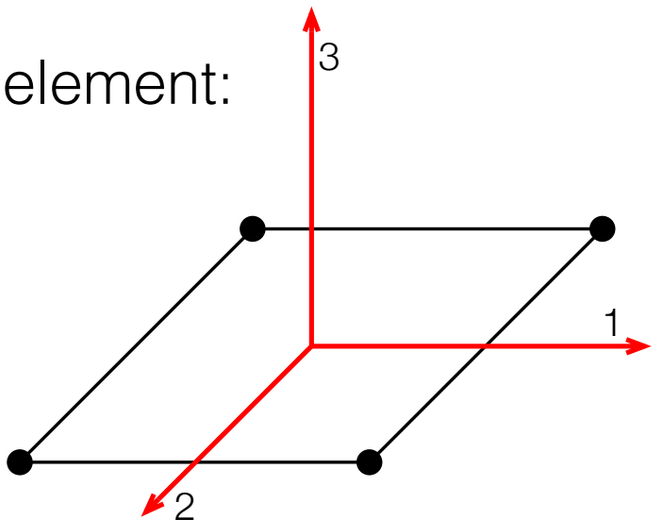
</td>
<td>

```
DSTRAN(1) = STRAININC(i,1)
DSTRAN(2) = STRAININC(i,2)
DSTRAN(3) = STRAININC(i,3)
DSTRAN(4) = 2.0*STRAININC(i,4)
if(NSHR.eq.3)then
   DSTRAN(5) = 2.0*STRAININC(i,6)
   DSTRAN(6) = 2.0*STRAININC(i,5)
endif
```

</td>
</tr>
</table>

(V)UMAT subroutine

Shell element:



Plane stress hypothesis:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma^e_{13} \\ \sigma_{12} & \sigma_{22} & \sigma^e_{23} \\ \sigma^e_{23} & \sigma^e_{13} & 0 \end{bmatrix}$$

Through-thickness shear stresses handled externally as elastic

```
**----------------------------------------------------------
*material,name=EXAMPLE_VUMAT_SHELL
*density
7.8e-9
*user material, CONSTANTS=24
**       E,        NU, BLANK,  BLANK, BLANK, BLANK,  BLANK, BLANK
 210000.0,       0.3,   0.0,    0.0,   0.0,   0.0,    0.0,   0.0
** SIGMA0,       T1,     Q1,     T2,     Q2,     T3,     Q3, BLANK
    250.0, 10000.0,   10.0, 1000.0, 100.0, 100.0, 1000.0,   0.0
**      WC,    DCRIT, BLANK,  BLANK, BLANK, BLANK,  BLANK, BLANK
    250.0,      1.0
*Transverse Shear
6730.7,6730.7, 0
*depvar, delete=3
3
1, P, "Equivalent plastic strain"
2, D, "Damage"
3,STATUS, "Status variable"
**----------------------------------------------------------
```

Transverse shear stiffnesses: K_{11}, K_{22}, K_{12}

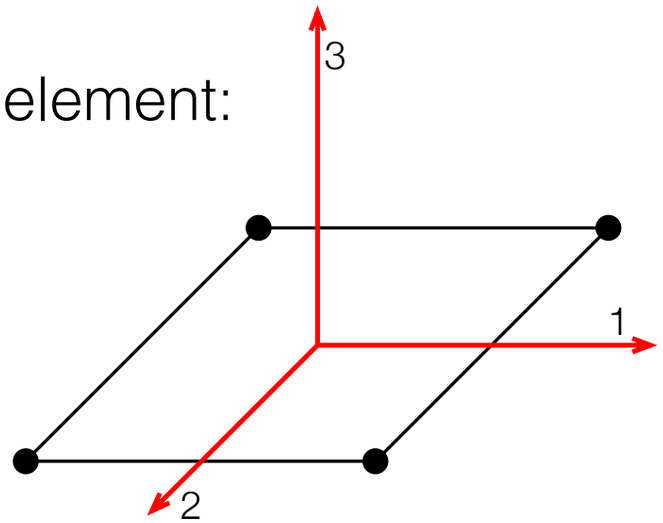
$$K_{11} = K_{22} = \frac{5}{6} G t_0$$

where G is the shear modulus and t_0 is the initial thickness of the shell element.

$$K_{12} = 0$$

(V)UMAT subroutine

Shell element:



Plane stress hypothesis:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma^e_{13} \\ \sigma_{12} & \sigma_{22} & \sigma^e_{23} \\ \sigma^e_{23} & \sigma^e_{13} & \boxed{0} \end{bmatrix}$$

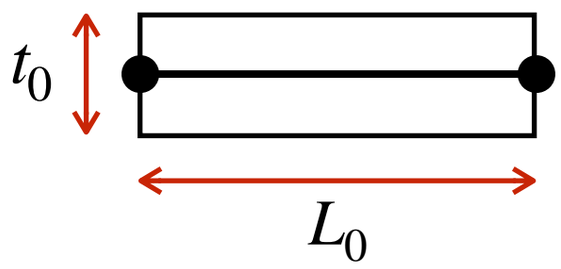
Through-thickness stress $\sigma_{33} = 0$

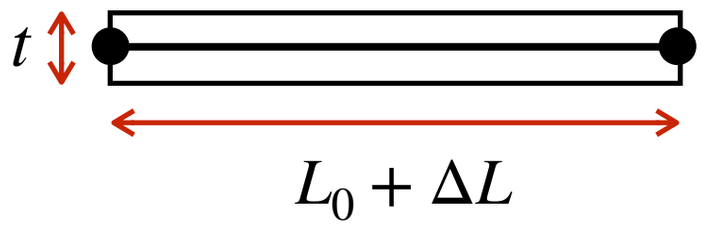
Plane stress hypothesis: $\sigma_{33} = 0$

```
!-----------------------------------
c          UNPACK STRESSES
!-----------------------------------
          STRESSNEW(i,1) = STRESS(1)
          STRESSNEW(i,2) = STRESS(2)
          STRESSNEW(i,3) = 0.0
          STRESSNEW(i,4) = STRESS(4)
          STRAININC(i,3) = DSTRAN(3)
!-----------------------------------
```

Thickness change: $\Delta\varepsilon_{33}$

Thickness change:

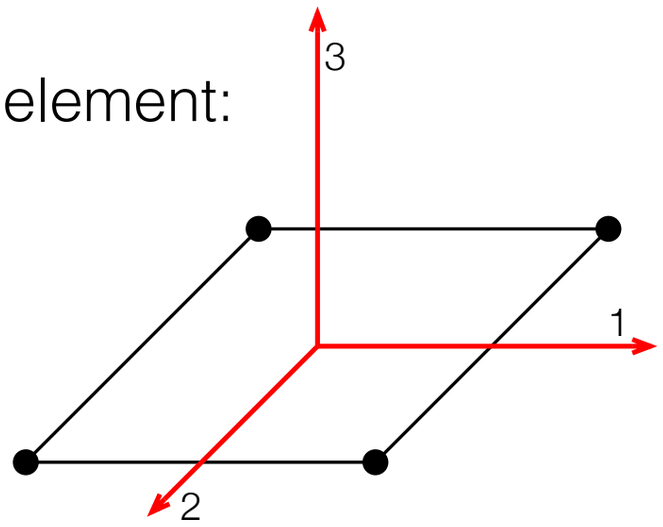




$$t = t_0 \exp\left(\varepsilon_{33}\right)$$

(V)UMAT subroutine

Shell element:



$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma^e_{13} \\ \sigma_{12} & \sigma_{22} & \sigma^e_{23} \\ \sigma^e_{23} & \sigma^e_{13} & 0 \end{bmatrix}$$

Through-thickness stress $\sigma_{33} = 0$

Enforcing plane stress in a VUMAT:

1. Reformulate all constitutive equations according to plane stress hypothesis
2. Use 3D constitutive equations and iterate to enforce plane stress hypothesis

(V)UMAT subroutine

2 Use 3D constitutive equations and iterate to enforce plane stress hypothesis

Iteration 1

Assume elastic response

$$\Delta\varepsilon_{33} = -\frac{\nu}{1-\nu}\left(\Delta\varepsilon_{11} + \Delta\varepsilon_{22}\right)$$

Compute $\boldsymbol{\sigma}^{t+\Delta t}$, store $\Delta\varepsilon_{33}$ and σ_{33}

Iteration 2

Assume plastic incompressibility

$$\Delta\varepsilon_{33} = -\left(\Delta\varepsilon_{11} + \Delta\varepsilon_{22}\right)$$

Compute $\boldsymbol{\sigma}^{t+\Delta t}$, store $\Delta\varepsilon_{33}$ and σ_{33}

Iteration n

Use a secant update for $\Delta\varepsilon_{33}$

$$\Delta\varepsilon_{33}|_n = \Delta\varepsilon_{33}|_{n-1} - H_t \,.\, \sigma_{33}|_{n-1}$$

$$H_t = \frac{\Delta\varepsilon_{33}|_{n-1} - \Delta\varepsilon_{33}|_{n-2}}{\sigma_{33}|_{n-1} - \sigma_{33}|_{n-2}}$$

$$\Delta\boldsymbol{\varepsilon} = \begin{bmatrix} \Delta\varepsilon_{11} & \Delta\varepsilon_{22} & \Delta\varepsilon_{33} & \Delta\varepsilon_{12} & 0 & 0 \end{bmatrix}^T$$

3D SUBROUTINE

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{22} & \sigma_{33} & \sigma_{12} & 0 & 0 \end{bmatrix}^T$$

If $|\sigma_{33}| > \xi\left(|\sigma_{11}| + |\sigma_{22}| + |\sigma_{12}|\right)$

ξ is a numerical tolerance ($1e^{-5}$)

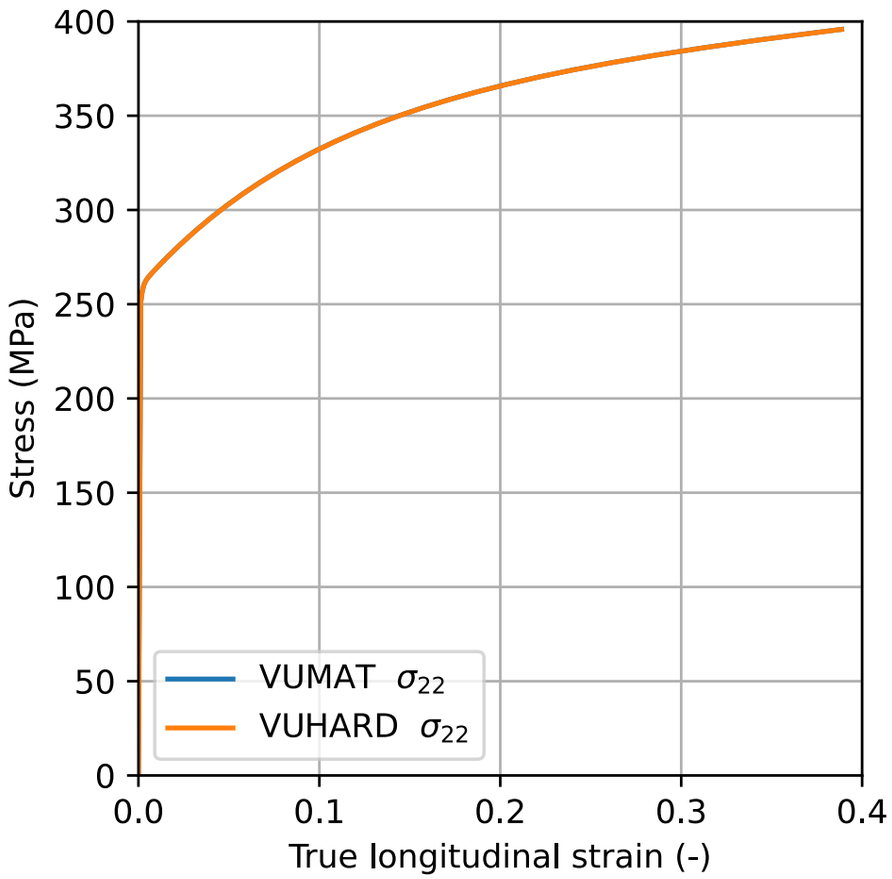
(V)UMAT subroutine

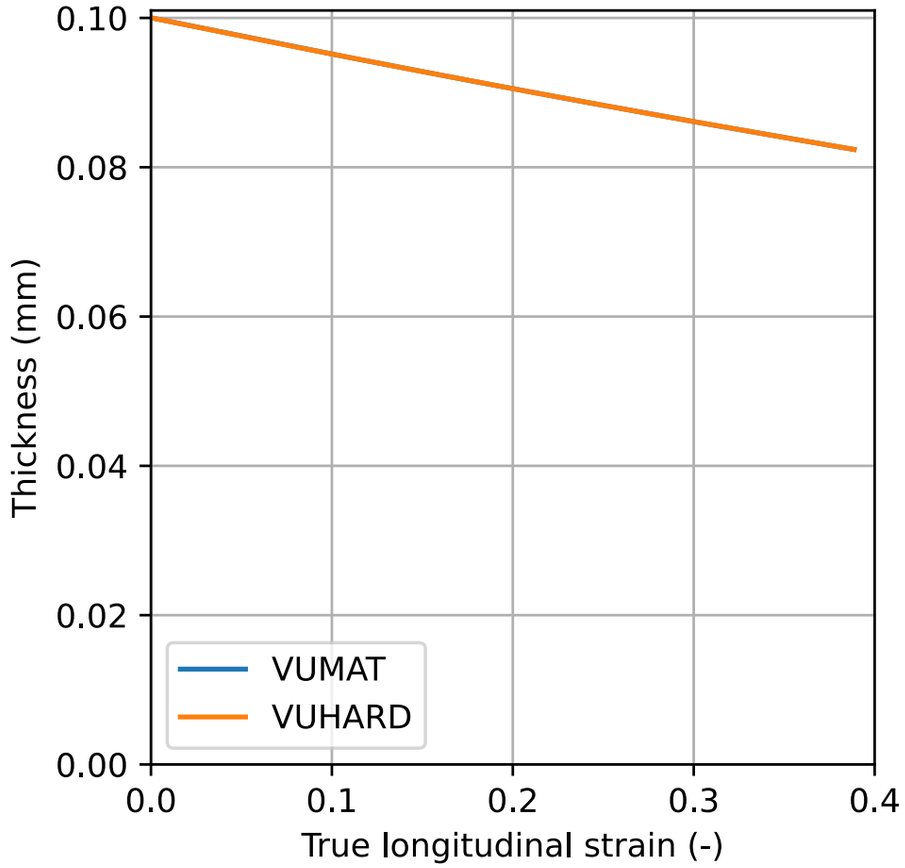
Enforcing plane stress in a VUMAT:

Method	1 Reformulate all constitutive equations according to plane stress hypothesis	2 Use 3D constitutive equations and iterate to enforce plane stress hypothesis
+	Computationally efficient	No need to change constitutive equations
−	Constitutive equations only valid for plane stress	Computational cost

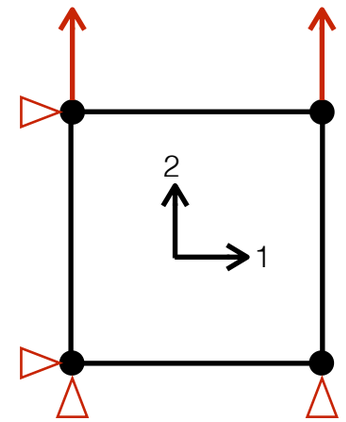
(V)UMAT subroutine

Using the VUMAT_SHELL.f subroutine and the VUHARD_V1.f subroutine:



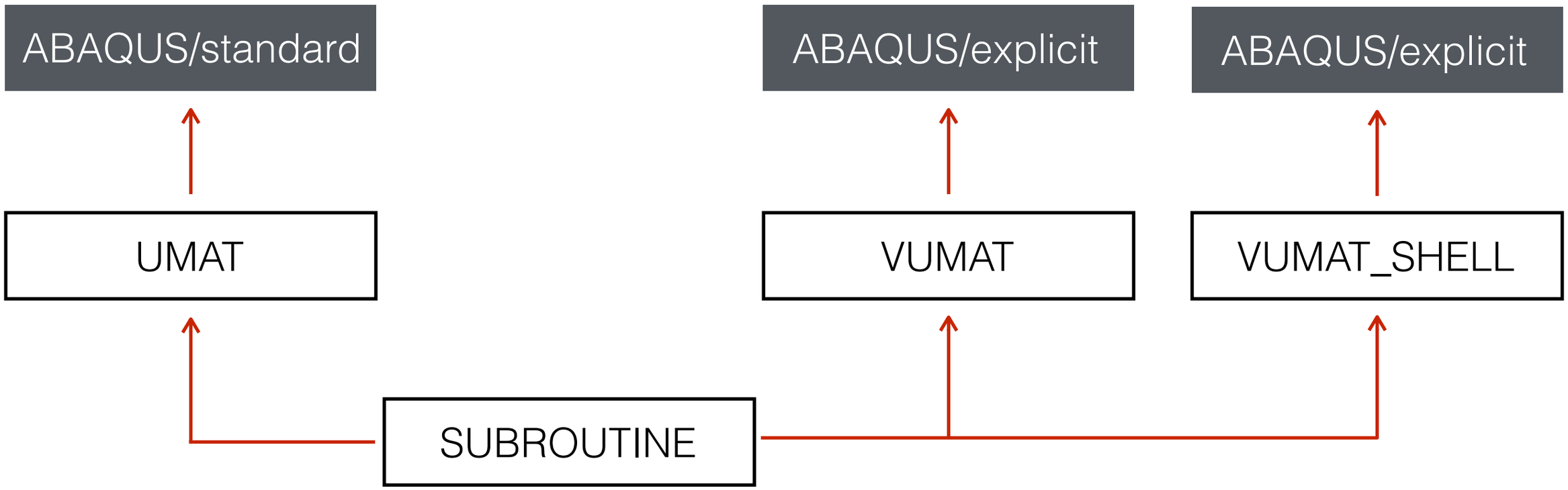


Single shell element
in uniaxial tension



(V)UMAT subroutine

A simplified approach to UMAT/VUMAT coding:



- Easier to debug
- A model available in both ABAQUS/standard and explicit
- Both plane-stress and full 3D with one code

- Less efficient code

(V)UEXTERNALDB subroutine

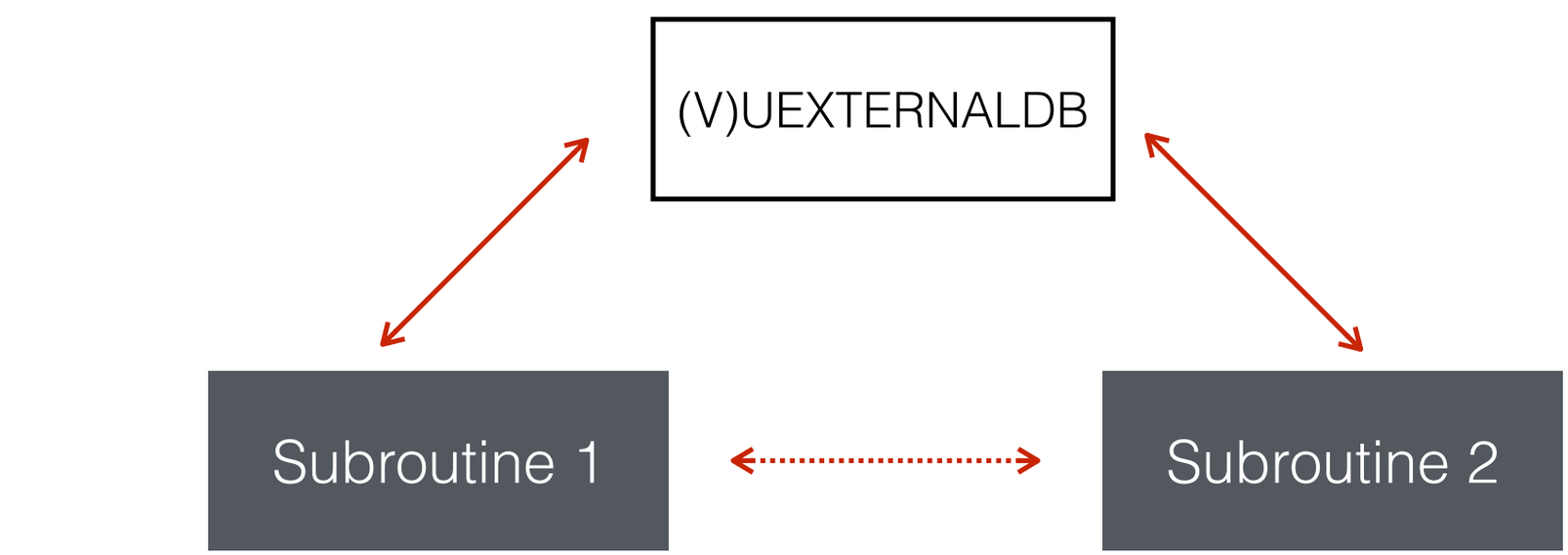
UEXTERNALDB: User subroutine to manage user-defined external databases and calculate model-independent history information.

VEXTERNALDB: User subroutine that gives control to the user at key moments of the analysis so that data can be exchanged dynamically among Abaqus user subroutines and with external programs or files.

(V)UEXTERNALDB subroutine

(V)UEXTERNALDB subroutines can be used to:

- Access external databases/files,
- Pass informations through the memory,
- Make computations which requires access to all integration points at once.



User subroutine **`UEXTERNALDB`**:

- is called once each at the beginning of the analysis, at the beginning of each increment, at the end of each increment, and at the end of the analysis (in addition, the user subroutine is also called once at the beginning of a restart analysis);

User subroutine **`VEXTERNALDB`**:

- is called once at the beginning of the analysis, at the beginning of each step, before each increment, at the start of each increment, at the end of each increment, at the end of each step, and finally at the end of the analysis;

(V)UEXTERNALDB subroutine

ABAQUS/standard

```
!-------------------------------------------
!     Start of the analysis
!-------------------------------------------
      if(LOP.eq.j_int_StartAnalysis)then
!-------------------------------------------
!     Start of the increment
!-------------------------------------------
      elseif(LOP.eq.j_int_StartIncrement)then
!-------------------------------------------
!     End of the increment
!-------------------------------------------
      elseif(LOP.eq.j_int_EndIncrement)then
!-------------------------------------------
!     End of the Analysis
!-------------------------------------------
      elseif(LOP.eq.j_int_EndAnalysis)then
      endif
!-------------------------------------------
!     End of subroutine
!-------------------------------------------
      RETURN
      END
```

ABAQUS/explicit

```
!-------------------------------------------
!     Start of the analysis
!-------------------------------------------
    if(lOp.eq.j_int_StartAnalysis)then
!-------------------------------------------
!     Setup of the increment
!-------------------------------------------
    elseif(lOp.eq.j_int_SetupIncrement)then
!-------------------------------------------
!     Start of the increment
!-------------------------------------------
    elseif(lOp.eq.j_int_StartIncrement)then
!-------------------------------------------
!     End of the increment
!-------------------------------------------
    elseif(lOp.eq.j_int_EndIncrement)then
!-------------------------------------------
!     End of the analysis
!-------------------------------------------
    elseif(lOp.eq.j_int_EndAnalysis)then
    endif
!-------------------------------------------
!     End of the subroutine
!-------------------------------------------
    return
    end
```

(V)UEXTERNALDB subroutine

ABAQUS/standard

```
!-----------------------------------------------------------------
!     Start of the analysis
!-----------------------------------------------------------------
      if(LOP.eq.j_int_StartAnalysis)then
        IF((kInc.eq.0).and.(myThreadID.eq.0))THEN
!-----------------------------------------------------------------
!          CREATE GLOBAL ARRAYS
!-----------------------------------------------------------------
          ptr_REAL = SMAFloatArrayCreate(ID_REAL, 24, 0.0)
```

Pointer

Pointer ID (unique number)

Size

Initial value

ABAQUS/explicit

```
!-----------------------------------------------------------------
!     Start of the analysis
!-----------------------------------------------------------------
      if(lOp.eq.j_int_StartAnalysis)then
        IF(kInc.eq.0)THEN
!-----------------------------------------------------------------
!          CREATE GLOBAL ARRAYS
!-----------------------------------------------------------------
          ptr_REAL = SMAFloatArrayCreate(ID_REAL, 24, 0.0)
```

Pointer

Pointer ID (unique number)

Size

Initial value

1 Create float (real) array in memory

Define array size

Link pointer

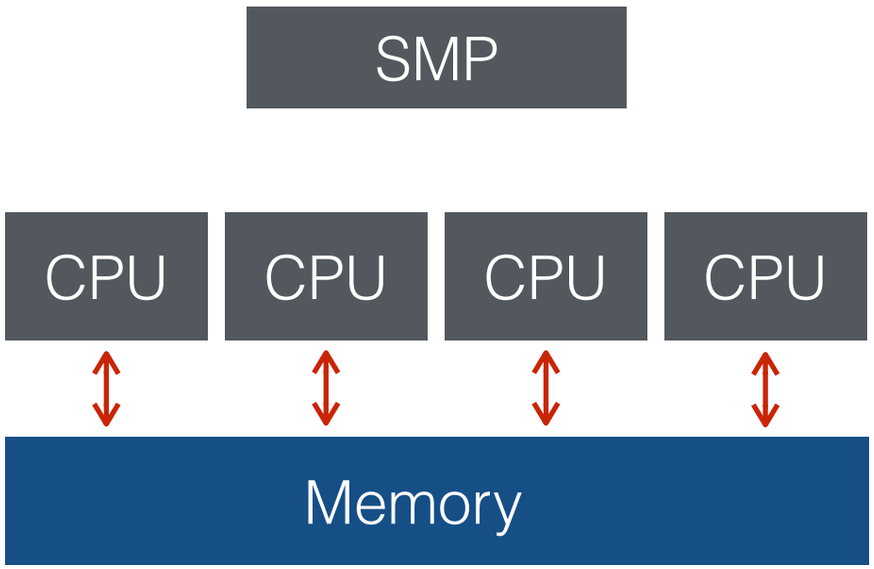
```
!-----Declare global variables
      REAL*8 PROPS_REAL(24)          ! ARRAY WITH REAL PROPERTIES
      integer ID_REAL                ! ID for pointers
      parameter(ID_REAL = 1)         ! ID for pointers
      pointer(ptr_REAL, PROPS_REAL) ! pointer link
!-----------------------------------------------------------------
```

Pointer ID

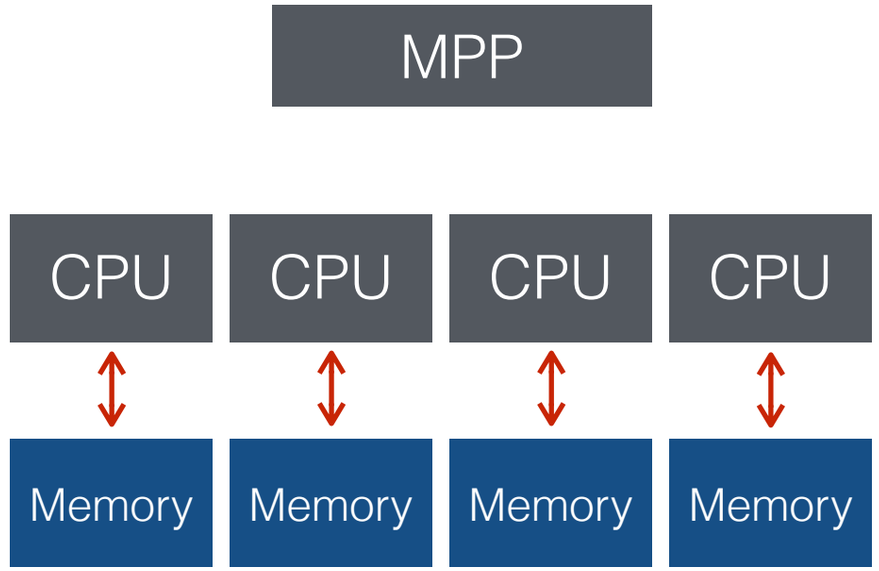
```
      ENDIF
!-----------------------------------------------------------------
```

```
      ENDIF
!-----------------------------------------------------------------
```

Interlude



CPUs share the same memory



CPUs have their own memory

+	CPUs have access to the same arrays	CPUs write in different arrays
−	CPUs have access to the same arrays	CPUs write in different arrays

Interlude

Should your code share the same arrays (for example non-local computations)?

	YES	NO

SMP	It will natively do it, global arrays will work,
	You still need to prevent the code to write at the same place, check UEXTERNALDB.f for example.
	`!---------- Get threadID ----------`
	`myThreadID = get_thread_id()`
	`IF((kInc.eq.0).and.(myThreadID.eq.0))THEN`
	Filter out threads
		You need to define local arrays via:
		`!---------- CREATE LOCAL ARRAYS ----------`
		`ptr_REAL = SMALocalFloatArrayCreate(ID_REAL, 24, 0.0)`
		`ptr_INT = SMALocalIntArrayCreate( ID_INT, 24, 0)`
		Local arrays

MPP	You need to use MPI (Message Passing Interface) commands
	
	Example:
	`call MPI_Bcast( N,size, MPI_INT,0, ABA_COMM_WORLD,ierr)`
	`call MPI_Bcast(W,size,MPI_DOUBLE_PRECISION,0, ABA_COMM_WORLD,ierr)`
		It will natively do it, global arrays will work

(V)UEL subroutine

UEL: **User subroutine to define an element.**

WARNING: This feature is intended for advanced users only. Its use in all but the simplest test examples will require considerable coding by the user/developer. "User-defined elements," Section 32.17.1 of the Abaqus Analysis User's Guide, should be read before proceeding.

VUEL: **User subroutine to define an element.**

(V)UEL subroutine

Overall principle:

(V)UEL subroutine

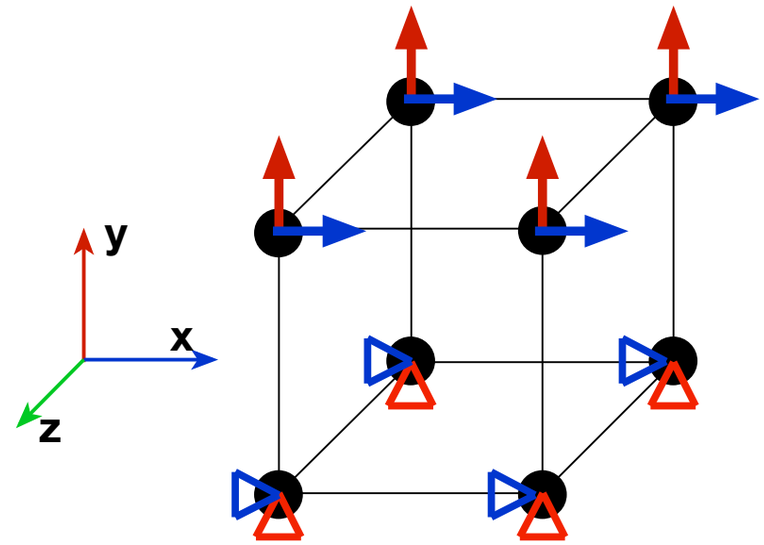
```
!-----------------------------------------------------------------------
!     Get Gauss points coordinates
!-----------------------------------------------------------------------
      call get_gauss_points(g,h,r,w,nint)
!-----------------------------------------------------------------------
!     Get shape functions
!-----------------------------------------------------------------------
      call get_shape_functions(N,g,h,r,nnode,nint,eltype)
!-----------------------------------------------------------------------
!     Get Jacobian functions
!-----------------------------------------------------------------------
      call get_jacobian_functions(J,detJ,g,h,r,position0,
     +                            nint,ncoord,nnode,eltype)
!-----------------------------------------------------------------------
!     Get derivatives of shape functions
!-----------------------------------------------------------------------
      call get_der_shape_functions(B,B2,J,g,h,r,
     +                             nnode,ncoord,nint,eltype)
!-----------------------------------------------------------------------
!     Compute strains in integration point
!-----------------------------------------------------------------------
      call compute_smallstrains(strain,B,U,
     +                          ndof,ncoord,nnode,nint,eltype)
!-----------------------------------------------------------------------
!     Compute strains in integration point
!-----------------------------------------------------------------------
      call compute_stresses(stress,CT,strain,dm,hisv,time,dt,
     +                      nint,nhisv,ndof,eltype)
!-----------------------------------------------------------------------
!     Update thickness
!-----------------------------------------------------------------------
      call update_thick(thick,strain,dm,
     +                  nint,ndof,ncoord,nnode,eltype)
!-----------------------------------------------------------------------
!     Compute Internal forces
!-----------------------------------------------------------------------
      call compute_Fint(Fint,B,stress,w,detJ,thick,
     +                  nint,ndof,ncoord,nnode,eltype)
!-----------------------------------------------------------------------
!     Compute Stiffness matrix
!-----------------------------------------------------------------------
      call compute_Kmatrix(Kmatrx,B,CT,w,DetJ,
     +                     nint,nnode,ncoord,ndof,eltype)
!-----------------------------------------------------------------------
!     Compute Mass matrix
!-----------------------------------------------------------------------
      call compute_Mmatrix(Mmatrx,rho,N,w,DetJ,
     +                     nint,nnode,ncoord,ndof,eltype)
!-----------------------------------------------------------------------
!     Compute Stable Time step
!-----------------------------------------------------------------------
      call compute_timestep(dtout,position,dm,
     +                      nnode,ncoord,ndof,eltype)
!-----------------------------------------------------------------------
```

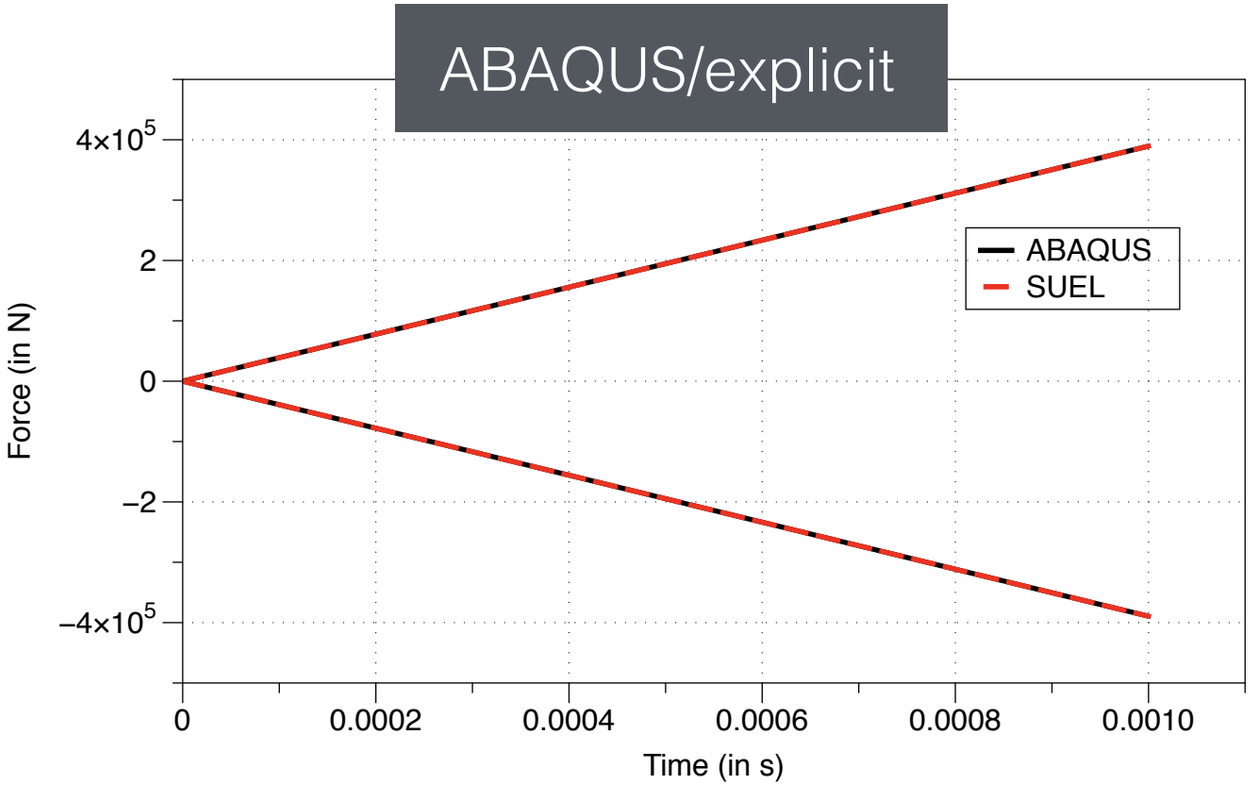
Example, 3D solid element and 2D plane strain/ stress, small-strain/displacement formulation

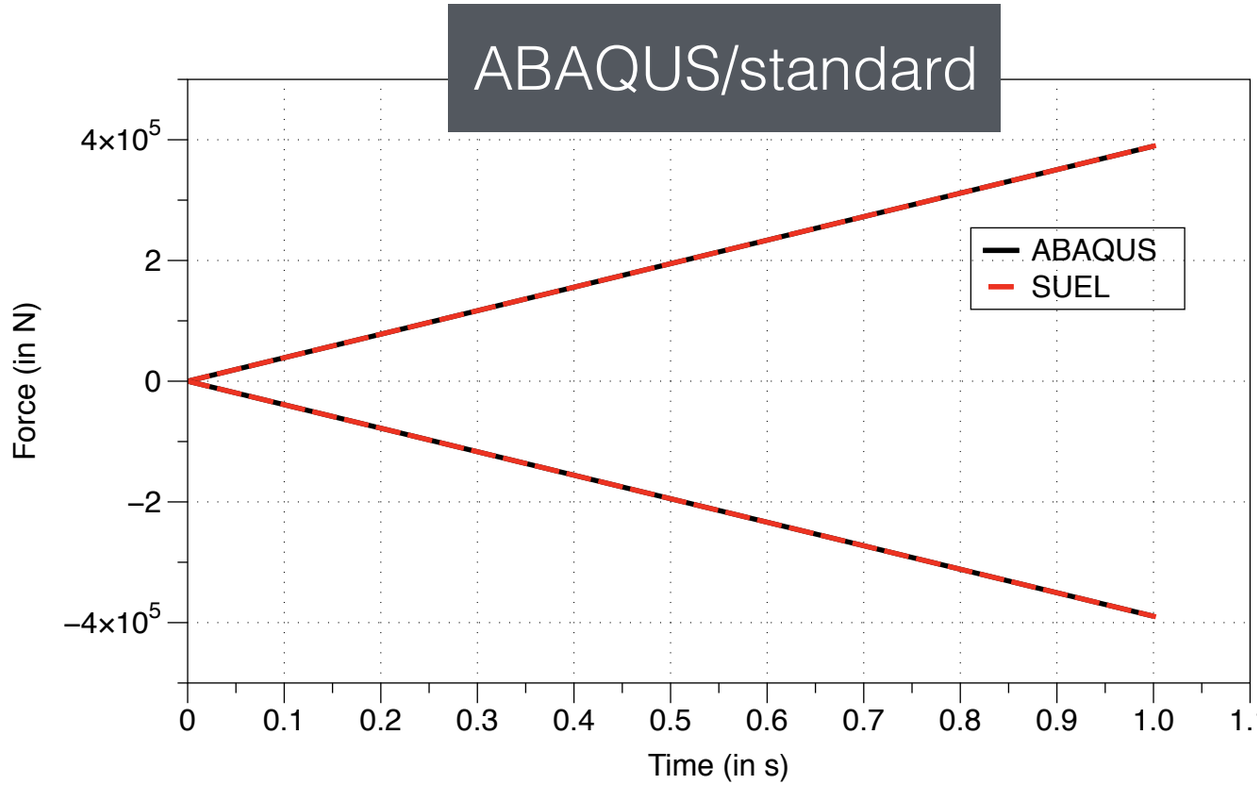
$$\mathbf{u}(g,h) = N^N(g,h)\mathbf{u}^N$$ Isoparametric element

(V)UEL subroutine

- Verification with an ABAQUS element under mixed mode deformation with an elastic material.
- UEL+VUEL script => 2200 lines of code…







Conclusions

Why should we develop/implement something into ABAQUS?

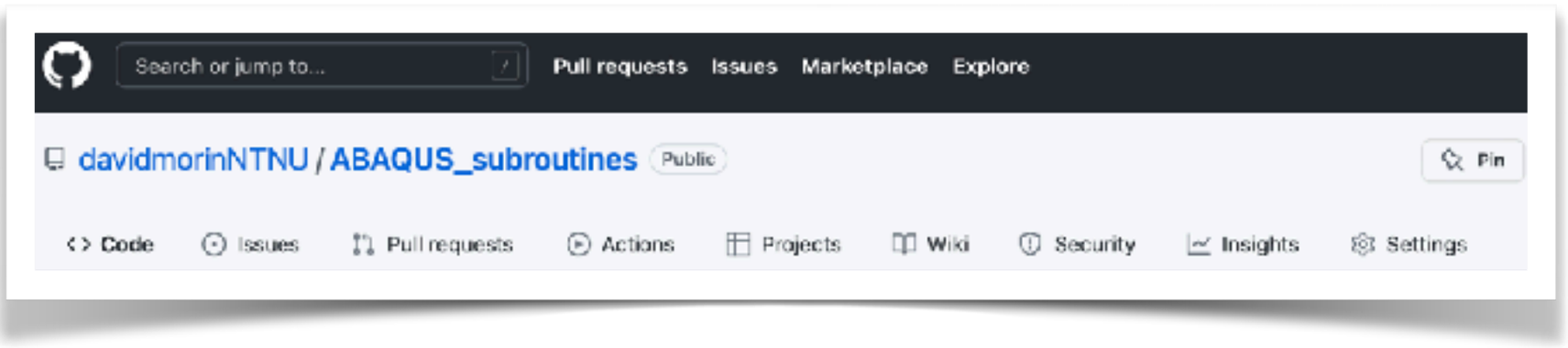
- Lack of a particular model
- Improvement of an existing model

Things to consider before implementing any model:

- Can you work with an existing model?
- Will someone else use your code?

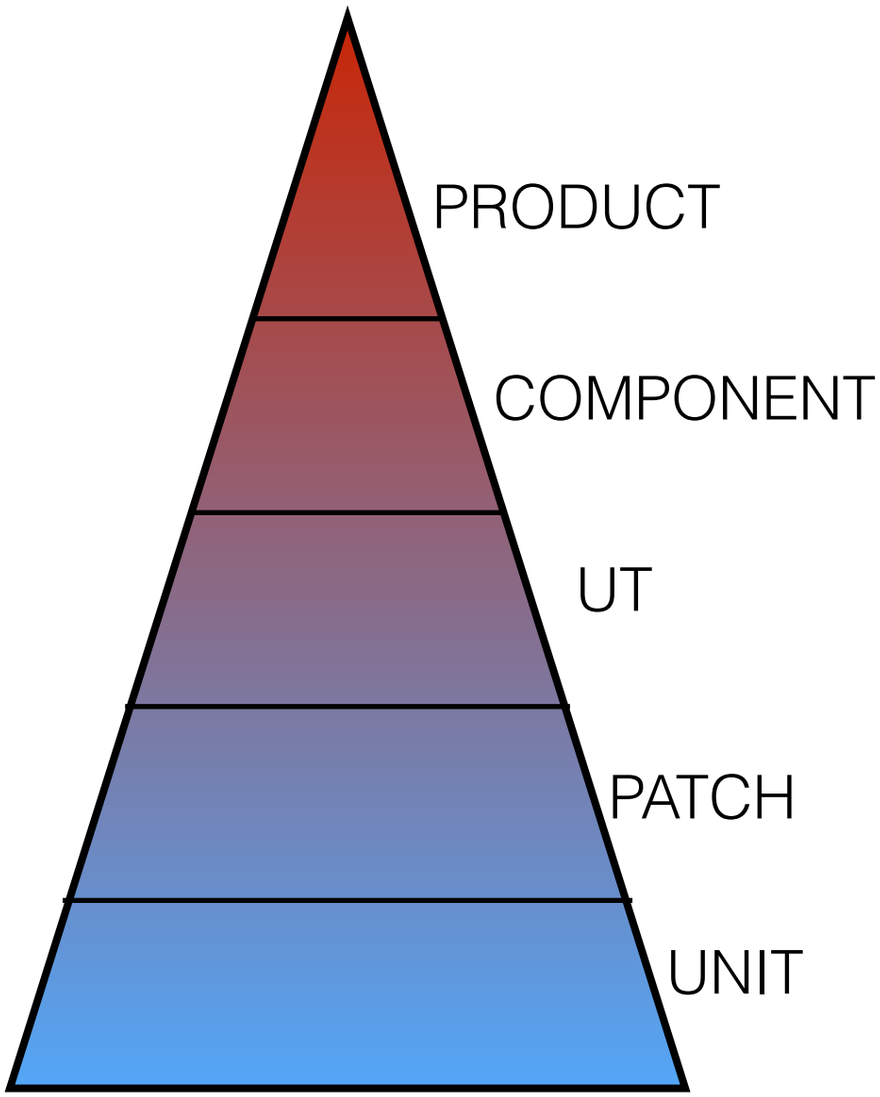
Use this as a starting point?

https://github.com/davidmorinNTNU/ABAQUS_subroutines

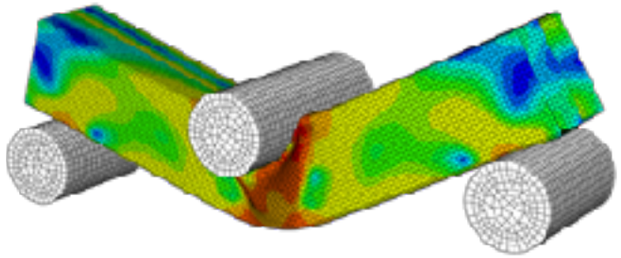


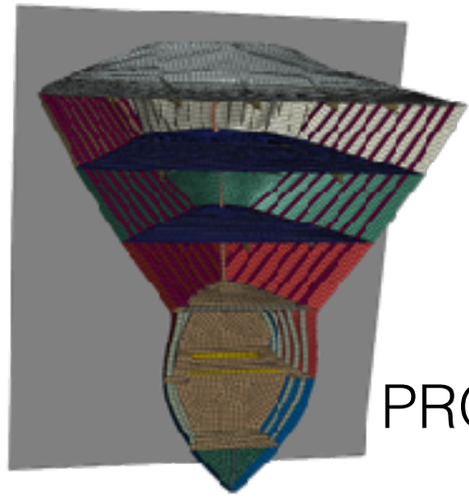
Conclusions

Verification of user-models:

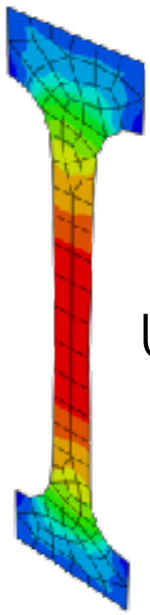


COMPONENT

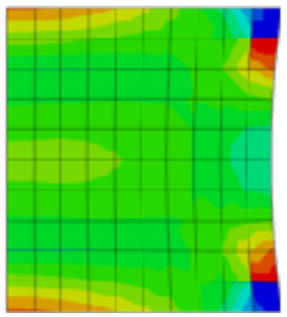




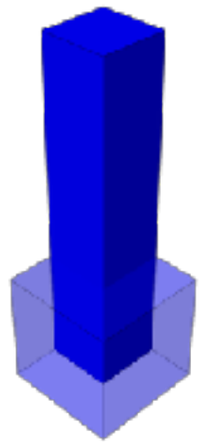
PRODUCT



UT



PATCH



UNIT

Thank you for your attention !