A user element for small and finite deformation for Abaqus

Nils Lange, Geralf Hütter

April 8, 2024

1 Description

The present package provides the implementation of elements for the UEL interface of Abaqus. The package is distributed under a CC BY-NC-SA 4.0 license. It is published together with MonolithFE^2. Please refer to [1] when using the package or any of its contents.

Features

- small deformations and finite deformations in Updated Lagrange formulation
- UMAT interface (equivalent to Abaqus) for user defined material behavior
- elastic-Mises-plastic material routine in rate formulation supplied
- plane stress, plane strain, axisymmetric and 3D-stress conditions
- Hardening Abaqus interface UHARD supported when Mises UMAT is being used
- UHARD for multilinear hardening supplied
- 3D, plane strain, plane stress and axisymmetric elements
- in small deformations the B Matrix and integration point volume is stored in Abaqus allocatable arrays, so they do not have to be computed in every timestep
- hyper ROM training data ist output to SVARS in the size of SVARS is larger than actually needed

Requirements

- UELlib (ABQinterface.f90, Math.f90, UEL_lib.f90 loaded in UEL.f)

2 Usage

Same usage as standard UELlib elements. In the Inputfile, specify the UEL as:

```
*USER ELEMENT, IPROPERTIES=0, PROPERTIES=%n_PROPS, TYPE=elementtype, NODES=%n_NODES,
 COORDINATES=%n_COORDS, VARIABLES=%n_SVARS
1, 2, 3   **or   1, 2    if 2D Elements are being used
```

Where n_{PROPS} is the number of Properties, which are supplied for the UMAT.

The hypoelastic-MISES-plastic UMAT, which calls the UHARD for multilinear Hardening expects the Properties in the following order:

```
*UEL PROPERTY
```
$\%E$, $\%\nu$, $\%Y_1$, $\%\varepsilon_1^{\mathrm{pl}}$, $\%Y_2$, $\%\varepsilon_2^{\mathrm{pl}}$, ...

Where E and ν are the elastic constants, Y_i is the yield stress and $\varepsilon_i^{\mathrm{pl}}$ the plastic strain. Only 8 properties per line are allowed.

The simulation is started with:

```
abqXXXX interactive job=Jobname user=bin/UEL-std
```

The program has to be precompiled with running

`make ABACALL=abqXXX MATERIAL=XXXX` (Linux)

`.\compile.bat` (Windows, a prompt asks for the Material Model)

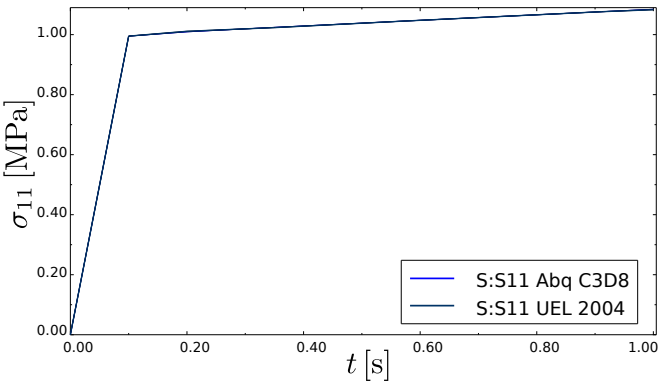
Note: The directive "!DEC$ FREEFORM" tells the Intel Fortran Compiler that the code is written in free-form Fortran. Recent version of Abaqus since 6.14 switch on the preprocessor by the option "-fpp" by default which is incompatible with the aforementioned compiler directive. For using the code with such recent versions, the option "-fpp" has to be removed in the Abaqus environment file or, alternatively, the compiler option "-free" has to be set.

3 Examples

Two Examples are provided. In the first example “Hex_uni_tension” and “Hex_uni_tension_UEL”, a 1mm×1mm×1mm one element cube is exposed to uniaxial tension without expansion prevention in the other two directions using finite kinematics with $\bar{u}_1(t) = 0.1mm \cdot t$. A elastic plastic material with $E = 100\,\mathrm{MPa}$, $\nu = 0.3$, $Y_1 = 1.0\,\mathrm{MPa}$, $\varepsilon_1^{\mathrm{pl}} = 0.0$, $Y_2 = 2.0\,\mathrm{MPa}$ and $\varepsilon_2^{\mathrm{pl}} = 1.0$ is being used. The resulting (true) cauchy stress σ_{11} is being compered with the Abaqus C3D8 implementation. Since only nodal UEL values are accessible in the Abaqus postprocessing, the stress is being calculated as:

$$\boldsymbol{\sigma} = J^{-1}\boldsymbol{P}\boldsymbol{F}^{\mathrm{T}} \tag{1}$$

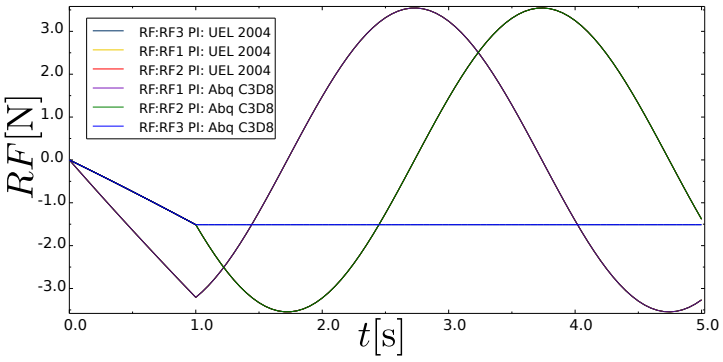
$$\sigma_{11}^{\mathrm{UEL}} = \frac{1+u_1}{(1+u_1)\cdot(1+u_2)\cdot(1+u_3)} \cdot \frac{RF_{\mathrm{total}}}{A_{\mathrm{nominal}}} \tag{2}$$



In the second example "Hex_uni_tension_rotation" and "Hex_uni_tension_rotation_UEL" the same cube (with elastic materialbehavior) is exposed to uniaxial tension with expansion prevention and then rotated 360° around the z-axis. The deformation gradient is:

$$\begin{aligned} \boldsymbol{F} &= \begin{bmatrix} 1.0 + 0.1\mathrm{s}^{-1} \cdot t & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} && 0.0\mathrm{s} \leqslant t \leqslant 1.0\mathrm{s} \\ \boldsymbol{F} &= \begin{bmatrix} 1.1\cos(\mathrm{s}^{-1}\frac{\pi}{2}t) & -\sin(\mathrm{s}^{-1}\frac{\pi}{2}t) & 0.0 \\ 1.1\sin(\mathrm{s}^{-1}\frac{\pi}{2}t) & \cos(\mathrm{s}^{-1}\frac{\pi}{2}t) & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} && 1.0\mathrm{s} < t \leqslant 4.0\mathrm{s} \end{aligned} \tag{3}$$

For comparison the reaction force of node 1 ($\boldsymbol{X} = \boldsymbol{0}$) is plotted using the Abaqus C3D8 and UEL 2004 Element.



4 Theory

For large deformations, the update Lagrange method is deployed. The element coordinates of the current configuration are computed as:

$$\underline{\underline{x}}^e(t) = \underline{\underline{X}}^e + \underline{\underline{u}}^e(t) \tag{4}$$

The B-matrix is a function of the current element coordinates:

$$\underline{\underline{B}}^e_t = \underline{\underline{B}}^e_t\left(\underline{\underline{x}}^e(t)\right) \tag{5}$$

The rate of deformation is calculated by the central difference formula after Hughes and Winget [2]:

$$\Delta\underline{D} = \mathrm{sym}\left(\frac{\partial \Delta\underline{u}}{\partial \underline{x}_{t+\Delta t/2}}\right) \tag{6}$$

with:

$$\underline{x}_{t+\Delta t/2} = \frac{1}{2}\cdot\left(\underline{x}_t + \underline{x}_{t+\Delta t}\right) \tag{7}$$

Written with B-Matrix:

$$\Delta\underline{D}^\alpha = \underline{\underline{B}}^\alpha_{t+\Delta t/2}\cdot\Delta\underline{u} \tag{8}$$

The stresses are rotated and updated as:

$$\underline{\sigma}_{t+\Delta t} = \Delta\underline{\underline{R}}\cdot\underline{\sigma}_t\cdot\Delta\underline{\underline{R}}^{\mathrm{T}} + \Delta\underline{\sigma}(\Delta\underline{D}) \tag{9}$$

where $\Delta\underline{\sigma}$ is to be defined by the UMAT as a function of $\Delta\underline{D}$. The rotation increment is calculated as:

$$\Delta\underline{\underline{R}} = (\underline{\underline{I}} - \frac{1}{2}\Delta\underline{\underline{W}})^{-1} \cdot (\underline{\underline{I}} + \frac{1}{2}\Delta\underline{\underline{W}}) \tag{10}$$

with

$$\Delta\underline{\underline{W}} = \text{asym}\left(\frac{\partial\Delta\underline{u}}{\partial\underline{x}_{t+\Delta t/2}}\right) \tag{11}$$

The vector of the internal forces is now calculated as:

$$\underline{f}^e_{\text{int}} = \sum_{\alpha=1}^{n_\alpha} w^\alpha \cdot \det(\underline{\underline{J}}^\alpha_t) \cdot \underline{\underline{B}}^{\alpha\,\text{T}}_t \cdot \underline{\sigma}^\alpha_{t+\Delta t} \tag{12}$$

The element stiffness matrix is computed as:

$$\underline{\underline{K}}^e = \sum_{\alpha=1}^{n_\alpha} w^\alpha \cdot \det(\underline{\underline{J}}^\alpha_t) \cdot \underline{\underline{B}}^{\alpha\,\text{T}}_t \cdot \underline{\underline{C}}^\alpha_{t+\Delta t} \cdot \underline{\underline{B}}^\alpha_t \tag{13}$$

Note that in this way the configuration of the last time step is chosen as the reference configuration.

The plane stress algorithm is a nested Newton algorithm after Dodds (see for details [3] p.362). It's worth mentioning, that a "monolithic" algorithm (as described in [3] p.367) could be computationally more efficient, but requires to "know" state variables from the last global Newton-Raphson loop, which is provided neither by the UMAT nor the UEL interface. A possible solution would be the use of optional FORTRAN 90 style arguments, so that values of the current NR increment could be provided by MonolithFE^2, while still being compatible with Abaqus. The nested Newton algorithm although being robust and efficient is sometimes leading to convergence problems at the global FE level.

5 Defining a different Material behavior

Defining the material behavior is similar to Abaqus. A material routine has to be written using the UMAT interface as declared in the Abaqus Manual [4]. The UMAT has to be named precisely "UMAT.f". The routine has to be written for 3D, plane strain or axisymmetric deformation states. The plane stress case is handled internally. The file UMAT.f can also contain various `SUBROUTINES` or `FUNTIONS` but no `MODULES`. Preferred is FORTRAN 90 free form code. When the UMAT is written in FORTRAN 77 fixed form style compiler directives have to be included before and after the subroutine as shown below:

```
!DEC$ NOFREEFORM

SUBROUTINE UMAT(...)

...user coding to define UMAT...

...all variables are defined just as in the Abaqus/Standard UMAT interface and have
the same type and dimension...

END SUBROUTINE UMAT

!DEC$ FREEFORM
```

The number of state variables required by the UMAT has to be specified in the Routine GetNSTATV. Name this routine precisely "GetNSTATV.f".

```
FUNCTION GetNSTATV(NTENS,NDI,PROPS)

INTEGER(KIND=AbqIK)::  GetNSTATV
INTEGER(KIND=AbqIK), INTENT(IN)::  NTENS,NDI
REAL(KIND=AbqRK),INTENT(IN)::  PROPS(:)

...user coding to define GetNSTATV. If NTENS==3 (plane stress) add +1 to that value
since the plane stress algorithm needs one internal state variable too...

END FUNCTION
```

Write a routine to check if the material parameters are reasonable and name it precisely "CheckMaterialParameters.f". Use the STDB_ABQERR function to stop the simulation and report errors to the inputfile [4]. If you do not want to check the parameters just return using the FORTRAN statement `RETURN`.

```
SUBROUTINE CheckMaterialParameters(PROPS)

REAL(KIND=AbqRK),INTENT(IN)::  PROPS(:)
...
!  Check the properties

if ( ...  )  then
CALL STDB_ABQERR(-3, "CheckMaterialParameters reports:  ...  ", 0, 0.0, " ")
else if ( ...  )  then
...
end if

END SUBROUTINE CheckMaterialParameters
```

The last file which must be present is called "UMATUtilityModules.f". It can include Modules, which can be accessed in the UMAT routine through the `USE` statement. All 4 files (and possibly more, which are accessed in the 4 files with `INCLUDE` statements) have to be put into a directory with a reasonable name inside of the `materialroutines` folder. When the program is compiled as described in section 2 specify `MATERIAL=` with the name of the just created directory.

6 Version history

date	description
2020-11-26	return correct PNEWDT; call UMAT in UXMAT FORTRAN 90 style; deliver identity matrix to UMAT as deformation gradient in small deformation simulations as done in Abaqus; deliver the correct GP COORDS to UMAT; return the symmetric Part of AMATRX; in UXMATMISES a algorithm for plane stress is added, which can be used for arbitrary UMATs
2020-12-10	Newton Algorithm in UMATMises.f corrected; handover of STATEVs from UMAT to UHARD corrected; geometrical stiffness in finite deformation UelCXU.f95 added
2021-03-10	major revision of the geometrical stiffness matrix
2021-19-07	material routines are now to be put into folder 'materialroutines' and compiled with `make MATERIAL=...`, example materialroutines added
2021-10-11	material routine definitions can now also contain modules
2022-10-05	optional argument in UEL interface, so additional information can be exchanged with the UEL, while being still compatible with Abaqus
2022-03-11	in large deformation, last timestep is chosen as the reference configuration, bug in hyper integration removed
2022-02-12	Explicit interface in UEL
2024-04-08	UEL now 100% Abaqus standard conform, GET_n_STATEV_elem and ExtendedUELModule removed

References

[1] N. Lange, G. Hütter, B. Kiefer: An efficient monolithic solution scheme for FE^2 problems, arxiv.org/2101.01802.

[2] T.J.R. Hughes, J. Winget : Finite Rotation Effects in Numerical Integration of Rate Constitutive Equations Arising in Large Deformation Analysis, International Journal for Numerical Methods in Engineering, vol. 15, pp. 1862–1867, 1980.

[3] EA de Souza Neto, D Perić, DRJ Own: Computational methods for plasticity, theory and applications, John Wiley Sons Ltd, 2008.

[4] ABAQUS/Standard User's Manual, Version 6.9, Michael Smith, Dassault Systèmes Simulia Corp, 2009