Supporting Information

Visualization of User Element in ABAQUS

Niraj Kumar Jha, Leibniz University Hannover, Germany

1 Introduction

Abaqus is a well known and widely used commercial finite element software. A large class of elements and material formulations have already been included into this commercial software for finite element analysis. However for specific applications e.g. gradient enhancement / non-local continuum /multi-field models, advanced users desire to implement their own elements via user subroutine (`UEL`/`UELMAT`). The underlying issue of custom user element is that it does not support visualization.
This supporting document serves as a basis for the reader on how to visualize the user-elements. Every analysis in Abaqus/CAE is done by using an input (`.inp`) file which contains a complete description of the numerical model. The input file is a text file and composed of several option blocks that contain one or more data lines followed by a keyword. It is easy to modify input file by using a text editor (if necessary), this key idea allow us to visualize custom user elements. The workflow for creating and modifying an input file for the visualization is as follows:

1. In a preprocessing step, create the model of a physical problem using Abaqus/CAE graphical user interface (GUI). Output the input file for necessary modifications. This step is detailed in Section 2.
2. With the `COMMON BLOCK` statement and state variable arrays `SVARS` user subroutine `UELMAT`/`UMAT` file is modified. This step is detailed in Section 3.
3. Once the finite element simulations with user routines are completed. Displacements, stresses, or other solution dependent variables are visualized in Abaqus/Viewer.

2 Preprocessing stage

At this stage one must define the computational model and create an Abaqus input file. The geometric modeling is done graphically using Abaqus/CAE or another preprocessing softwares like ANSA, HyperMesh, Gmsh etc.

2.1 Creating a model in Abaqus/CAE

This section provides the basic steps to create a three dimensional model. To illustrate each of the steps, we will first create a rubber block model which is under tensile loading (see Fig. 2.1). Abaqus/CAE is divided into several modules, starting from defining the geometry, material properties, and generating a mesh. For this tutorial, we will perform the following tasks:

- **Part** - Sketch a two-dimensional profile and create a part representing the block.
- **Property** - Define the material properties and other section definitions.
- **Assembly** - Assemble the model and choose instance type (Dependent or Independent). Create all necessary node and element sets.
- **Step** - Configure the analysis procedure. For this tutorial the Static analysis consist of two steps:

1. An initial step, in which we apply a boundary condition that constraints one end (left side) of the block.
2. A general, Static analysis step, in which we apply a displacement on the right side of the block.

- **Mesh** - With the mesh module one can generate the finite element mesh. Abaqus/CAE is used to create the mesh, the element shape, and element types. There are number of in-built meshing techniques but the default meshing technique assigned to the model is indicated by the color of the model.

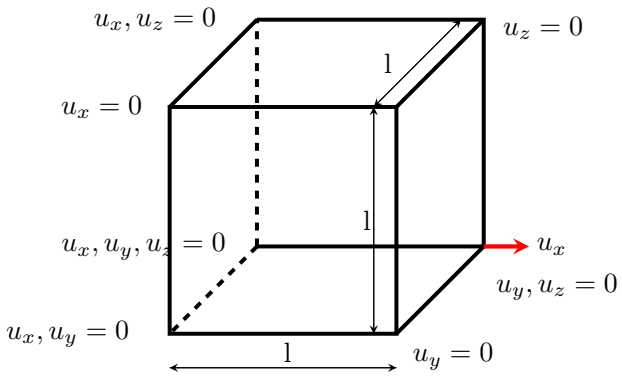

Figure 2.1: Uniaxial Cube Model

- **Job** - Once analysis is configured then Job module is used to create a job. Also an input file can be generated by clicking "Job" → "Write Input".

2.2 Format of the input file

In this section, we describe the format of the Abaqus input file. The block illustrated in Fig. 2.1 is discretized into 64 solid elements. The data structure of the computational model is explained below.

```
*Heading
 Static analysis of rubber block
** Job name : Job-1 Model name: Model-1
*Part, name=block
*Node, Nset=All
 1,0,0,0
 2,0.25,0,0
 ..................
 ..................
 124,0.75,1,1
 125,1,1,1
*Nset,Nset=All,generate
 1, 125, 1
*Nset,Nset=Left
 1,6,11,16,21,26,...,116,121
*Nset,Nset=Right
 5,10,15,20,25,30,..,120,125
*Nset,Nset=Bottom
 1,2,3,4,5,26,27,..., 104,105
```

Heading

Node & Node sets

```
*Element,type=C3D8,Elset=All
 1, 1,2,7,6,26,27,32,31
 2, 2,3,8,7,27,28,33,32
 3, 3,4,9,8,28,29,34,33
 4, 4,5,10,9,29,30,35,34
 .................
 .................
 63, 93,94,99,98,118,119,124,123
 64, 94,95,100,99,119,120,125,124
*Elset, Elset=All, generate
 1, 64, 1
*Solid section, Elset=All, material=Rubber
*End Part
**********************************************************
*Material, name=Rubber
*Hyperelastic, Neo Hooke
 10, 0.1
**********************************************************
*Step, Name= Load-unload, nlgeom=yes, inc=1000
*Static
 0.001,0.05,,0.001
*Boundary
 Left, 1
 Bottom, 2
 All, 3
*Boundary,type=Displacement
 Right,1,1,1.0
 **
 **
*Output, Field
*Node Output, Nset=All
 u
*Element Output, Elset=All
 s,le
 **
*End Step
```

Element & Element sets

Material card

Steps definitions

As we notice that, the input file consists of all the information about nodes and their coordinates, elements and their connectivity, node sets, element sets, boundary conditions, output variables etc. The input file consists of three different types of lines. Keyword line, data line and comment line. Data line is often followed by keyword line and the comment line is the optional one. The keyword line starts with a "*", the comment line with "**", and the data line has no prior symbol. These are illustrated below.

```
*Node                                   Keyword Line
 1, 1.0,1.5,0.5                         Data Line
 **This is the defnition of a node      Comment Line
```

The basic structure of input file consists of six parts, Heading, Node, Element, Node and Element set, Material and Step definition. These are discussed in detail below.

2.2.1 Heading

This part of the input file consists of the name of the analysis which we are going to perform

```
*Heading
 Static analysis of rubber block
```

The text underneath this command describes the model that will appear in the output (`.odb`) database.

2.2.2 Node definition

This part of the input file consists of the node numbers and the associated coordinates. This part basically defines the nodes.

```
*Nodes
 1,0,0,0
 2,0.25,0,0
 .
 .
 124,0.75,1,1
 125,1,1,1
```

As illustrated above, the keyword `*Node` is followed by the data line `1,0,0,0`. It is mandatory that all the entries in the data line are separated by commas. The first entry signifies the node number, the second, third and fourth entries signifies `X`, `Y` and `Z` co-ordinates respectively.

2.2.3 Element definition

This part of the input file consists of the element number, type and the nodes associated to it. This part basically defines the element which is very similar to the node definition.

```
*Element, type = C3D8
 1, 1,2,7,6,26,27,32,31
 2, 2,3,8,7,27,28,33,32
 .
 .
 63, 93,94,99,98,118,119,124,123
 64, 94,95,100,99,119,120,125,124
```

As illustrated above, the keyword `*Element` is followed by `type` and then a data line `1, 1, 2, 7, 6, 26, 27, 32, 31`. Here `type` is a mandatory parameter which describes the element type. In the above example, `C3D8` stands for three dimensional continuum (fully integrated) element with eight nodes. The first entry signifies the element number and is followed by nodal connectivity.

2.2.4 Node and Element set definition

This part of the input file consists of groups of nodes and elements with a common name. Basically this part groups the nodes and elements into sets known as node sets and element sets respectively.

```
*Nset,Nset=All
 1, 125, 1
*Elset, Elset=All
 1, 64, 1
```

As illustrated above, the keywords `*Nset` and `*Elset` are followed by the name of sets and corresponding data lines.

2.2.5 Material definition

This part of the input file consists of the material definition of element.

```
*Material, name=Rubber
*Hyperelastic, Neo Hooke
 10, 0.1
```

The keyword `*Material` is followed by the material class keyword, corresponding parameters and data lines. For example `*Hyperelastic` and `*Elastic` are two different material class keywords. As illustrated above, the parameter `Neo Hooke` of `*Hyperelastic` specifies the material model which requires two material parameters to be specified in the subsequent data line. The order of constant is important and one can refer Abaqus documentation for further information.

2.2.6 Step definition

This part of the input file consists of details about the analysis such as boundary conditions, loads, output parameter requests etc.

```
*Step, Name= Load-unload, nlgeom=yes, inc=1000
*Static
 0.001,0.05,,0.01
*Boundary
 Left, 1
 Bottom, 2
 All, 3
*Boundary,type=Displacement
 Right,1,1,1.0
 **
 **
*Output, Field
*Node Output, Nset=All
 u
*Element Output, Elset=All
 s,le
 **
*End Step
```

As illustrated above, the keyword `*Step` is used to start a step. There are many optional parameters to this keyword and only three are used in this example. The keyword `Name` is used for assigning the name of step. `nlgeom` is used specify the analysis type whether linear or non-linear. The keyword `inc` is used to specify the maximum number of time increments allowed per step. This parameter is used to limit the incrementations to a certain value beyond which the execution terminates. The keyword `*Static` specifies the type of analysis. The data line after this passes the information about the analysis. The first entry signifies the initial time increment which is later modified in case of automatic increment. The second entry signifies the time per step. The third entry in this example is left empty signifies the minimum time increment allowed. The fourth entry signifies the maximum time increment allowed.
The keyword `*Boundary` and the `*Boundary, type = Displacement` are used to specify fixed and displacement boundary conditions respectively. The first entry of this data line specifies the node number or the node set name, the second specifies the starting degree of freedom that has the displacement boundary condition, the third signifies the ending degree of freedom that has the displacement boundary condition. There are other variants of this displacement boundary condition, for further information one can refer to Abaqus documentation. Apart from displacement boundary or `Dirichlet boundary conditions`, forces or `Neumann boundary conditions` can also be specified with the following keyword `*Dload` and `*Cload`.
Now once boundary conditions are specified, the output requests should be passed to the Abaqus solver. `*Output` is the keyword which are of two types field and history. Field output gives the spatial distribution of variable at certain point of time whereas the history output gives the variation of variable over time at a certain point in space. Both are activated using the parameters `Field` and `History` respectively along with `*Output`. `*Output` is followed either by `*Element Output` or `*Node Output` as illustrated in the example. `*End Step` ends the step definition.

2.3 Modification in an input file

In this section, we discuss the changes that should be made in the input file for use with user-elements. The main body of the input file remains unchanged except element / element sets and material card.

```
*User Element, Type=U1, Nodes=8, Coordinates=3, Var=96, Integration=8,Tensor=Threed
 1,2,3
```

```
*Element,Type=U1,Elset=All
 1, 1,2,7,6,26,27,32,31
 2, 2,3,8,7,27,28,33,32
 .
 .
```

As illustrated above, `*User Element` consists of many parameters and a data line followed by element definitions. The parameter `Type` specifies the type of user element and acts as an identifier, `U1` for this case. `Nodes` specify the number of nodes of user element and `Coordinates` specify the number of coordinates. `Var` specifies the total number of solution dependent state variables per element. The total number of solution dependent state variables is equal to the number of state variables per integration point multiplied by the number of integration points. The keyword `Integration` specifies the number of integration points per element. `Tensor` is the actual parameter which describes the type e.g. `Threed`. The data line underneath the keyword `*User Element` specifies the number of degrees of freedom per node.
Along with the definition of user element, an equivalent (`Dummy`) standard element with same nodes but different numbering system with some (`offset = 100000`) value is adopted. By this the nodes of user element and the (`Dummy`) standard elements are overlayed to each other.

```
*Element, Type=C3D8,Elset=Dummy
 100001, 1,2,7,6,26,27,32,31
 100002, 2,3,8,7,27,28,33,32
 .
 .
*Solid Section, Elset=Dummy, Material=Dummymat
*Material, Name=Dummymat
*User Material, constants=2
 1.0e-11, 0.3
*Depvar
 12
```

It can also be observed that `Dummy` element set has user defined material class `User Material` which is followed by `*Depvar` option. This option is used to store the solution dependent state variables e.g. stresses and strains at Gauss points. With this keyword one can visualize SDV's on `Dummy` elements. In order to visualize the SDV's the keyword `*Element Output` is used. This is illustrated below.

```
*Element Output, Elset=Dummy
 SDV
```

With the slight modifications in `UELMAT`, which is further discussed in the subsequent section, one can map the solution dependent variables of user elements on `Dummy` element.

3 Modification in Fortran file

The user subroutine `UEL/UELMAT` consists of many other subroutines which are necessary to define shape functions, setup the stiffness matrix (`AMATRX`), residual vector (`RHS`), state variables (`SVARS`) etc. There is a main subroutine `UELMAT` in which other subroutines e.g. shape functions, stiffness matrix, jacobian etc. are defined. The basic structure of `UELMAT` subroutine is summarized below.

3.1 User Subroutine - UEL/UELMAT

```
SUBROUTINE UELMAT(RHS, AMATRX, SVARS, ENERGY, NDOFEL, NRHS,
1 NSVARS, PROPS, NPROPS, COORDS, MCRD, NNODE, U, DU, V, A,
2 JTYPE, TIME, DTIME, KSTEP, KINC, JELEM, PARAMS, NDLOAD,
3 JDLTYP, ADLMAG, PREDEF, NPREDF, LFLAGS, MLVARX, DDLMAG,
4 MDLOAD, PNEWDT, JPROPS, NJPROP, PERIOD, MATERIALLIB)
```

```
C
  INCLUDE 'ABA_PARAM.INC'
C
  DIMENSION RHS(MLVARX,*), AMATRX(NDOFEL, NDOFEL), PROPS(*),
  1  SVARS(*),ENERGY(*), COORDS(MCRD, NNODE), U(NDOFEL),
  2  DU(MLVARX,*), V(NDOFEL), A(NDOFEL), TIME(2), PARAMS(*),
  3  JDLTYP(MDLOAD,*), ADLMAG(MDLOAD,*), DDLMAG(MDLOAD,*),
  4  PREDEF(2, NPREDF, NNODE), LFLAGS(*), JPROPS(*)
C
   PARAMETER (NDIM=3, NDOF=3, NDI=3, NSHR=3, NNODEMAX=8,
   1     NTENS=6, NINPT=8, NSVINT=12, NELEMENT=64)
C
   INTEGER JELEM
   DATA WGHT /ONE, ONE, ONE, ONE, ONE, ONE, ONE, ONE/
   .
   .
   .
   .     (INCLUDE USER CODE HERE)
   .
   .
   .
   RETURN
   END
```

As illustrated above `SUBROUTINE UELMAT` is the header file that is followed by variable declarations. In the header file all the standard variables are declared which communicates with Abaqus solver. There are many variable types in fortran such as integers, real floating point numbers, characters, strings, arrays etc. This illustrated with simple examples below.

```
DIMENSION RHS(2,1)
PARAMETER (NDIM = 3)
INTEGER K
```

The keyword `DIMENSION` is used to declare the dimension of a variable. In the above example `DIMENSION RHS(2,1)` declares the variable `RHS` as an array of dimension (2x1). `PARAMETER` is used to assign a value to a variable. In the above example, the variable `NDIM` set to three (3). `INTEGER` is used to declare the variable `K` as integer.

3.2 Changes in Fortran file

The underlying idea of visualization is to transfer the variables via `SVARS` from `UELMAT` to the `UMAT` subroutine. This is achieved by adding the following lines to `UELMAT`

```
REAL*8 UVAR
INTEGER JELEM
COMMON/KUSER/UVAR(NELEMENT,NSVINT,NINPT)
```

In `UELMAT`, `SVARS` is loaded into `UVAR` and `COMMON` block make the variables `UVAR` available to both `UELMAT` and `UMAT`.

```
DO K1=1,NSVINT
 UVAR(JELEM,K1,KINTK) = SVARS(NSVINT*(KINTK-1)+K1)
END DO
```

With `UMAT` interface it is possible to define any constitutive model of arbitrary complexity. To code user material, following inputs are required: definition of stress, stress rate only (in co-rotational framework), dependence on time, temperature, or field variable, internal state variables, either explicitly or rate form. In this report, we only discuss the changes to be made for visualization.

```
SUBROUTINE UMAT(STRESS, STATEV, DDSDDE, SSE, SPD, SCD, RPL,
1 DDSDDT, DRPLDE, DRPLDT, STRAN, DSTRAN, TIME, DTIME, TEMP,
2 DTEMP, PREDEF, DPRED, CMNAME, NDI, NSHR, NTENS, NSTATV,
```

```
3 PROPS, NPROPS, COORDS, DROT, PNEWDT, CELENT, DFGRD0,
4 DFGRD1, NOEL, NPT, LAYER, KSPT, KSTEP, KINC)

INCLUDE 'ABA_PARAM.INC'

DIMENSION STATEV(NSTATV)
PARAMETER(NINT=8, NSTV=12, NELEMENT=64, ELEMOFFSET=100000)
INTEGER NELEMAN, K1

COMMON/KUSER/UVAR(NELEMENT,NSTV,NINT)
NELEMAN=NOEL-ELEMOFFSET

DO K1=1,NSTV
 STATEV(K1)=UVAR(NELEMAN,K1,NPT)
END DO

RETURN
END
```

UELMAT to UMAT

Thus, state variables from `UELMAT` is transferred to `UMAT` and stored in `STATEV` which can be easily visualized in `Dummy` element.

Acknowledgement : The author would like to thank Masters student Hema Rajesh of IIT Bombay for the kind assistance.

References

ABAQUS/Standard Analysis User's Manual (2014). SIMULIA

E-mail address: niraj.kumar.jha@ibnm.uni-hannover.de