**Mini Documentation – Abaqus/Standard & Abaqus/Explicit**

IP: Integration Point

@ time : at the start of the increment

@ time : at the end of the increment

**A) Main Subroutines**

Remember: 

**Subroutine Calls**

**IMPLICIT**: 1) UFIELD,  2) UMAT,  3) UMATHT

**EXPLICIT**: 1) VUMAT,  2) VUMATHT,  3) VHETVAL, 4) VUFIELD

**1) UMAT (VUMAT):** The algorithm is as follows:

**Given:** ,  ( in VUMAT) , , , , and 

**Find:** , ,  (to be used in UMATHT (VUMATHT) in the  formulation only),

 (“RPL” - in UMAT / “flux(k)” in VHETVAL)

and , , ,  (in UMAT only).

**2) UMATHT (VUMATHT):** The algorithm is as follows:

**Given:** , and , , ,

and  (in the  formulation only)

**Find:** , 

and , , , (the last 3 in UMATHT only).

**B) Auxiliary Subroutines**

**ABAQUS/STANDARD**

Globally Available Arrays (using module ktransfer):

A1) Parameters for Linear 4-Node (2D) and 8-Node (3D) elements

* kxdim = Problem's dimension (2 in 2D, 3 in 3D problems)
* kxmtrx = Auxialiary integer to define dimensions of DELTA etc.

(4 in 2D, 6 in 3D problems) (In Quad/Hex Codes Only)

* kxnodel = Nodes per element (4 in 4-node, 8 in 8-node elements)
* kxntens = Total components of tensors (4 in 2D, 6 in 3D problems)
* kxelpernode = Max number of elements belonging to a node (allocate a large

enough number)

* kxelem (= kmaxel) = Largest element label in the model (may skip between numbers)
* kxnode (= kmaxnodes) = Largest node label in the model (may skip between numbers)
* ktotalnodes = Total number of nodes in the model
* ktotalel = Total number of elements in the model
* ngaus = Number of Gauss integration points (4 in 4-node, 8 in 8-node

elements)

i.e. parameter( kxdim = 2,

+ kxnodel = 4,

+ kxmtrx = 4,

+ kxelpernode = 10,

+ kxelem = 4,

+ kxnode = 9,

+ ktotalnodes = 9,

+ ktotalel = 4,

+ ngaus = 4 )

A2) Parameters for Quadratic 8-Node (2D) elements

* coorT(50 000,ngaus,2): Coordinates of IPs at time 
* ShT(50 000,ngaus): Pressure at IPs at time 
* gradT(50 000,ngaus,2): Pressure Gradient at IPs at time 
* ngaus: Number of Gauss Points (4 for Reduced & 4 for Full Integration)

Note: The number 50 000 is used to allocate a large space in case more elements are used in the mesh.

B) Arrays (for Linear 4-Node (2D) and 8-Node (3D) elements)

* PELEM(kxelem,ngaus): Pressure at the IP of each element at time .
* PNODAL(kxnode): Average Pressure at each node at time . This array is used at the start of the next increment to calculate the pressure gradient at IPs.
* GRADP(kxelem,ngaus,kxdim): Pressure Gradient at IPs at time .
* ADETJEL(kxelem,kxnodel): Array containing the determinant of the Jacobian matrix (). Every row (the row number implies the element label) contains the values of the determinant for each node of the corresponding element.
* IELCONN(kxelem,kxnodel): Array containing element connectivities. (calculated once at the beginning of the step). Every row (the row number implies the element label) contains node labels--- see below for an example ---.
* INODETOEL(kxnode,kxelpernode,2): Array containing at every row:

a) The number of elements the node (implied by row number) belongs to (1st column),

b) The element labels this node belongs to (rest of the columns) (calculated once at the beginning of the step),

c) The local node number (for 4-Node & for 8-Node elements) this node holds in the element stored in the corresponding column --- see below for an example---.

1) UEXTERNALDB: Subroutine used (see also Abaqus Documentation for details):

* To initialize the arrays *coorT, gradT, ShT* (for quadratic 8-Node (2D) elements) or the arrays *PELEM, PNODAL, GRADP* (for linear 4-Node (2D) or 8-Node (3D) elements) which are globally available and used for the calculation of pressure gradient at the IPs of every element.
* To define only once the unit matrices *DELTA, AIMX, AKMX* which are globally available.
* To calculate element and node connectivities at the start of the step (subroutine KNODETOELCON).
* To calculate the determinant of the Jacobian (array *ADETJEL*) for the nodes of all elements at the end of each increment (subroutine KDETJEL).
* To calculate nodal pressures at the end of each increment (subroutine KPNODAL).

2) KGRADP: (see also “Pressure\_Gradient.docx” for more details on specific equations):

* Is called in each element at each IP and calculates the corresponding pressure gradient (*gradT*) using an interpolation from the pressures at the IPs. This is used for **8-node, quadratic plane elements** with either full (3x3x3 IPs) or reduced (2x2 IPs) integration.
* The pressure gradient is calculated using a) The coordinates of the IPs, b) The values of the pressures at the IPs **at the end of the previous increment.**

3a) KGRADP2D: (see also “Pressure\_Gradient.docx” for more details on specific equations):

* Is called in each element at each IP and calculates the corresponding pressure gradient (*gradT*) using an interpolation from element averaged pressures at the nodes. This is used in **4-node, linear plane elements** with either full (2x2 IPs) or reduced (1 IPs) integration.
* The pressure gradient is calculated using a) The coordinates of the element’s nodes, b) The values of the pressures at the Nodes **at the end of the previous increment.**

3b) KGRADP3D: (see also “Pressure\_Gradient.docx” for more details on specific equations):

* Is called in each element at each IP and calculates the corresponding pressure gradient (*gradT*) using an interpolation from element averaged pressures at the nodes. This is used in **8-node, linear *3D* elements** with either full (2x2x2 IPs) or reduced (1 IPs) integration.
* The pressure gradient is calculated using a) The coordinates of the element’s nodes, b) The values of the pressures at the Nodes **at the end of the previous increment.**

4) UFIELD: (see also Abaqus Documentation for details):

This subroutine is used to store the current coordinates of the nodes for each element in the globally available array coorT(kxelem, kxnodel, kxdim) so they are available in UMAT for the pressure gradient calculation in a **4-node element with either reduced (1 GP) or full integration (2x2 GP) [2D]** or an **8-node element with either reduced (1 GP) or full integration (2x2x2 GP) [3D]**.

5) DISP: (see also Abaqus Documentation for details):

* The subroutine is used to impose the asymptotic elastic displacement field of Mode I Fracture:

i.e., first term: , 

* We also need to define the following variables (Same values from step def. of input file)
  + *SIFMAX:* This is the final normalized SIF that we want to apply (i.e., ).
  + *TMAX:* This the total time for the step.

**ABAQUS/EXPLICIT**

Globally Available Parameters & Arrays (using module ktransfer):

A) Parameters

* kxdim = Problem's dimension (2 in 2D, 3 in 3D problems)
* kxnodel = Nodes per element (4 in 4-node, 8 in 8-node elements)
* kxntens = Total components of tensors (4 in 2D, 6 in 3D problems)
* kxelpernode = Max number of elements belonging to a node (allocate a large

enough number)

* kxelem (= kmaxel) = Largest element label in the model (may skip numbers)
* kxnode (= kmaxnodes) = Largest node label in the model (may skip numbers)

i.e. parameter( kxdim = 2,

+ kxnodel = 4,

+ kxntens = 4,

+ kxelpernode = 10,

+ kxelem = 5068,

+ kxnode = 3426 )

B) Arrays

* PELEM(kxelem): Pressure at the reduced IP of each element at time .
* PNODAL(kxnode): Average Pressure at each node at time . This array is used at the start of the next increment to calculate the pressure gradient at the reduced IPs.
* GRADP(kxelem,kxdim): Pressure Gradient at IPs @ time .
* IELCONN(kxelem,kxnodel): Array containing element connectivities. (calculated once a the beginning of the step). Every row (the row number implies the element label) contains node labels--- see below for an example ---.
* INODETOEL(kxnode,kxelpernode): Array containing at every row: a) The number of elements the node (implied by row number) belongs to (1st column), b) the element labels this node belongs to (rest of the columns) (calculated once at the beginning of the step) --- see below for an example.

1) VEXTERNALDB: Subroutine used (see also Abaqus Documentation for details):

* To initialize the arrays *PELEM, PNODAL, GRADP* which are globally available and used for the calculation of pressure gradient at the IP of every element.
* To calculate element and node connectivities at the start of the step (subroutine KNODETOELCON).
* To calculate nodal pressures at the end of each increment (subroutine KPNODAL).

2) KGRADP2D (KGRADP3D): (see also “Pressure\_Gradient.docx” for more details on specific equations):

* Is called in each IP (*nblock*) and calculates the pressure gradient (*GRADP)* at the reduced IP of the element (located at (0,0) for 2D and (0,0,0) for 3D **w.r.t. the natural coordinate system** ).
* The pressure gradient is calculated using a) The coordinates of the nodes, b) The values of the averaged pressures at the nodes **at the end of the previous increment.**

3) VUFIELD: (see also Abaqus Documentation for details):

This subroutine is used to store the current coordinates of the nodes for each element in the globally available array coorT(kxelem, kxnodel, kxdim) so they are available in UMAT for the pressure gradient calculation of a **4-node element [2D]** or an **8-node element with reduced integration [3D]**.

4) VDISP: (see also Abaqus Documentation for details):

* The subroutine is used to impose the asymptotic elastic displacement field of the Mode I Fracture:

i.e., first term: , 

* We also need to define the following variables (Same values from step def. of input file)
  + *SIFMAX:* This is the final normalized SIF that we want to apply (i.e., ).
  + *TMAX:* This the total time for the step.

5) VHETVAL: (see also Abaqus Documentation for details):

This subroutine is used to calculate the “heat sink” term  in Explicit codes since this cannot be defined in VUMAT or VUMATHT.

**OTHERS (ABAQUS/STANDARD & ABAQUS/EXPLICIT)**

1) module KTRANSFER: We use this module instead of common blocks (It has similar functionality).

* **CAUTION!!** When you use the code with modules make sure that all actions performed on globally available arrays and/or variables (i.e., read, write etc.) ***are put in between “mutexes”*** (see also Abaqus Documentation for details on the subject). These should be initiated at the beginning of the analysis in the (V)UEXTERNALDB subroutine before they can be used. Also, when running an analysis through the command line make sure to use “thread parallelization” option instead of the (default) “mpi parallelization”. This is done by adding the option “mp\_mode=threads” after the “cpus=” option, i.e.,
  + **Abaqus Implicit:** abaqus job=… user=… cpus=… mp\_mode=threads
  + **Abaqus Explicit:**  abaqus job=… user=… double=both cpus=… mp\_mode=threads

2) KNODETOELCON:

* In this subroutine 2 arrays are calculated and made globally available:
  + IELCONN: Array containing element connectivities (for non-existing elements, all entries in the corresponding row are 0):

element number same as row number

Node 1 Node 2 Node 3 Node 4

e.g., (line 441, elem. 441) 4 17 28 29

(line 442, elem. 442) 0 0 0 0 (non-exist. elem.)

(line 443, elem. 443) 0 0 0 0 (non-exist. elem.)

(line 444, elem. 444) 127 135 1441 1442

* + INODETOEL: Array containing the number of elements, the element labels the corresponding node belongs to and the local node number (1,…,4 for 4-Node and 1,…,8 for 8-Node elements) the node has in each of these elements (for non-existing nodes, all entries in the corresponding row are 0):

node number same as row number

INODETOEL(…,…,1) number of elements

this node belongs to Elem. 1 Elem. 2 Elem. 3 Elem. 4 …

e.g., (line 5, node 5) 4 101 107 322 451

(line 6, node 6) 2 1101 1102 0 0 (non-exist. node)

(line 7, node 7) 0 0 0 0 0 (non-exist. node)

(line 8, node 8) 4 231 135 1641 1742

INODETOEL(…,…,2) Local Node Local Node Local Node Local Node

- Num. in El. 1 Num. in El. 2 Num in El. 3 Num. in El. 4 …

e.g., (line 5, node 5) 0 2 4 1 2

(line 6, node 6) 0 3 1 0 0

(line 7, node 7) 0 0 0 0 0

(line 8, node 8) 0 3 4 1 1

* The subroutine is called **at the start of the step**.

3) KPNODAL: This subroutine:

* Is used to calculate the averaged nodal pressures for all nodes in the mesh using the *PELEM* and *INODETOEL* arrays and stores them in the *PNODAL* array.
* Is called **at the end of every increment.** Then, the information **is used in the beginning of the next increment** in order to calculate the pressure gradient.
* Quick interpretation of the main loop (over **all node labels**) in the subroutine:
  + *N*: Number of elements that this node belongs to (i.e., it belongs to 4 elements).
  + *INODETOEL(INODE,I)[I=2,1+N]:* For each value of I, this variable returns an element label which node *INODE* belongs to.
  + *PELEM(INODETOEL(INODE,I))[I=2,1+N]* For each value of I, this variable returns the pressure at the IP of the *INODETOEL(INODE,I)* element.

4) KDETJEL: This subroutine:

* Is used to calculate the determinant of the Jacobian matrix () for all nodes and all elements in the mesh using the *IELCONN* and *INODETOEL* arrays and stores it in the *ADETJEL* array.
* Is called **at the end of every increment** before calling KPNODAL subroutine**.** Then, the information **is used in KPNODAL** in order to calculate nodal pressures using weighted element pressures based on the “element area” converging to a specific node.

5) KCT: This subroutine is used to calculate ,  and .

6) KNT: This subroutine is used to calculate  and . The following expressions are implemented:

 or ,



8) KDAMAGE: This subroutine is used to calculate , ,  and 

**C) Miscellaneous**

**Analyses Related**

* ***(CAUTION!!)***: Before running an analysis of the **SSY problem** (both in Abaqus/Standard & Explicit) **always remember to check DISP subroutine (VDISP in Explicit)** because we also define there the following:
  + Young’s modulus (*E*)
  + Poisson’s ratio (*ANU*)
  + Final normalized SIF  (see below) (*SIFMAX*)
  + Total Time of the step (*TMAX*)
* In Abaqus/Explicit the option ENHANCED for hourglass control in the **SSY problem** gives better results compared to the STIFFNESS option. => **We use HOURGLASS=ENHANCED in this case.**
* In Abaqus/Explicit in order to have a quasi-static analysis (so as to have equivalence with implicit)

we keep in mind:

* + To define a **SMALL DESNITY** .
  + The TOTAL TIME of the analysis **also affects the kinetic energy**.
  + ALWAYS MAKE SURE THAT: **Kinetic Energy << Strain Energy**
* In Abaqus/Explicit a first estimate for the stable time increment can be given as (see also Abaqus Documentation for details --- Fully coupled thermal-stress analysis ---)

Mechanical Problem:  , 

where 

, 

Thermal Problem: , ,

where , , 

* With the option \*DYNAMIC TEMPERATURE DISPLACEMENT when we use in \*BOUNDARY the

option TYPE=DISPLACEMENT (even with VDISP) we need to use Amplitude curves.

* ***(CAUTION!!)***: When ***Abaqus/Explicit*** is used along with the element deletion technique to simulate crack propagation, Abaqus sets all properties passed into VUMATHT to 0 (e.g., density etc.). In order to avoid erroneous results (i.e., divisions with 0), a provision is added in this case to manually set user defined SDVs or arguments to be updated (such as internal energy U, derivatives etc.) to 0.

**General things to keep in mind**

* In all codes we make sure that the bottom **right corner node** **is added to the outer boundary layer**

where the asymptotic displacement field is imposed. NOT in XAXIS.

* The dimensions of both meshes (Aravas & Paneda for the problem of SSY) now use the same scale (we have divided all X & Y coordinates in Paneda's mesh by 5x10^-4) so **the same normalizedcan be used now in both cases and the analyses will be equivalent.**

**D) Small Scale Yielding Problem Details (Crack)**

**Boundary Condition Explanation (**VUMAT is the same in both cases**)**

* “Constant ” BC: Impose on free surfaces (OUTERN & CRACK\_SURFACE) a value of 
* “Zero Flux” BC: Don't impose any BC for  on free surfaces (OUTERN & CRACK\_SURFACE)

**Properties and Normalization (Example Properties from Sofronis’ papers)**

The mechanical properties of the material and the properties related to hydrogen diffusion are summarized in Tables 1 and 2 (Taha & Sofronis, 2001):

*Table 1*: Mechanical properties

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | Hardening Exponent |
| 207 | 0.3 | 250 | 5 |

, 

*Table 2*: Diffusion Properties

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Dimensions & material properties used in the input file are normalized as follows:

Dimensions & Time: , , ,

Mechanical Properties: , ,

Diffusion Properties: , , , ,

Loads: .

*Table 3*: Normalizing Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  | 8.314 | 300 |

**Loads & BC in each Solver**

*Table 4*: Loads and BC in Abaqus/Standard (see also Taha & Sofronis, 2001).

|  |  |
| --- | --- |
| Load Level |  |
| Total Time |  |
| Constant Concentration B.C. |  |
| Zero Flux B.C. |  |

*Table 5*: Loads and BC in Abaqus/Explicit.

|  |  |
| --- | --- |
| Load Level |  |
| Total Time |  |
| Zero Flux B.C. |  |