Abaqus Documentation

For running an Abaqus simulation the first step is creating the input files.

1. Start Abaqus

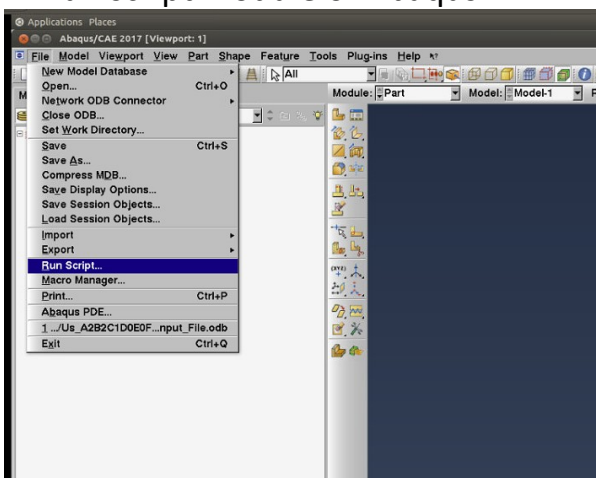
Abaqus can be accessed with the following command:
Module load abaqus/6.14
After loading Abaqus the cae can be loaded with:
Abaqus cae mesa
Please note that in the following workflow only Abaqus version 6.14 can be used. Using Abaqus 2017 will cause problems.

2. Creating input Files

There are 3 main input files which need to be provided.

2.1 Geometry input file

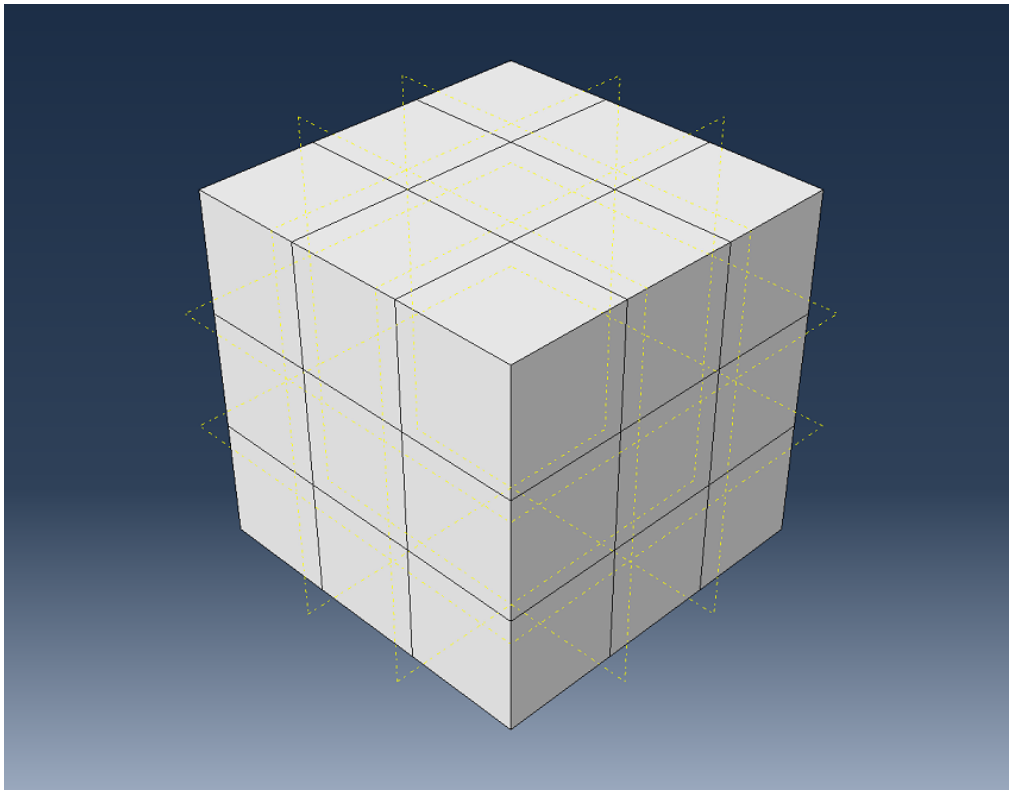
At the first step the geometry description of the model must be created. This can be done using a python script: ***abaqus_macro_more_elements.py.*** Number of elements and RVE size can be selected in this script. This script can must be loaded in Run script module of Abaqus.



After running the script, an input file is created which is named: ***inputfile_model.inp.*** We need to make this geometry periodic using another script which is called: ***Create_PeriodicBC_EDGE_3D.py.*** The input file created in the previous step must be at the same folder and the input file name inside the Create_PeriodicBC_EDGE_3D.py script must match this name. please

note that this script must be executed only through terminal. Running from the Abaqus cae will cause empty vertices.inp file.
The command in the terminal is:

Abaqus python Create_PeriodicBC_EDGE_3D.py



And a folder is created which is called **PeriodicData.** In this folder the main geometry input file has been created which is called: **geometry_Periodic.inp.** The content of this file creates the body of our final input file which will be used to run the jobs. This geometry file contains different parts:

- *Heading*

All the information related to model name, job name, Abaqus version can be seen in this part.

```
** Job name: inputfile_model Model name: Model-1
** Generated by: Abaqus/CAE 2017
*Preprint, echo=NO, model=NO, history=NO, contact=NO
**
```

- *PARTS*

In our simple case of RVE we have only one part. Using the include command 5 important files defining the periodic boundary condition for the RVE are added to this part. It should be noted that these files have been created after running the script ***Create_PeriodicBC_EDGE_3D.py.***

```
*Part, name=Part-1
*Include, input=PeriodicData/LeftToRight.inp
*Include, input=PeriodicData/BottomToTop.inp
*Include, input=PeriodicData/FrontToRear.inp
*Include, input=PeriodicData/Edges.inp
*Include, input=PeriodicData/Corners.inp
*End Part
```

- *Assembly*
  - *Nodes*

In this part a list with total number of nodes can be seen. Each node has its own id in x,y,z coordinates. It should be noted that as we are dealing with cubic elements each independent cube has 8 nodes, since nodes are common between neighboring cubes, in case of 27 elements total number of independent nodes would be 64.

```
*Node
      1,  0.189999998,  0.189999998,  0.189999998
```

  - *Elements*

After defining the nodes, they need to be assigned to elements. The type of element here is C3D8 (Continuum mechanics, 3D and as mentioned before having 8 nodes). Each element has 8 nodes.

```
*Element, type=C3D8
1, 5, 6, 7, 8, 1, 2, 3, 4
2, 11,  6,  5, 12,  9,  2,  1, 10
3, 15,  6, 11, 16, 13,  2,  9, 14
4, 20,  7, 21, 22, 17,  3, 18, 19
5, 24,  8,  7, 20, 23,  4,  3, 17
6, 27,  8, 24, 28, 25,  4, 23, 26
7, 30,  5,  8, 27, 29,  1,  4, 25
8, 32, 12,  5, 30, 31, 10,  1, 29
 9,  1,  2,  3,  4, 33, 34, 35, 36
10,  9,  2,  1, 10, 37, 34, 33, 38
11, 13,  2,  9, 14, 39, 34, 37, 40
12, 17,  3, 18, 19, 41, 35, 42, 43
13, 23,  4,  3, 17, 44, 36, 35, 41
14, 25,  4, 23, 26, 45, 36, 44, 46
15, 29,  1,  4, 25, 47, 33, 36, 45
16, 31, 10,  1, 29, 48, 38, 33, 47
17,  3, 18, 42, 35,  2, 13, 39, 34
18, 52, 49, 50, 51,  6, 15, 21,  7
19, 51, 50, 53, 54,  7, 21, 22, 20
20, 56, 51, 54, 55,  8,  7, 20, 24
21, 56, 55, 57, 58,  8, 24, 28, 27
22, 59, 52, 51, 56,  5,  6,  7,  8
23, 61, 60, 49, 52, 11, 16, 15,  6
24, 59, 56, 58, 62,  5,  8, 27, 30
25, 64, 59, 62, 63, 12,  5, 30, 32
26,  6, 15, 21,  7,  2, 13, 18,  3
27, 11,  6, 52, 61, 12,  5, 59, 64
```

- *Node set and Element set*

After defining nodes and assigning 8 nodes to each element, they need to be grouped as element sets which can be considered as grains. In a simple case we just need one element per grain.

```
*Nset, nset=Set-1
  7, 20, 21, 22, 50, 51, 53, 54
*Elset, elset=Set-1
 19,
```

- *Section*

Assigning different microstructural description to each grain (elementset) is done in this part. It is possible to assign different materials to each grain but in the simplest case the only difference in different materials is their orientation. Usually, we used our own material definition which will be added in the next part.

```
** Section: Section-14
*Solid Section, elset=Set-14, material=Material-14
```

2.2 Material input file

In the created **geometry_Periodic.inp** file there is a material description which we do not use, and it needs to be replaced with one's own orientation description. For creating the material input file, the first step would be to provide a text file with orientation descriptions with three Euler angles. Note that the number of these orientations should be equal to number of grains or elementsets. In this case we have 27 elements and one element per grain so 27 orientations need to be provided.

MTEX toolbox in Matlab can be used for creating a list of orientations. Using this toolbox, the output will be a list of n random orientations in the notion of Euler angles. Note that this method can only be used for random texture.

1. Start Matlab
2. Change Matlab directory to where the MTEX is installed
3. Type command startup_mtex
4. Define crystal symmetry

```
cs = crystalSymmetry('m3m')
```

5. Sample n discrete random orientation acc. to the defined symmetry

```
ori = orientation.rand(n,cs)
```

6. Export orientations as .txt file with default options (Bunge Euler angles in deg) s

```
ori.export('Orientation.txt')
```

The final file will be a text file with n orientations with name Orientation.txt

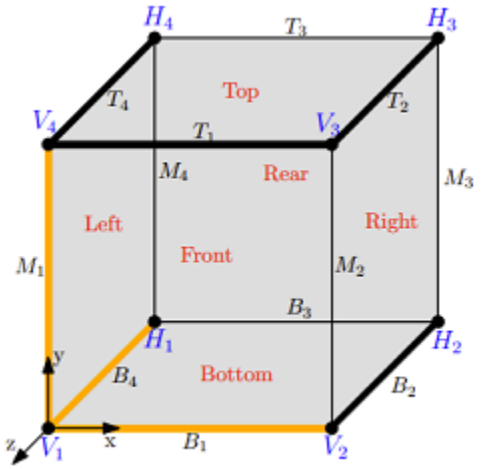
Since the orientation file is in degrees, please remember to change the orientations to radians, it is very important since it will cause conflicts.

For this purpose there is an script “Degrees_to_Radians.py” which can be used, please remember to place orientation file at the same location and also enter the file name in the script correctly.

2.3 Load input file

In this in file we need to define the boundary conditions which leads to a more realistic deformation of the RVE. The main idea is that the relative displacement between opposing nodes is the same. The nodal displacement of opposite nodes is coupled and still allowing a global strain in this direction. We can define the relative displacement between opposite nodes which should be equal to the displacement an average strain

In this case the loads are applied to representative nodes at the corners of the RVE.



As can be seen in this figure, the independent nodes are shown in orange and the dependent ones in grey. The global boundary conditions must be applied to V_1, V_2, V_4 and H_1 that are the only independent nodes within the RVE.

$$u_x^R - u_x^L = \varepsilon_{xx} \cdot l_x \qquad \rightarrow \qquad \frac{u_x^R - u_x^L}{l_x} = \varepsilon_{xx},$$

So, it is possible to have an average strain between opposite nodes. Using periodic boundary conditions are important for determination of realistic deformation of RVEs. During the implementation of the periodic boundary conditions, it is necessary to couple opposite nodes but allowing appearance of a global strain.

The boundary condition can only be applied to the four corner nodes. V_1, V_2, H_1, V_4. The surfaces are subdivided into three different sets of surface nodesets, edge nodesets and the vertices which must be treated differently.

Surface Nodes	Edge nodes	Corner Nodes
Left to Right nodeset: $\mathbf{u}_i^{\text{Right}} - \mathbf{u}_i^{\text{Left}} = \mathbf{u}^{V_2} - \mathbf{u}^{V_1}$ **Bottom to Top nodeset:** $\mathbf{u}_i^{\text{Top}} - \mathbf{u}_i^{\text{Bottom}} = \mathbf{u}^{V_4} - \mathbf{u}^{V_1}$ **Rear to Front nodeset:** $\mathbf{u}_i^{\text{Rear}} - \mathbf{u}_i^{\text{Front}} = \mathbf{u}^{H_1} - \mathbf{u}^{V_1}.$	**X-Y plane:** $\mathbf{u}_i^{T_2} - \mathbf{u}_i^{T_4} = \mathbf{u}^{V_2} - \mathbf{u}^{V_1}$ $\mathbf{u}_i^{B_2} - \mathbf{u}_i^{B_4} = \mathbf{u}^{V_2} - \mathbf{u}^{V_1}$ $\mathbf{u}_i^{T_4} - \mathbf{u}_i^{B_4} = \mathbf{u}^{V_4} - \mathbf{u}^{V_1}$ **Y-Z plane:** $\mathbf{u}_i^{T_3} - \mathbf{u}_i^{T_1} = \mathbf{u}^{H_1} - \mathbf{u}^{V_1}$ $\mathbf{u}_i^{B_3} - \mathbf{u}_i^{B_1} = \mathbf{u}^{H_1} - \mathbf{u}^{V_1}$ $\mathbf{u}_i^{T_1} - \mathbf{u}_i^{B_1} = \mathbf{u}^{V_4} - \mathbf{u}^{V_1}$ **X-Z plane:** $\mathbf{u}_i^{M_3} - \mathbf{u}_i^{M_4} = \mathbf{u}^{V_2} - \mathbf{u}^{V_1}$ $\mathbf{u}_i^{M_2} - \mathbf{u}_i^{M_1} = \mathbf{u}^{V_2} - \mathbf{u}^{V_1}$ $\mathbf{u}_i^{M_4} - \mathbf{u}_i^{M_1} = \mathbf{u}^{H_1} - \mathbf{u}^{V_1}$	$\mathbf{u}^{H_3} - \mathbf{u}^{V_3} = \mathbf{u}^{H_1} - \mathbf{u}^{V_1}$ $\mathbf{u}^{H_2} - \mathbf{u}^{V_2} = \mathbf{u}^{H_1} - \mathbf{u}^{V_1}$ $\mathbf{u}^{H_4} - \mathbf{u}^{V_4} = \mathbf{u}^{H_1} - \mathbf{u}^{V_1}$ $\mathbf{u}^{V_3} - \mathbf{u}^{V_4} = \mathbf{u}^{V_2} - \mathbf{u}^{V_1}.$

For each load case a single load file must be created. For 300 load cases we will have 300 load files and 300 jobs will be executed.

In the load file there are different descriptions

```
** STEP: Loading
**
*Step, name=Loading, nlgeom=YES, inc=500000, unsymm=YES, solver=ITERATIVE
*Static
1, 250, 1e-3, 1.0
```

The first number is initial time increment. Second number is time period of the load step. The third number is minimum time increment, and the last number is maximum time increment.

```
** Name: Load Type: Stress BC
*Cload
V2,1, -0.3752356825122646
V4,2, -0.5460770134887019
H1,3, 53.75659709123606
H1,2, 0.0
V2,3, 0.0
V4,1, 0.0
**
```

In this part the loads are described. As seen the type of the load is stress Boundary condition and Cload means that the load is a concentrated load.

V2,1	σ_{xx}
V4,2	σ_{yy}
H1,3	σ_{zz}
H1,2	σ_{zy}
V2,3	σ_{xz}
V4,1	σ_{yx}

```
** FIELD OUTPUT: F-Output-1
**
*Output, field
*Node Output
U,RF,CF,COORD
*Element Output, directions=YES
SDV156,SDV157,SDV158
SDV159,SDV160,SDV161
S,LE
** HISTORY OUTPUT: H-Output-1
**
```

In this part the field output is defined.

Node output indicates the requested nodal filed out puts.

U	Displacements
RF	Reaction forces
CF	Point loads and moments
COORD	Location of the nodes in current config

The element output indicates the element field outputs. As can be seen here SDV 156-161 are solution dependent variables and in this case based on their definition in UMAT they denote plastic strains obtained from deformation gradient.

S and LE are standard element integration point outputs from Abaqus. S is stress components and LE is logarithmic strain components.

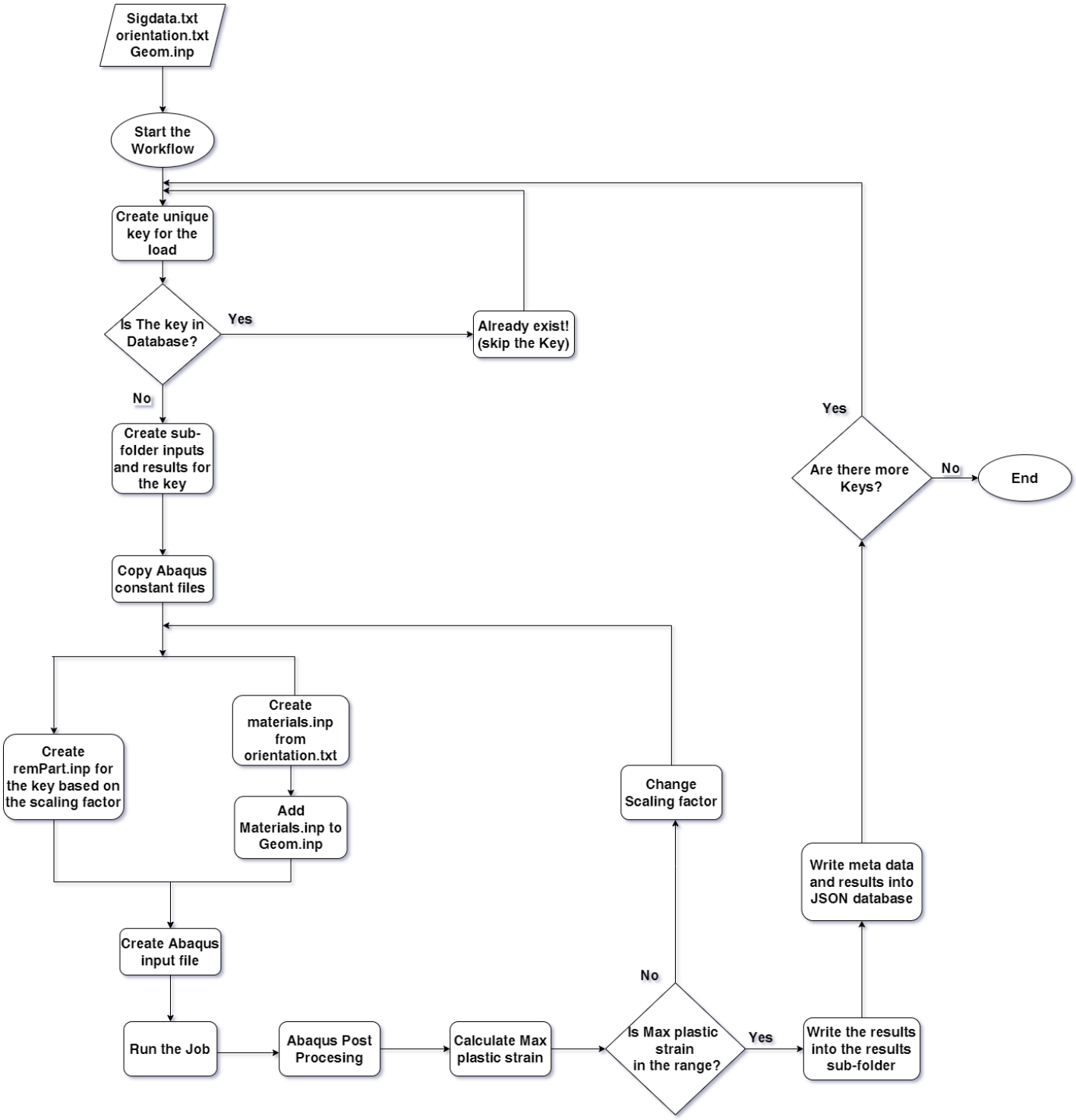
The load cases should be provided in a text file. and the designed workflow will generate load cases in this described format by going through the load file line by line. In each line there should be 6 numbers showing the unit stresses in 6-dimensional stress space.

3. Running Jobs

Usually for starting the workflow and running the jobs only 3 input files need to be generated and the rest can be reused and modified by the designed workflow. The orientation.txt the sig.txt and the geometry input file must be placed in Abaqus_Temp_Files folder and all the other files must be placed in the Abaqus_Constant_Files folder.

By starting the workflow, a unit key name will be created for each load case. the load file is created. Using the geometry file, the material file, and this load file a final input file for Abaqus is prepared and the job is executed.

For each generated key after running the job the related meta data will be written automatically to a JSON file and based on the post processing script the result stresses and strains will be calculated and saved.


Sigdata.txt
orientation.txt
Geom.inp
Start the Workflow
Create unique key for the load
Is The key in Database?
Yes
Already exist! (skip the Key)
No
Create sub-folder inputs and results for the key
Copy Abaqus constant files
Create materials.inp from orientation.txt
Add Materials.inp to Geom.inp
Create remPart.inp for the key based on the scaling factor
Create Abaqus input file
Run the Job
Abaqus Post Procesing
Calculate Max plastic strain
Is Max plastic strain in the range?
No
Change Scaling factor
Yes
Write the results into the results sub-folder
Write meta data and results into JSON database
Are there more Keys?
Yes
No
End