School of Engineering
Brown University

ENGN 1750: Advanced Mechanics of Solids

ABAQUS PYTHON SCRIPTING TUTORIAL

This tutorial will take you all the steps required to

(1) Read an ABAQUS output database with python (optional advanced topic); and

(2) Automate a parameter study using ABAQUS with a python script (optional advanced topic).

This tutorial uses the CAE database created in the ABAQUS tutorial – you must complete that tutorial before you can attempt this one.

Using Python to read and process data from an output database

Python can automate many ABAQUS pre-and post-processing operations. There are several ways to start the ABAQUS python scripting environment. You can run it from the ABAQUS command menu (you can find it in the Start menu). Open the start menu, then change the directory to the ABAQUS working directory (use the DOS cd command). Check that you can see the impact_simulation.odb file. You can start ABAQUS python with either of the following commands:

- abaqus python is the standard way, and does not require a license token, but only allows you to access a basic set of python modules
- abaqus viewer –noGUI will allow you to load modules needed to process an odb
- abaqus cae –noGUI will allow you to execute any ABAQUS/CAE operation – from model definition to postprocessing – through python

Try abaqus viewer –noGUI now (if the command doesn't work, check the path environment variable for your PC)

You are now in the python interpreter of ABAQUS. You can load the odb as follows:

```
>>> from odbAccess import openOdb
>>>odb = openOdb('Impact-Simulation.odb')
```

You can examine the top-level contents of the odb with

```
>>>dir(odb)
```

and you can use the python 'print' command to look at the odb in more detail. Try the following:

```
>>> print odb.jobData
>>> print odb.steps
>>> print odb.steps['Impact']
>>> print odb.steps['Impact'].timePeriod
```

As you can see, the python interpreter can access just about everything about an ABAQUS simulation (but it is not easy to figure out exactly how variables you might be interested in have been labeled, and the manuals – at least those that I can get to work, which are only for old ABAQUS versions - are virtually useless)

A useful operation for accessing history variables is

```
>>> step1 = odb.steps['Impact']
>>> print step1.historyRegions.keys()
```

This tells you the names that ABAQUS has assigned to any node or element sets you used in history output requests. The 'node' (which is probably named SPHERE-1.*nn* where *nn* is a number) is the reference node for the sphere.

To see what data are available for this node, you can use
```
>>> region = step1.historyRegions['Node SPHERE-1.nn']
```
(use the name of the reference node in the previous step)
```
>>> print region.historyOutputs.keys()
```
This tells you the variable names that were requested when the Step was defined in ABAQUS/CAE

To extract the data, you can use
```
>>> u2data = region.historyOutputs['U2'].data
>>> print u2data
```

You can then use most python commands to operate on the data (but ABAQUS is often using old versions of python, and also doesn't have python packages like Numpy installed)

Exit the python interpreter with
```
>>> quit()
```

The commandline python interpreter can be useful as a quick way to check variable names, but for most practical purposes it's better to store an abaqus python script in a file. For example you can cut and paste the short script below into a text editor and save it as process_impact.py extension. You can run the script from the abaqus command window
```
abaqus viewer –noGUI process_impact.py
```

```
#
#     Script to read data from odb file and save it to a csv file
#
from odbAccess import *
import sys

odb = openOdb('Impact-Simulation.odb')


step1 = odb.steps['Impact']
# Edit the line below to enter the correct node number
region = step1.historyRegions['Node SPHERE-1.nn']

displacementData = region.historyOutputs['U2'].data

nn = len(displacementData)
dispFile = open('disp.csv','w')
for i in range(0, nn):
     dispFile.write('%10.4E,%10.4E \n'%(displacementData[i][0], displacementData[i][1]))
dispFile.close()
```

Python scripts are particularly helpful to automate parametric studies – you can write a script to change a variable in an ABAQUS/CAE model, re-run the calculation, and save the result of the calculation to a file. For example, we will put together a simple script to repeatedly run ABAQUS with different sphere masses, and save a file storing the data necessary to plot the dimensionless contact time $t_c V_0 / R$ and the maximum normalized impact force $F_{\max} / (\mu R^2)$ as a function of the dimensionless group $\mu R^3 / (m V_0^2)$.

The first step is to work out how to change the mass using a python script. To do this, re-open ABAQUS/CAE, and change the mass of the sphere, as follows.

a. Open the Property module
b. Select the Sphere part
c. Use Special>Inertia>Manager to open the inertia manager
d. Select the sphere inertia in the manager and press Edit
e. Change the sphere mass to 0.1 and press OK
f. Save the model database and exit CAE

Now, navigate to the abaqus working directory, and open the file called abaqus.rpy with a text editor. You will see a python script that accomplishes all the changes you just made to the model database. In particular, the line

```
mdb.models['Model-1'].parts['Sphere'].engineeringFeatures.inertias['Inertia-1'].setValues(mass=0.1)
```

changes the sphere mass. We can wrap a script around this to vary the mass, run a simulation, and store the data.

You can cut and paste the script below (or download a file from the EN175 website) into a text file named impact_parameter_study.py

You can run it in an ABAQUS command window with abaqus cae –noGUI impact_parameter_study.py

```
#
#   Python script to run parametric study
#
# Run this from an abaqus command window with abaqus cae -noGUI impact_parameter_study.py
# Make sure that the cae model database file is in the ABAQUS working directory
#
from abaqus import *
from abaqusConstants import *
from caeModules import *
from odbAccess import *
import sys
openMdb('Sphere-film-impact.cae')

# Sphere radius, film shear modulus, and impact velocity
# We could read these from the mdb as well, of course, but simpler to hard code
R = 0.05
mu = 1
V0 = 0.05
minmass = 0.01
maxmass = 0.15
nmasses = 6

paramvals = []
tcvals = []
fmaxvals = []

# Loop over different values of sphere mass
for i in range (0,nmasses):
     m = minmass + (maxmass-minmass)*i/(nmasses-1)
     dimGroup = mu*R**3/(m*V0**2)
     paramvals.append(dimGroup)
     mdb.models['Model-1'].parts['Sphere'].engineeringFeatures.inertias['Inertia-1'].setValues( \
     mass=m)

     thisJob = mdb.jobs['Impact-Simulation']
     thisJob.submit(consistencyChecking=OFF)
     thisJob.waitForCompletion()

     odb = openOdb('Impact-Simulation.odb')

#  Extract the acceleration-time data for the sphere.  You may need to change the name of the node

     step1 = odb.steps['Impact']
     region = step1.historyRegions['Node SPHERE-1.65']
     acceldata = region.historyOutputs['A2'].data

     nn = len(acceldata)
#     Find the contact time by searching the accels from the end backwards to find the first nonzero
value
     for n in reversed(xrange(nn)):
           if (acceldata[n][1]>0.000001):
              tcontact = acceldata[n][0]
              break

#    Find the max contact force as the max value of mass*acceleration
     maxforce = m*max([aval[1] for aval in acceldata])

#    Store data for printing later
     tcvals.append(tcontact*V0/R)
     fmaxvals.append(maxforce/(mu*R**2))

     odb.close()

resultsFile = open('contactData.csv','w')
for i in range(0, nmasses):
     resultsFile.write('%10.4E,%10.4E,%10.4E\n'% (paramvals[i],tcvals[i],fmaxvals[i]))
resultsFile.close()
```