HDF5: A new approach to Interoperability in Finite Element tools.

Anshuman S Bhadauria[1]*

Abstract
In this paper we discuss a methodology to export data from Abaqus Output Database directly to HDF5 containers. We give a brief overview of the HDF5 data model, and why is it suitable to be used with Finite Element problems. Then we introduce the Abaqus ODB and it's detailed structure. Following that we explain the procedure to import external libraries to Abaqus-Python and discuss the scripting procedure. Finally we discuss an example HDF5 that is created using our scripts and the future scope of the work.

Keywords
Abaqus — Python — HDF5 — Interoperability — h5py — Data transfer — Finite Elements

[1] *International Center for Numerical Methods in Engineering (CIMNE), CIMNE Building C1, Campus Nord UPC, Barcelona, Spain, 08034*
*__Corresponding author__: anshumanbhadauria8@gmail.com

Contents

Introduction

With the tremendous amount of research done in the field of *Numerical Methods*, several new algorithms and software tools (academic and commercial) have been developed in the past decades. The advent of software tools has not only made the application of the methods easier, but it has also drastically increased the number of users and applications of these numerical methods. The independent development of these tools by companies, research centers, individuals has made available these tools for everybody at no to low costs.

These *Finite Element Tools* can exchange data with each other through some of the well known data formats like VTK,STL, XDMF etc. [1] some of the fields like Mesh, geometry, loads, etc. Each one these formats has been standardized to transfer a particular kind of data related to the problem. In this paper, I would like to propose a method to customize a neutral format HDF5, for data import and export in Abaqus using a Python API for HDF5.

1. Methodology

1.1 HDF5 Data Model

HDF5[2] is a unique technology suite that makes possible the management of extremely large and complex data collections. The HDF5 technology suite includes: A versatile data model that can represent very complex data objects and a wide variety of metadata.

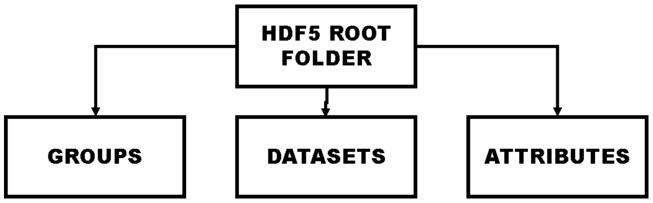


Figure 1. HDF5 Data Model

The HDF5 data model comprises of *Groups,Datasets,&*

Attributes. The groups give HDF the hierarchical data structure which helps in data organization in an effective manner. Whereas the Datasets are the multidimensional arrays storing simulation data. Last but not the least, the attributes, which are analogous 'labels', help make the datasets self descriptive. These HDF objects can be used to customize a HDF5 file according to ones needs. In this paper, we use a Python API called *h5py* to establish a data exchange from *Abaqus Output Database* to HDF5 & HDF5 to *Abaqus Model Databse*. The Python API[3] gives us access to the methods available in the HDF5 class. These methods further allow us to edit, read, write the HDF files from Abaqus-Python interface. The process of importing external library like *h5py* can be really tricky because the Abaqus version of Python is compiled specially for Abaqus and comes in with outdated or no versions of python libraries like *NumPy, Six, Cython* which are essential for *h5py* to work.

1.2 Importing Essential Python Libraries

Although the process is bit involved but it is possible to import *h5py* in Abaqus-Python by doing following steps:

- Install a copy of Python version which is same as the version which comes along with Abaqus. In our case, i.e. *Abaqus v6.14* comes with *Python v2.7.3*.
- Secondly, a copy of *h5py* needs to be installed on this Python version. This can be done by installing *pip* https://pip.readthedocs.org/en/stable/, but we recommend that instead of installing Python and all the related separately install a scientific package of python which comes pre-installed with all the required libraries. We used [4] which comes pre-installed with *h5py*.
- Finally, the folders of Python libraries *h5py, Six, Cython, NumPy* should be copied from site-packages folder of your Python installation, in our case from Anaconda installation folder, to the Abaqus-Python site-packages.

Also care needs to be taken, when copying the folders to the Abaqus site-packages as it already contains a folder named *NumPy* . The version already present in Abaqus is *NumPy v1.6.2* which is not supported by *h5py* libraries, additionally Abaqus GUI will not run if you replace the inbuilt *NumPy*. So we need to keep both *NumPy* folders in Abaqus site-packages, the problem can be solved by renaming one of them depending which one we need to use. To conclude, use *NumPy v1.6.2* when Abaqus GUI is needed, and use *NumPy 1.7.2*, which came with the scientific package when Abaqus NO GUI is used.

1.3 Abaqus Output Database(ODB)

Abaqus ODB is a perfect example of a *Hierarchical Structured Database*, as can be seen in 2. In order to retrieve the data from ODB, some knowledge of Python classes & methods and list & dictionary comprehension is useful. TO get a higher perspective of the database, a Python library *TextRepr* can be invoked. *TextRepr* gives a prettified view, which is human readable, of the ODB.

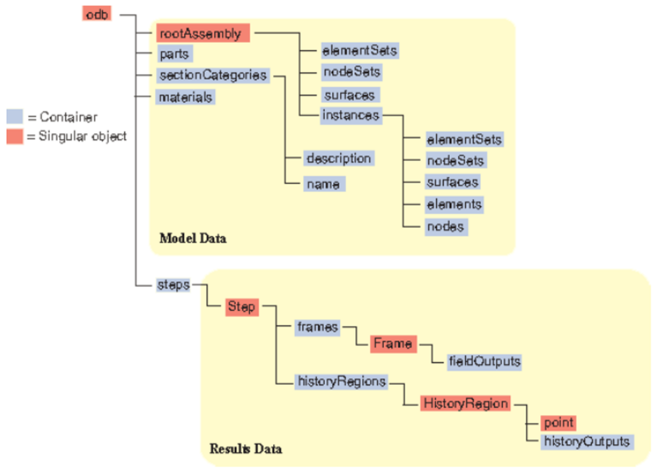


Figure 2. Hierarchical Structure of Abaqus Output Database

Abaqus Python sports some additional data structures other than normal data structures like arrays, lists, tuples and dictionaries. Although the complete details of these data structures **??** is out of the scope of this paper, it can be comfortably said that comprehensions of these new data structures like *Repository Object, OdbMeshNodeArray Object* etc. don't differ much from normal data structures.

1.4 Scripting for Data Comprehension

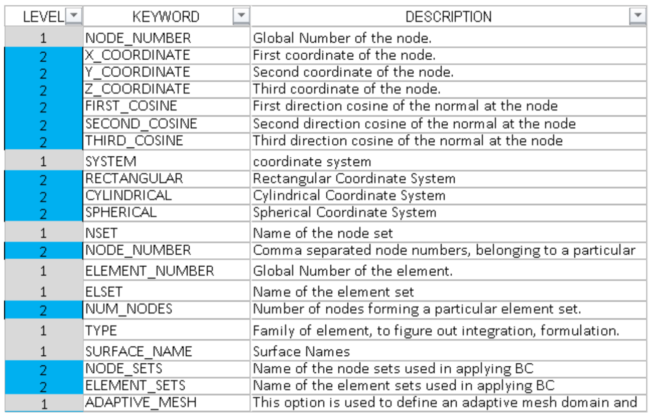
Once we have successfully imported the required libraries into Abaqus-Python, following the procedure defined in previous section we can start retrieving data from the ODB. In the following few steps we demonstrate how a *NodeSet or ElementSet* can be exported to HDF5 directly from Abaqus.

- First, we import the relevant libraries and open the the database which we want to export, followed by creation of a new HDF5 file.

- We can transfer data from the arrays, and repository objects of the ODB to the Groups and Datasets of the newly created HDF5 file using Python scripting, for eg. see [Appendix].

1.5 Metadata

Metadata is one of the most essential part of the standardization process of HDF5 files. This ”data about data”, i.e. the right metadata can make the datasets self-descriptive. Since all the finite element tools are built on an underlying programming language, by using an API we can establish exchange rules between tools and HDF5. Finally when we have sufficient metadata from wide variety of problems and tools we can think of a global template which can enable perfect interoperability.



LEVEL	KEYWORD	DESCRIPTION
1	NODE_NUMBER	Global Number of the node.
2	X_COORDINATE	First coordinate of the node.
2	Y_COORDINATE	Second coordinate of the node.
2	Z_COORDINATE	Third coordinate of the node.
2	FIRST_COSINE	First direction cosine of the normal at the node
2	SECOND_COSINE	Second direction cosine of the normal at the node
2	THIRD_COSINE	Third direction cosine of the normal at the node
1	SYSTEM	coordinate system
2	RECTANGULAR	Rectangular Coordinate System
2	CYLINDRICAL	Cylindrical Coordinate System
2	SPHERICAL	Spherical Coordinate System
1	NSET	Name of the node set
2	NODE_NUMBER	Comma separated node numbers, belonging to a particular
1	ELEMENT_NUMBER	Global Number of the element.
1	ELSET	Name of the element set
2	NUM_NODES	Number of nodes forming a particular element set.
1	TYPE	Family of element, to figure out integration, formulation.
1	SURFACE_NAME	Surface Names
2	NODE_SETS	Name of the node sets used in applying BC
2	ELEMENT_SETS	Name of the element sets used in applying BC
1	ADAPTIVE_MESH	This option is used to define an adaptive mesh domain and

Figure 3. An example of keywords summarized for Thermo Mechanical problems.

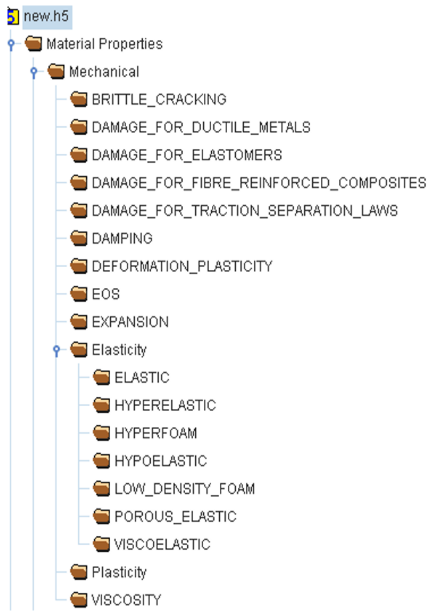
The collection of metadata keywords is done using documentation of Abaqus v6.14. Each keyword represents quantity which is used to describe a part of the problem i.e. loading, nodes, geometry etc and will help to uniquely identify a quantity in a file.

2. Results & Discussion

The customization of HDF5 containers was done for thermo-mechanical problems, using *h5py*, a Python API for HDF5 files. The metadata for the problem was collected from Abaqus v6.14, and a methodology for the export of data from Abaqus to HDF5 containers was established in Fig.4.

2.1 Future Work

The hierarchical structure of ODB and MDB(Model Database) are similar, and they serve the two main purposes in Abaqus software. The methodology we devised



```
new.h5
  Material Properties
    Mechanical
      BRITTLE_CRACKING
      DAMAGE_FOR_DUCTILE_METALS
      DAMAGE_FOR_ELASTOMERS
      DAMAGE_FOR_FIBRE_REINFORCED_COMPOSITES
      DAMAGE_FOR_TRACTION_SEPARATION_LAWS
      DAMPING
      DEFORMATION_PLASTICITY
      EOS
      EXPANSION
      Elasticity
        ELASTIC
        HYPERELASTIC
        HYPERFOAM
        HYPOELASTIC
        LOW_DENSITY_FOAM
        POROUS_ELASTIC
        VISCOELASTIC
      Plasticity
      VISCOSITY
```

Figure 4. An HDF5 file customized using metadata from Abaqus

to extract data from ODB, can also be used to write data to a MDB, directly from HDF5 files after some alterations in the scripts. The ODB can be understood as a read-only mode, whereas MDB can be considered as read-write mode.

Acknowledgments

This project is a part of the InterOperability group which is working under the Integrated Computational Materials Engineering expert group (ICMEg). The ICMEg project is supported by the European Commission under grant FP7 606711. [1][5]

References

[1] J.S. . George. Whitebook “communication standards in icme”. *ICMEg*, 2, 2015.

[2] HDF-Group. https://www.hdfgroup.org/hdf5/.

[3] HDF-For-Python. http://www.h5py.org/.

[4] Continuum-Analytics. https://repo.continuum.io/archive/.

[5] 3DS. Abaqus documentation v6.14.

```
#This script exports Element Set, Node Set and Nodal results from ODB to HDF5
 ↪  file.
from odb import *
import h5py
import numpy as np
#Open an existing Abaqus Odb%
odb=openOdb('indentation_axi1.odb')
#Create a new hdf5 file%
k=h5py.File('nava.hdf5','w')
#Create groups to match the hierarchy of the odb or as desired%
grp1k=k.create_group('NodesK')
grp2k=k.create_group('ElementsK')
grp3k=k.create_group('ResultsK')
#Create datasets associated with these groups%
dataset1k=grp2k.create_dataset('elemConnK',(400,4),dtype='i')
dataset2k=grp1k.create_dataset('nodeCoorK',(400,3),dtype='f')
dataset3k=grp3k.create_dataset('resultsK',(400,1))

#Comprehend the Elements connectivity data from Odb into the hdf5 file%
for i in range(0,np.size(odb.rootAssembly.instances['I_INDENTER']....
elementSets['ALL_ELEMENTS'].elements)):
    for j in range(np.size(odb.rootAssembly.instances['I_INDENTER']....
    elementSets['ALL_ELEMENTS'].elements[1].connectivity)):
        dataset1k[i,j]=odb.rootAssembly.instances['I_INDENTER']....
        elementSets['ALL_ELEMENTS'].elements[i].connectivity[j]
dataset1k.attrs['title']="Connectivity of elements"

#Comprehend the Nodes coordinates data from Odb into the hdf5 file%
for g in range(np.size(odb.rootAssembly.instances['I_INDENTER']....
nodeSets['ALL_NODES'].nodes)):
        for l in range(0,np.size(odb.rootAssembly.instances['I_INDENTER']....
        nodeSets['ALL_NODES'].nodes[1].coordinates)):
                dataset2k[g,l]=odb.rootAssembly.instances['I_INDENTER']....
                nodeSets['ALL_NODES'].nodes[g].coordinates[l]

#Comprehend a field output request, using a list comprehension because
#Abaqus repository is not iterable.#
requestedOutputs=[]
requestedOutputs=odb.steps['LOADING'].frames[-1].fieldOutputs.keys()
#for key in range(len(requestedOutputs)):
for node in range(np.size(odb.rootAssembly.instances['I_INDENTER']....
nodeSets['ALL_NODES'].nodes)):
        dataset3k[node]=odb.steps['LOADING'].frames[-1].fieldOutputs['U']....
        values[node].magnitude
```

```
# This script transfers an Abaqus .mat file to HDF5 containers#
from odb import *
import h5py
import numpy as np

#Open an existing Abaqus Odb%
odb=openOdb('discbrake_sst_axi.odb')

#Create a new hdf5 file%
k=h5py.File('mat.hdf5','w')

#Create groups to match the hierarchy of the odb or as desired%
grp1=k.create_group('Material')
grp2=grp1.create_group('Mechanical')
grp3=grp2.create_group('Elastic')
grp4=grp2.create_group('Plastic')
grp5=grp2.create_group('Expansion')

#Create datasets associated with these groups
dataset1=grp3.create_dataset('elasticity',(40,3),dtype='f')
dataset2=grp4.create_dataset('hardening',(40,3),dtype='f')
dataset3=grp5.create_dataset('expansion',(40,2),dtype='f')

#Comprehend the Elasticity data from Odb into the hdf5 file%
for i,j in enumerate(odb.materials['FONTE'].elastic.table):
        dataset1[i]=j
dataset1.attrs['title']="Elasticity model"

#Comprehend the Nodes coordinates data from Odb into the hdf5 file%
for k,l in enumerate(odb.materials['FONTE'].plastic.table):
        dataset2[k]=l
dataset2.attrs['title']="Plasticity model"

#Comprehend a field output request, using a list comprehension because Abaqus
↪ repository is not iterable.#
for m,n in enumerate(odb.materials['FONTE'].expansion.table):
        dataset3[m]=n
dataset3.attrs['title']="Expansion model"
```