HDF5: A new approach to interoperability in Finite Element tools

Anshuman Singh
UPC Barcelona

What Is Interoperability?

Definition of INTEROPERABILITY: ability of a system to work with or use the parts or equipment of another system.

Few of the many finite element software available today !!!

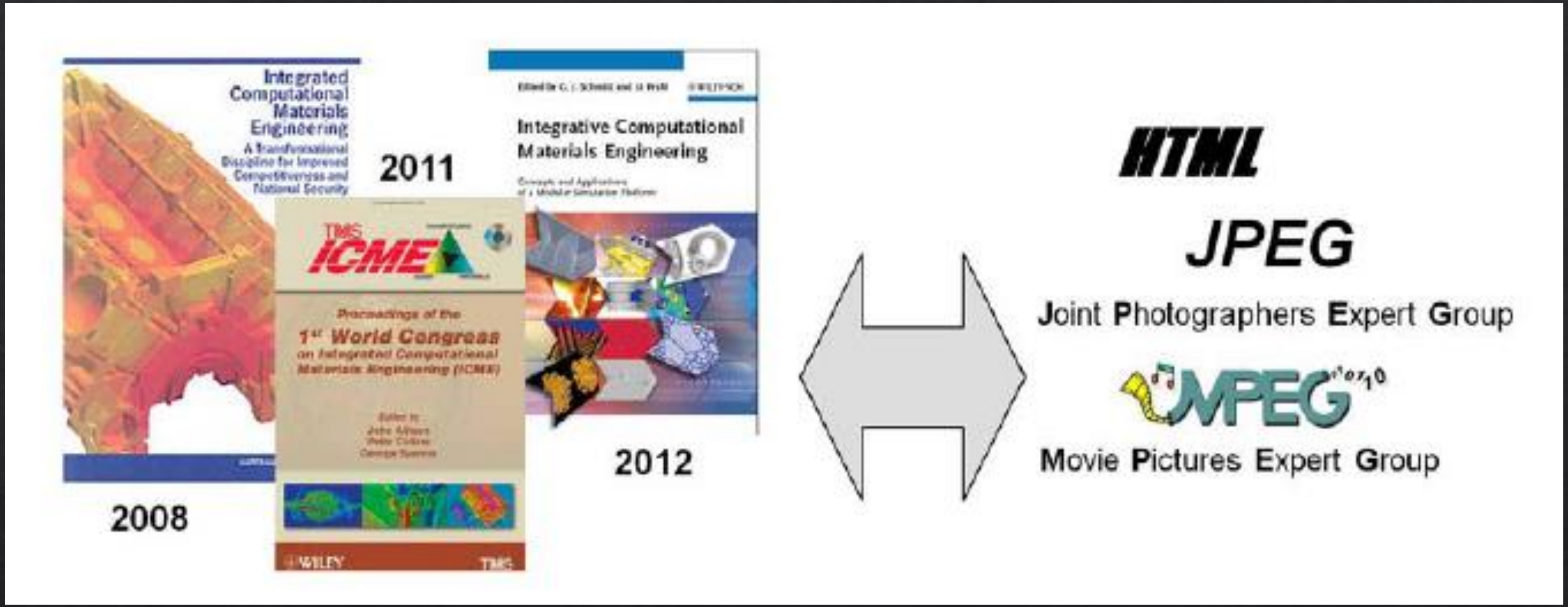
CFD-FASTRAN
NASTRAN
COMSOL
DIANA
COMSOL
ADINA
AutoForm
FEDEM
ANSYS
KRATOS
Advance
FREE-FEM
LS-DYNA
MOOSE
FEATFLOW
ABAQUS
ALGOR
Altair
DUNE
GiD
OPENFOAM
ANSA
Agros2D

So to put it in simple words,

"We want to be enable seamless communication between different tools. "

How do we plan to tackle the InterOp problem?

Bigger Picture !!!


2008
2011
2012
Integrated Computational Materials Engineering
Integrative Computational Materials Engineering
ICME
1st World Congress on Integrated Computational Materials Engineering (ICME)
HTML
JPEG
Joint Photographers Expert Group
MPEG
Movie Pictures Expert Group

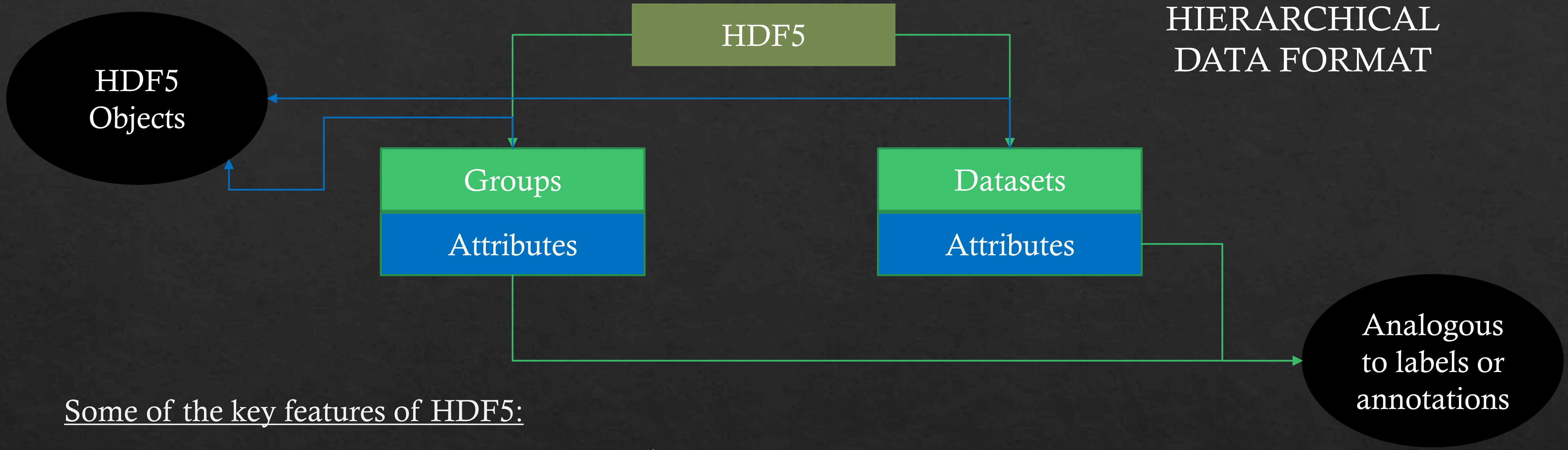
*ICMEg : Integrative Computational Materials Engineering expert group

“Develop a common language by standardizing and generalizing data formats for the exchange of simulation results. “

My Goal

- ❖ Enable import of external libraries to Abaqus-Python.
- ❖ Define a procedure to import/export data from Abaqus ODB using HDF5 standard.
- ❖ Create an elaborate list of keywords, which will be used as metadata for communication.
- ❖ To explain you guys, what the hell am I talking about.

What is HDF5?



HIERARCHICAL DATA FORMAT

Some of the key features of HDF5:

Unlimited size, extensibility, and portability	Practically no size limit, APIs across all major prog. languages.
General data model	Simple but versatile data model to match individual needs.
Unlimited variety of datatypes	Efficient mechanism for describing data.
Flexible, efficient I/O	Supports parallel data access, more customized schemes possible.
Flexible data storage	Supports compression, chunking for efficient storage and retrieval.

Example Of HDF5 Performance

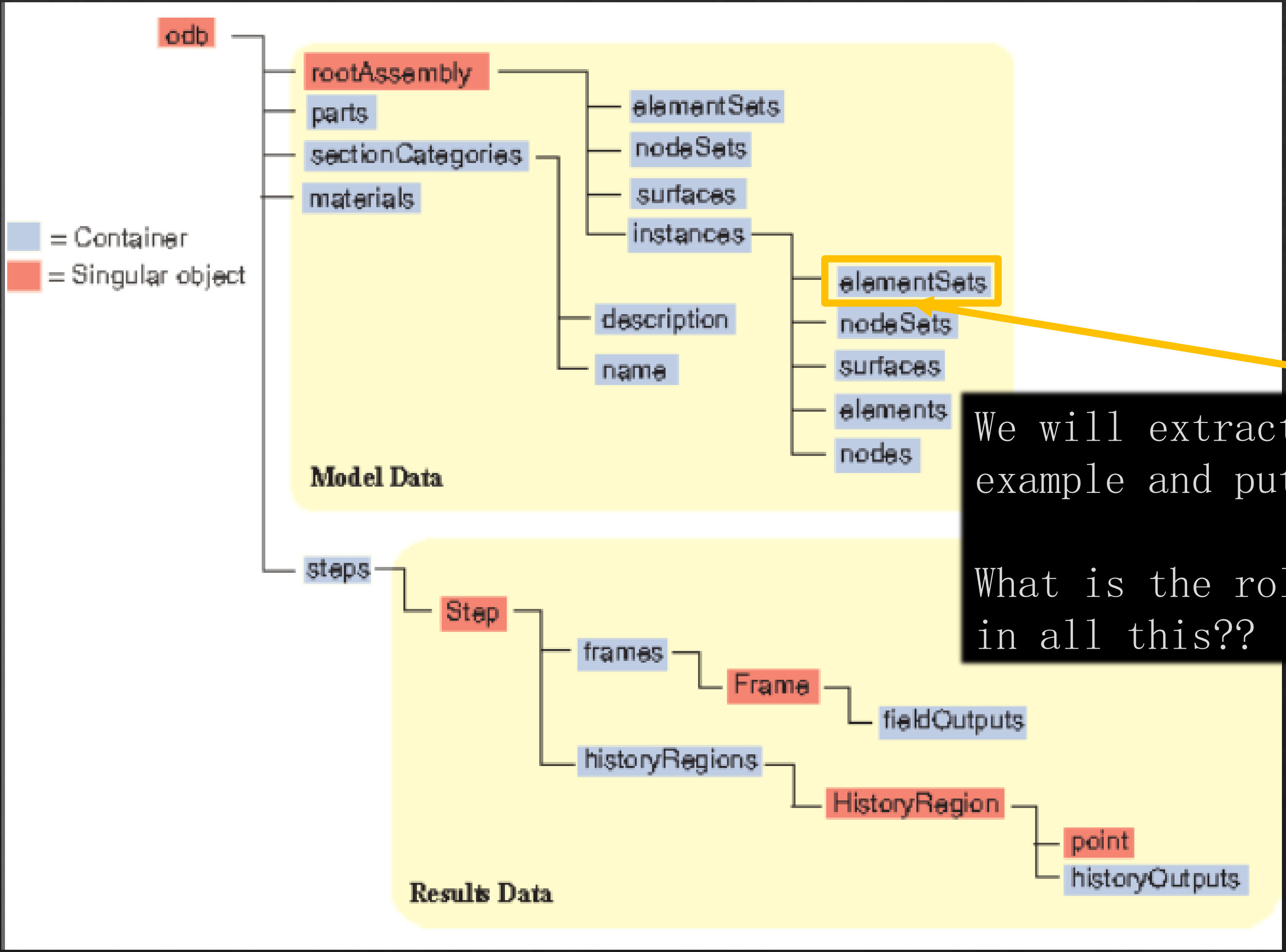
The question the scientists wanted to answer is why some particles from the sun are accelerated to very high energy and form auroras or wreak havoc on electronics.

In total, 10 separate trillion-particle datasets, each ranging between 30 and 42 terabytes in size, were written as HDF5 files at rates reaching 90 percent of maximum and a sustained rate of 27 out of a possible 35 gigabytes per second.

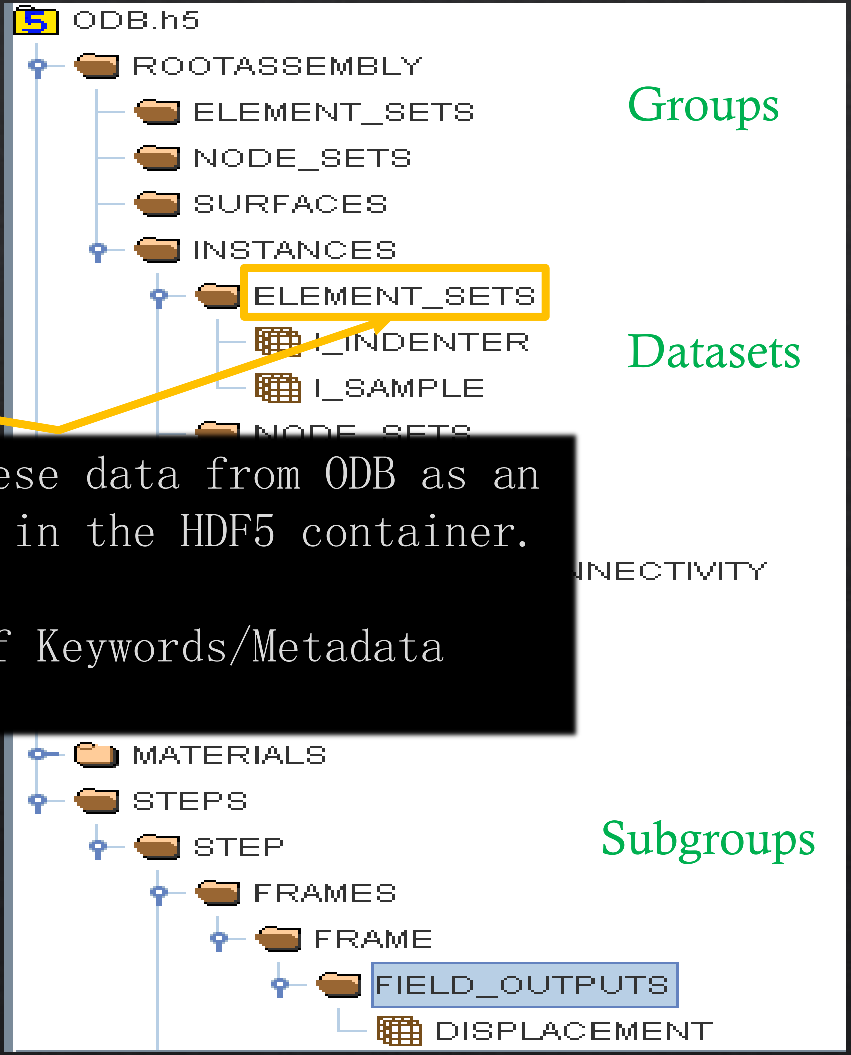
"HDF5 is on the trajectory toward exascale computing i.e. a billion billion computations per second.."

Hierarchical Structural Comparison

Abaqus Output Database Structure



HDF5 Container Structure



We will extract these data from ODB as an example and put it in the HDF5 container.

What is the role of Keywords/Metadata in all this??

This is what the output file from Abaqus/Python will look like, along with groups, subgroups, datasets and attributes.

STRUCTURE(Internal View): Abaqus ODB

As an example, we access the rootAssembly and extract data of element connectivity and node coordinates.

Structure of the problem.

```
({'analysisTitle':  'Indentation',
  'closed': False,
  'customData': None,
  'description': 'DDB object',
  'diagnosticData': ({'analysisErrors': 'OdbSequenceAnalysisError object',
                      'analysisWarnings': 'OdbSequenceAnalysisWarning object',
                      'isXplDoublePrecision': False,
                      'jobStatus': JOB_STATUS_COMPLETED_SUCCESSFULLY,
                      'jobTime': 'OdbJobTime object',
                      'numDomains': 1,
                      'numberOfAnalysisErrors': 0,
                      'numberOfAnalysisWarnings': 3,
                      'numberOfSteps': 3,
                      'numericalProblemSummary': 'OdbNumericalProblemSummary
                               object',
                      'steps': 'OdbSequenceDiagnosticStep object'}),
  'isReadOnly': True,
  'jobData': ({'analysisCode': ABAQUS_STANDARD,
              'creationTime': 'Sun Oct 25 00:04:21 Romance Daylight Time 2015',
               'machineName': '',
               'modificationTime': 'Sun Oct 25 00:04:45 Romance Daylight Time 2015',
               'name': 'C:/Users/Anshuman/indentation_axil.odb',
               'precision': SINGLE_PRECISION,
               'productAddOns': 'tuple object',
               'version': 'Abaqus/Standard Student Edition 6.14-2'}),
  'materials': {'INDENTER_MAT': 'Material object',
                'SAMPLE_MAT': 'Material object'},
  'name': 'C:/Users/Anshuman/indentation_axil.odb',
  'parts': {'P_INDENTER': 'Part object',
            'P_SAMPLE': 'Part object'},
 'path': 'C:/Users/Anshuman/indentation_axil.odb',
  'profiles': {},'readInternalSets': False,
  'rootAssembly': ({'connectorOrientations': 'ConnectorOrientationArray object',
                    'datumCsyses': 'Repository object',
                    'elementSets': 'Repository object',
                    'elements': 'OdbMeshElementArray object',
                    'instances': 'Repository object',
                    'name': 'ASSEMBLY',
                    'nodeSets': 'Repository object',
                    'nodes': 'OdbMeshNodeArray object',
                    'pretensionSections': 'OdbPretensionSectionArray object',
                    'rigidBodies': 'OdbRigidBodyArray object',
                    'sectionAssignments': 'SectionAssignmentArray object',
                    'surfaces': 'Repository object'}),
  'sectionCategories': {'solid < INDENTER_MAT >': 'SectionCategory object',
                         'solid < SAMPLE_MAT >': 'SectionCategory object'},
  'sections':           {'Section-ASSEMBLY_I_INDENTER_ALL_ELEMENTS':
  'sectorDefinition': None,
  'steps':              {'LOADING': 'OdbStep object',
                        'PRELOADING': 'OdbStep object',
                        'UNLOADING': 'OdbStep object'},
  'userData':           ({'annotations': 'Repository object',
                        'xyDataObjects': 'Repository object'})})
```

Indepth view of the rootAssembly

```
({'connectorOrientations': [],
  'datumCsyses': {},
  'elementSets': {' ALL ELEMENTS': 'OdbSet object',
                  'ErrElemAnisotropicMaterial': 'OdbSet object',
                  'WarnElemDistorted': 'OdbSet object'},
  'elements': [],
  'instances': {'I_INDENTER': 'OdbInstance object',
                'I_SAMPLE': 'OdbInstance object'},
  'name': 'ASSEMBLY',
  'nodeSets': {' ALL NODES': 'OdbSet object',
               'REFERENCE_POINT_I_INDENTER      296': 'OdbSet object'},
  'nodes': [],
  'pretensionSections': [],
  'rigidBodies': [],
  'sectionAssignments': [],
  'surfaces': {}})
```

The OdbSet objects are like arrays which can be accessed like we access normal arrays.

🟨	Text in orange represents Level 1
🟩	Text in green represents data we access as example.

INTEGRATION: How to connect Abaqus ODB With HDF5?

```
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('random.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))
dataset2k=grp1k.create_dataset('nodeCoorK',(400,3))
dataset3k=grp3k.create_dataset('resultsK',(400,9))

#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]

#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,key]=odb.steps['LOADING'].frames[-1].fieldOutputs[requestedOutputs[key]].values[node].magnitude
```

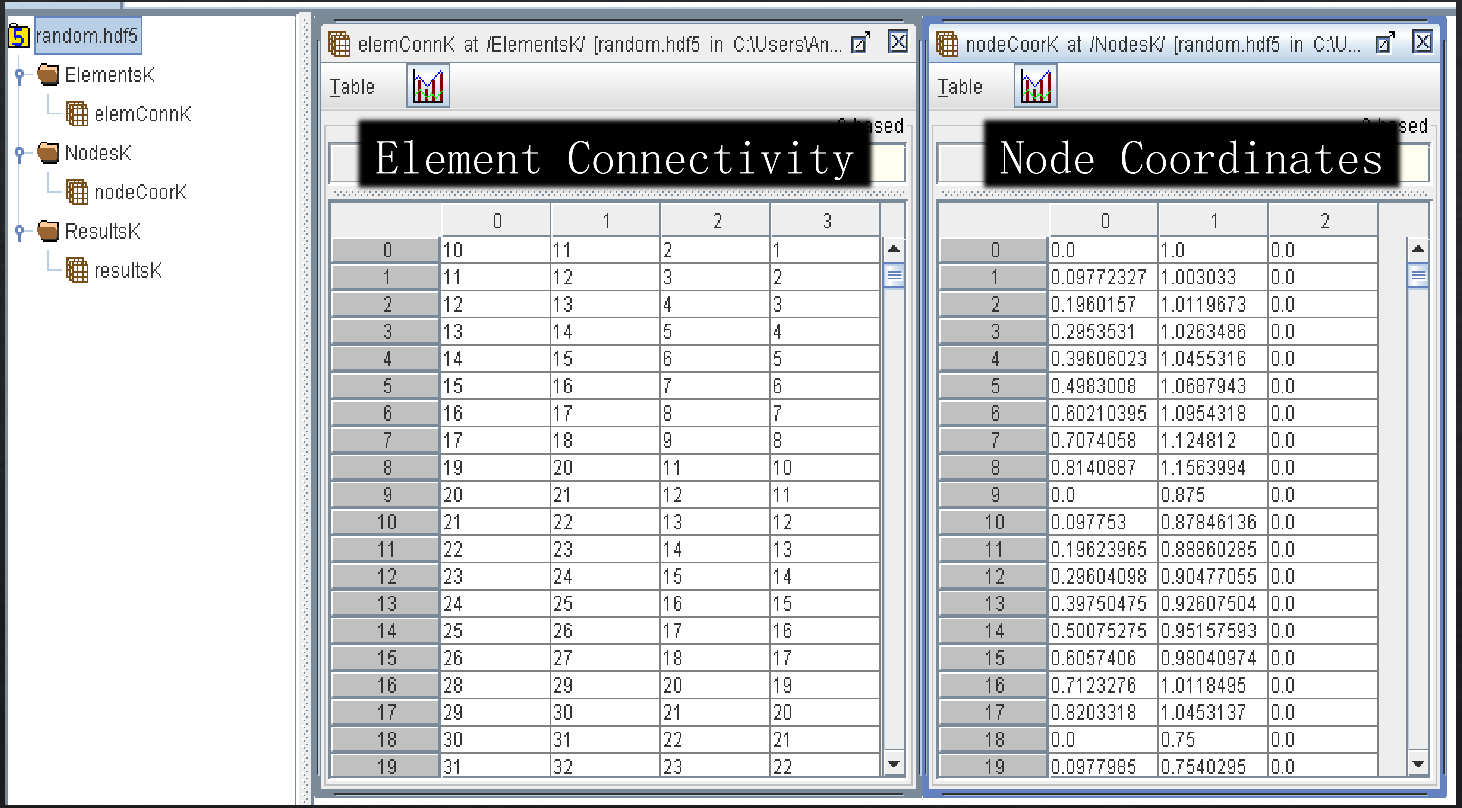
The biggest complication.

❖ This is the script written, to be executed in Abaqus-Python, to produce the example HDF5 file.

h5py is the Python API for HDF5.

RESULT: Successful Export of Data to HDF5

An HDF5 file generated from within Abaqus, showing Element Connectivity , Node Coordinates, and some results from one of the analytical steps..


random.hdf5
ElementsK
elemConnK
NodesK
nodeCoorK
ResultsK
resultsK
elemConnK at /ElementsK/ [random.hdf5 in C:\Users\An...
nodeCoorK at /NodesK/ [random.hdf5 in C:\U...
Table

Element Connectivity

	0	1	2	3
0	10	11	2	1
1	11	12	3	2
2	12	13	4	3
3	13	14	5	4
4	14	15	6	5
5	15	16	7	6
6	16	17	8	7
7	17	18	9	8
8	19	20	11	10
9	20	21	12	11
10	21	22	13	12
11	22	23	14	13
12	23	24	15	14
13	24	25	16	15
14	25	26	17	16
15	26	27	18	17
16	28	29	20	19
17	29	30	21	20
18	30	31	22	21
19	31	32	23	22

Node Coordinates

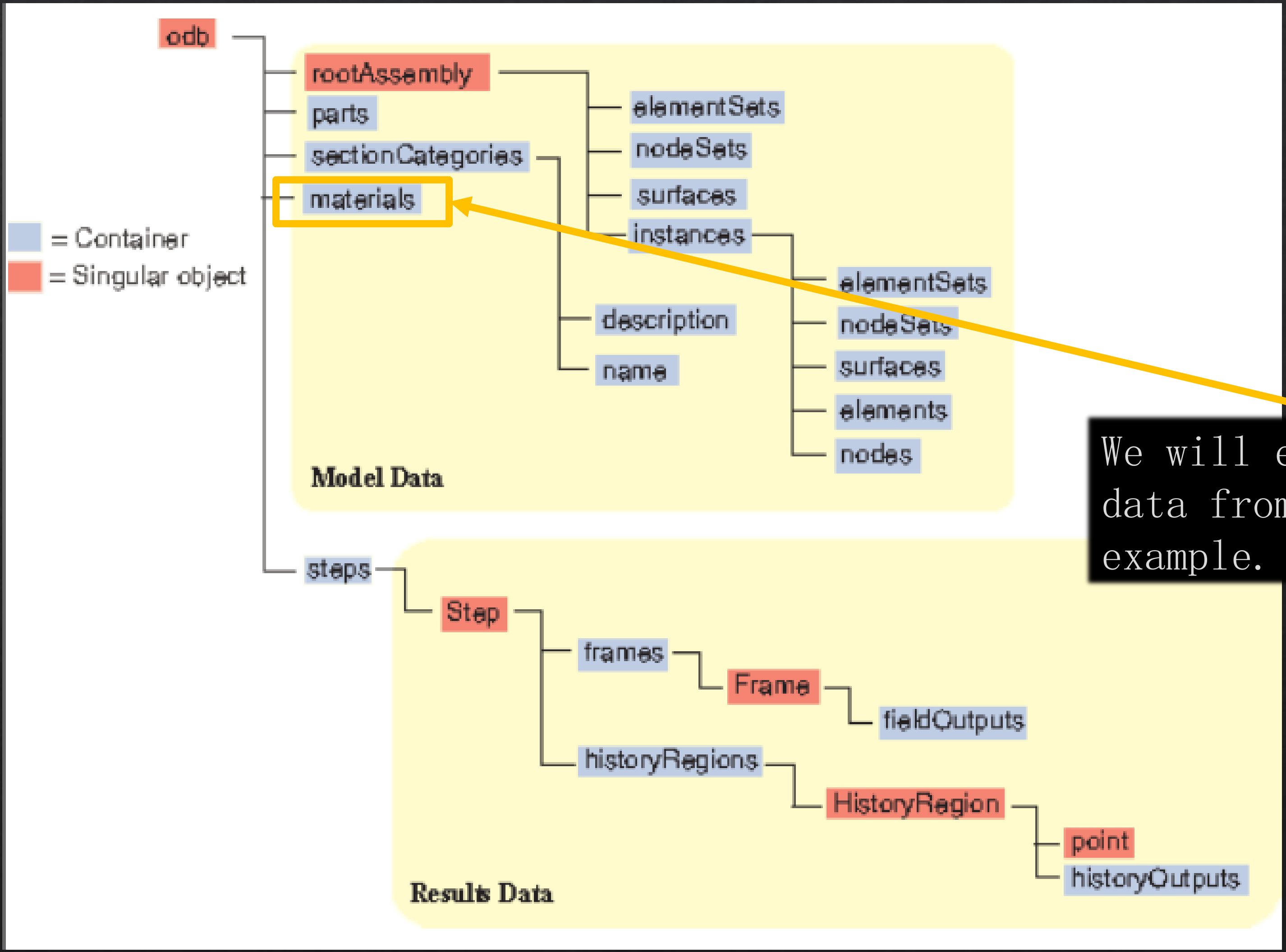
	0	1	2
0	0.0	1.0	0.0
1	0.09772327	1.003033	0.0
2	0.1960157	1.0119673	0.0
3	0.2953531	1.0263486	0.0
4	0.39606023	1.0455316	0.0
5	0.4983008	1.0687943	0.0
6	0.60210395	1.0954318	0.0
7	0.7074058	1.124812	0.0
8	0.8140887	1.1563994	0.0
9	0.0	0.875	0.0
10	0.097753	0.87846136	0.0
11	0.19623965	0.88860285	0.0
12	0.29604098	0.90477055	0.0
13	0.39750475	0.92607504	0.0
14	0.50075275	0.95157593	0.0
15	0.6057406	0.98040974	0.0
16	0.7123276	1.0118495	0.0
17	0.8203318	1.0453137	0.0
18	0.0	0.75	0.0
19	0.0977985	0.7540295	0.0

Finally, other post processing tools which have superior plotting and data manipulating capabilities like R and Python, can be used For further post-processing.

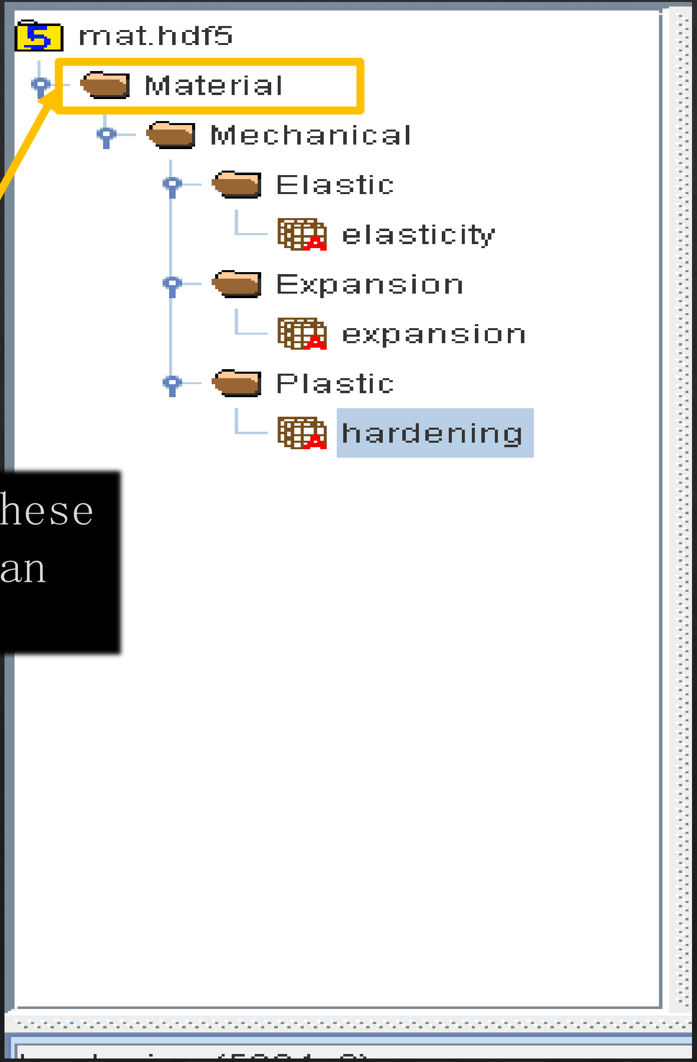
Last but not the least, all the advantages of The HDF5 format, like chunking, Compression, Attributes, Fancy Indexing are naturally inherited.

Material (*.mat) File Export

Abaqus Output Database Structure



Equivalent HDF5 File Structure



We will extract these data from ODB as an example.

This is what the output file from Abaqus/Python will look like, along with groups, subgroups, datasets and attributes.

STRUCTURE(Internal View): Abaqus ODB

As an example, we access the rootAssembly and extract data of element connectivity and node coordinates.

Structure of the problem.

```
({’analysisTitle’: ’Cross-section of a disc’,
  ’closed’: False,
  ’customData’: None,
  ’description’: ’DDB object’,
  ’diagnosticData’: ({’analysisErrors’: ’OdbSequenceAnalysisError object’,
                      ’analysisWarnings’: ’OdbSequenceAnalysisWarning object’,
                      ’isXplDoublePrecision’: False,
                      ’jobStatus’: JOB_STATUS_COMPLETED_SUCCESSFULLY,
                      ’jobTime’: ’OdbJobTime object’,
                      ’numDomains’: 1,
                      ’numberOfAnalysisErrors’: 0,
                      ’numberOfAnalysisWarnings’: 0,
                      ’numberOfSteps’: 1,
                      ’numericalProblemSummary’: ’OdbNumericalProblemSummary object’,
                      ’steps’: ’OdbSequenceDiagnosticStep object’}),
  ’isReadOnly’: True,
  ’jobData’: ({’analysisCode’: ABAQUS_STANDARD,
               ’creationTime’: ’Sun Nov 01 12:27:41 Romance Standard Time 2015’,
               ’machineName’: ’’,
               ’modificationTime’: ’Sun Nov 01 12:27:42 Romance Standard Time 2015’,
               ’name’: ’C:/Users/Anshuman/discbrake_sst_axi.odb’,
               ’precision’: SINGLE_PRECISION,
               ’productAddOns’: ’tuple object’,
               ’version’: ’Abaqus/Standard Student Edition 6.14-2’}),
  ’materials’: {’FONTE’: ’Material object’},
  ’name’: ’C:/Users/Anshuman/discbrake_sst_axi.odb’,
  ’parts’: {’PART-1’: ’Part object’},
  ’path’: ’C:/Users/Anshuman/discbrake_sst_axi.odb’,
  ’profiles’: {},
  ’readInternalSets’: False,
  ’rootAssembly’: ({’connectorOrientations’: ’ConnectorOrientationArray object’,
                    ’datumCsyses’: ’Repository object’,
                    ’elementSets’: ’Repository object’,
                    ’elements’: ’OdbMeshElementArray object’,
                    ’instances’: ’Repository object’,
                    ’name’: ’Assembly-1’,
                    ’nodeSets’: ’Repository object’,
                    ’nodes’: ’OdbMeshNodeArray object’,
                    ’pretensionSections’: ’OdbPretensionSectionArray object’,
                    ’rigidBodies’: ’OdbRigidBodyArray object’,
                    ’sectionAssignments’: ’SectionAssignmentArray object’,
                    ’surfaces’: ’Repository object’}),
  ’sectionCategories’: {’solid < FONTE >’: ’SectionCategory object’},
  ’sections’: {’Section-DISC’: ’HomogeneousSolidSection object’},
  ’sectorDefinition’: None,
  ’steps’: {’Step-1’: ’OdbStep object’},
  ’userData’: ({’annotations’: ’Repository object’,
                ’xyDataObjects’: ’Repository object’})})
```

In-depth view of the Materials

```
{’FONTE’: ({’description’: ’’,
            ’elastic’: ({’dependencies’: 0,
                         ’moduli’: LONG_TERM,
                         ’noCompression’: OFF,
                         ’noTension’: OFF,
                         ’table’: ’tuple object’,
                         ’temperatureDependency’: ON,
                         ’type’: ISOTROPIC}),
            ’expansion’: ({’dependencies’: 0,
                           ’table’: ’tuple object’,
                           ’temperatureDependency’: ON,
                           ’type’: ISOTROPIC,
                           ’userSubroutine’: OFF,
                           ’zero’: 0.0}),
            ’materialIdentifier’: ’’,
            ’name’: ’FONTE’,
            ’plastic’: ({’dataType’: HALF_CYCLE,
                         ’dependencies’: 0,
                         ’hardening’: KINEMATIC,
                         ’numBackstresses’: 0,
                         ’rate’: OFF,
                         ’strainRangeDependency’: OFF,
                         ’table’: ’tuple object’,
                         ’temperatureDependency’: ON}),
```

The tuple objects are accessed exactly like we would access a nested dictionary object in normal Python.

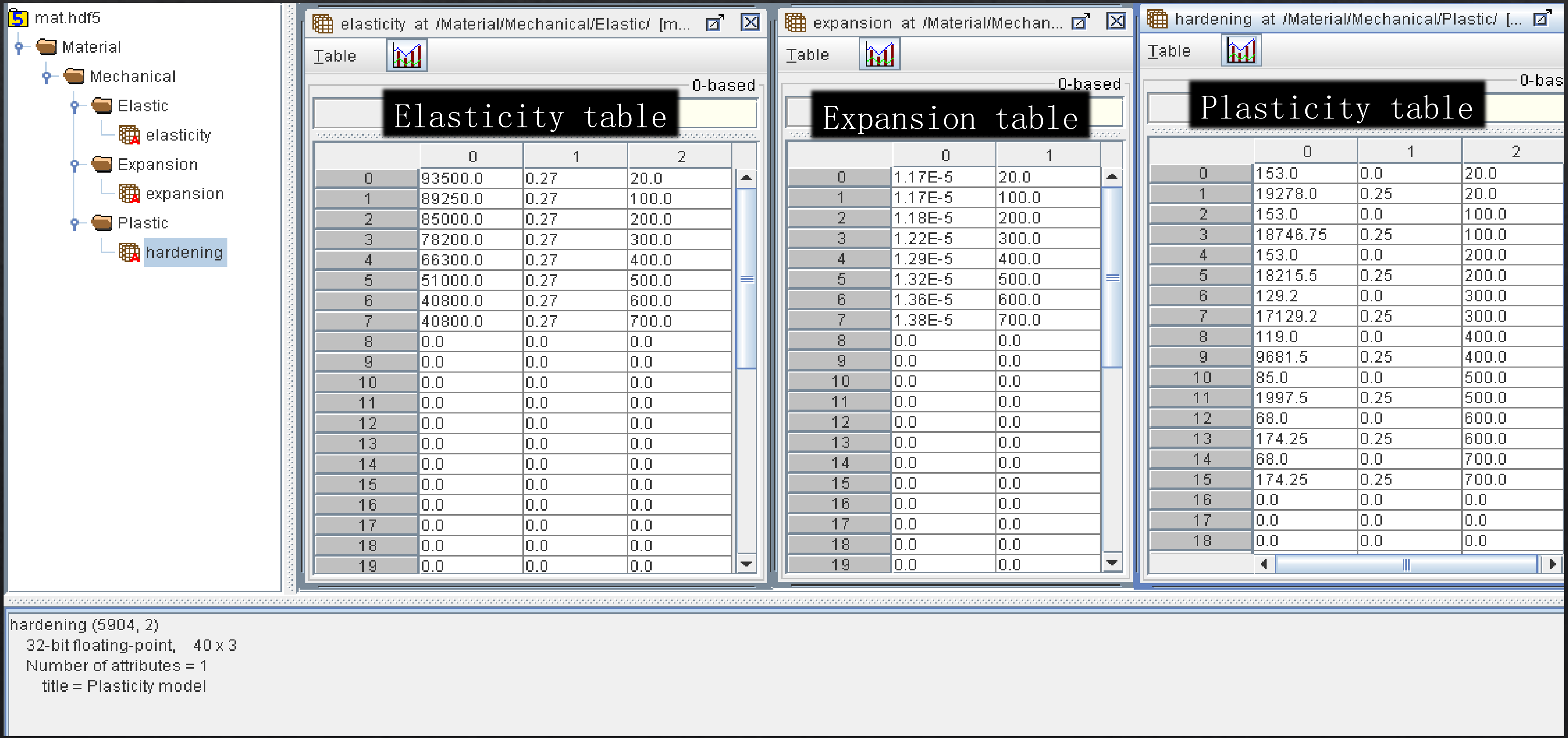
INTEGRATION: How to connect Abaqus ODB With HDF5?

```
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"
```

❖ This is the script written, to be executed in Abaqus-Python, to produce the material HDF5 file.

RESULT: Successful Export of Data to HDF5

An HDF5 file generated from Abaqus-Python, showing Material Elasticity, Plasticity and Expansion, tables.



```
mat.hdf5
  Material
    Mechanical
      Elastic
        elasticity
      Expansion
        expansion
      Plastic
        hardening
```

elasticity at /Material/Mechanical/Elastic/ [m...

Table

0-based

Elasticity table

	0	1	2
0	93500.0	0.27	20.0
1	89250.0	0.27	100.0
2	85000.0	0.27	200.0
3	78200.0	0.27	300.0
4	66300.0	0.27	400.0
5	51000.0	0.27	500.0
6	40800.0	0.27	600.0
7	40800.0	0.27	700.0
8	0.0	0.0	0.0
9	0.0	0.0	0.0
10	0.0	0.0	0.0
11	0.0	0.0	0.0
12	0.0	0.0	0.0
13	0.0	0.0	0.0
14	0.0	0.0	0.0
15	0.0	0.0	0.0
16	0.0	0.0	0.0
17	0.0	0.0	0.0
18	0.0	0.0	0.0
19	0.0	0.0	0.0

expansion at /Material/Mechan...

Table

0-based

Expansion table

	0	1
0	1.17E-5	20.0
1	1.17E-5	100.0
2	1.18E-5	200.0
3	1.22E-5	300.0
4	1.29E-5	400.0
5	1.32E-5	500.0
6	1.36E-5	600.0
7	1.38E-5	700.0
8	0.0	0.0
9	0.0	0.0
10	0.0	0.0
11	0.0	0.0
12	0.0	0.0
13	0.0	0.0
14	0.0	0.0
15	0.0	0.0
16	0.0	0.0
17	0.0	0.0
18	0.0	0.0
19	0.0	0.0

hardening at /Material/Mechanical/Plastic/ [...

Table

0-bas

Plasticity table

	0	1	2
0	153.0	0.0	20.0
1	19278.0	0.25	20.0
2	153.0	0.0	100.0
3	18746.75	0.25	100.0
4	153.0	0.0	200.0
5	18215.5	0.25	200.0
6	129.2	0.0	300.0
7	17129.2	0.25	300.0
8	119.0	0.0	400.0
9	9681.5	0.25	400.0
10	85.0	0.0	500.0
11	1997.5	0.25	500.0
12	68.0	0.0	600.0
13	174.25	0.25	600.0
14	68.0	0.0	700.0
15	174.25	0.25	700.0
16	0.0	0.0	0.0
17	0.0	0.0	0.0
18	0.0	0.0	0.0

```
hardening (5904, 2)
  32-bit floating-point,    40 x 3
  Number of attributes = 1
    title = Plasticity model
```

Future

- To develop the codes for complete transfer of the problem data (including Boundary, Material, Loads etc., along with metadata.
- Enable communication with other tools for eg. KRATOS.
- Extended this procedure to enable export of images produced by Abaqus through HDF5.

THANK YOU !!!