General_Geometry_Thickness_Analysis.py
User Instructions

Written by: Jack Consolini
Updated on: 2022-12-06

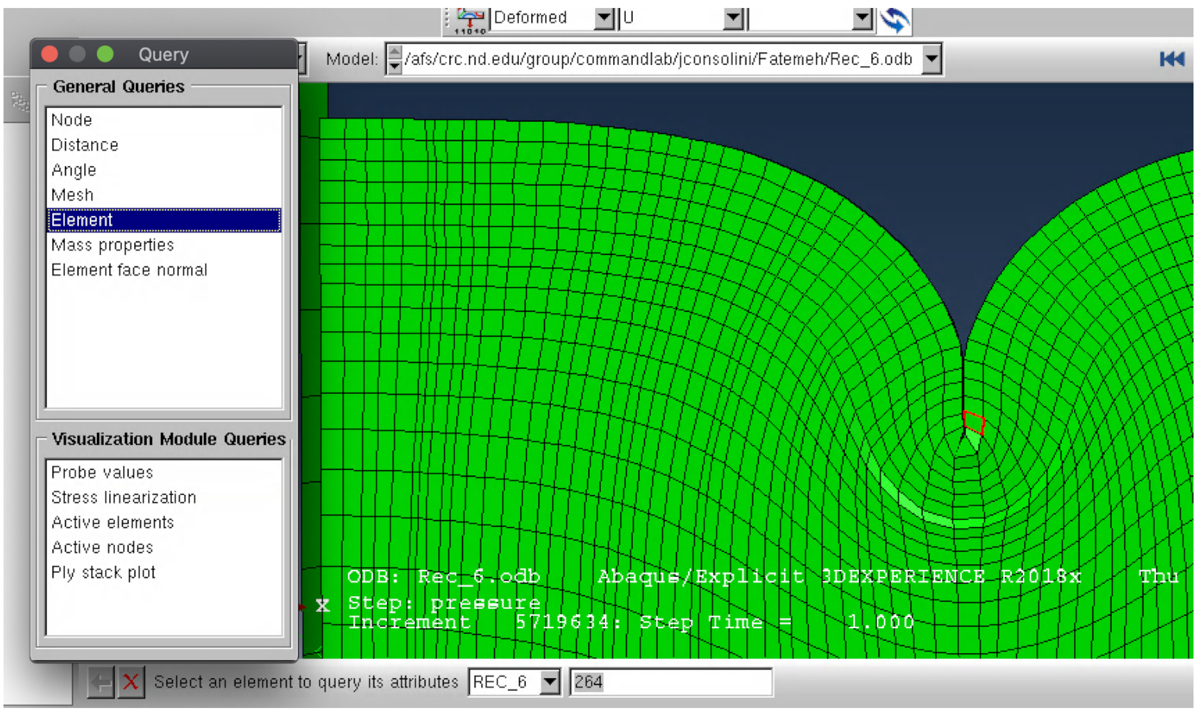
Step 1: Extracting element numbers and their associated nodes

a) Run the code ***General_Geometry_Thickness_Analysis.py***, but comment out the final line and make sure the function `element_info(odbName())` is the only function running:

```
element_info(example_simulation())
# solve(example_simulation())
```

b) The created .csv file contains element numbers and their associated nodes. Open that .csv file, named: ***odbName_information_for_gyri_and_sulci_selection.csv*** and use the element-node information to select the bounds for your element sections.

c) To select your element section with element number bounds, first need to visually pick out the regions of interest (likely the curved regions) in the Abaqus GUI. This is accomplished by: opening Abaqus → Querying the elements you want → and selecting the boundary elements of your desired region.



d) **(OPTIONAL)** Edit ***odbName_information_for_gyri_and_sulci_selection.csv*** with some indication of the element's you have selected for your sections. This is helpful for quality checking that ***General_Geometry_Thickness_Analysis.py*** created the correct element sections. Also save the **.csv** as a **.xlsm** so that your edits are not erased.

	A	B	C	D	E	F	G	H	I	J	K
230	204	447	6	1912	6	446	6	1917	6		
231	210	446	6	1917	6	445	6	1922	6		
232	216	445	6.4	1922	6.4	444	6.4	1927	6.4		
233	222	444	6.4	1927	6.4	443	6.4	1932	6.4		
234	228	443	6.4	1932	6.4	442	6.4	1937	6.4		
235	234	442	6.4	1937	6.4	441	6.4	1942	6.4		
236	235	1938	6.4	518	6.4	1943	6.4	519	6.4	closest to subcortex	
237	236	1939	6.8	1938	6.8	1944	6.8	1943	6.8		
238	237	1940	6.8	1939	6.8	1945	6.8	1944	6.8		
239	238	1941	6.8	1940	6.8	1946	6.8	1945	6.8		
240	239	1942	6.8	1941	6.8	1947	6.8	1946	6.8		
241	240	441	6.8	1942	6.8	440	6.8	1947	6.8	Sulci 1 (element 1, leftmost)	
242	241	1943	7.2	519	7.2	1948	7.2	520	7.2	closest to subcortex	
243	242	1944	7.2	1943	7.2	1949	7.2	1948	7.2		
244	243	1945	7.2	1944	7.2	1950	7.2	1949	7.2		
245	244	1946	7.2	1945	7.2	1951	7.2	1950	7.2		
246	245	1947	7.2	1946	7.2	1952	7.2	1951	7.2		
247	246	440	7.6	1947	7.6	439	7.6	1952	7.6	Sulci 1 (element 2)	
248	247	1948	7.6	520	7.6	1953	7.6	521	7.6	closest to subcortex	
249	248	1949	7.6	1948	7.6	1954	7.6	1953	7.6		
250	249	1950	7.6	1949	7.6	1955	7.6	1954	7.6		
251	250	1951	7.6	1950	7.6	1956	7.6	1955	7.6		
252	251	1952	8	1951	8	1957	8	1956	8		
253	252	439	8	1952	8	438	8	1957	8	Sulci 1 (element 3)	
254	253	1953	8	521	8	1958	8	522	8	closest to subcortex	
255	254	1954	8	1953	8	1959	8	1958	8		
256	255	1955	8	1954	8	1960	8	1959	8		
257	256	1956	8.4	1955	8.4	1961	8.4	1960	8.4		
258	257	1957	8.4	1956	8.4	1962	8.4	1961	8.4		
259	258	438	8.4	1957	8.4	437	8.4	1962	8.4	Sulci 1 (element 4)	
260	259	1958	8.4	522	8.4	1963	8.4	523	8.4	closest to subcortex	
261	260	1959	8.4	1958	8.4	1964	8.4	1963	8.4		
262	261	1960	8.8	1959	8.8	1965	8.8	1964	8.8		
263	262	1961	8.8	1960	8.8	1966	8.8	1965	8.8		
264	263	1962	8.8	1961	8.8	1967	8.8	1966	8.8		
265	264	437	8.8	1962	8.8	436	8.8	1967	8.8	Sulci 1 (element 5, rightmost)	
266	265	1963	8.8	523	8.8	1968	8.8	524	8.8	closest to subcortex	

e) Once you have determined the element bounds for a number of sections (Note, for proper comparison, element sections should be the same number of elements wide), you can then add the bounds to ***General_Geometry_Thickness_Analysis.py*** within a list, where the bounds should be input as:
   i) **Lower bound:** 0.5 less than the lowest element number
      Ex. if 235 is the lowest number, make the bound 234.5
   ii) **Upper Bound:** 0.5 higher than the highest element number
      Ex. if 264 is the lowest number, make the bound 264.5

With the bounds input for all the element sections, the remaining relevant information can be obtained from the odb file.

Step 2: Supply object class information

a) You will want to fill the relevant .odb file information into the object class. You should know this information from running your simulation/it can be obtained by opening the .odb file in the Abaqus GUI. The following is an image of the filled out class, unfortunately an example file cannot be provided because of privacy restrictions, however, the inputs should be fairly straight forward if you have your .odb file open in the Abaqus GUI. Note that inputs are case-sensitive.

```
class example_simulation:
    def __init__(self):
        self.odbName = 'Rec_6.odb'
        self.lamina = ['CORTEX']
        self.strainEnergy = 'ELSE'
        self.part = 'REC_6'
        self.step = 'pressure'
        self.elementTypeNumber = 'C3D8'
        self.adjustmentLower = [0]
        self.adjustmentUpper = [0]
        self.adjustmentUpperCoords = [True]
        self.connectivityLower = [5]
        self.connectivityUpper = [6]
        self.sectionList = ['SULCI1.1', 'GYRI1.1']
        self.xCoordLowerList = [234.5, 540.5]
        self.xCoordUpperList = [264.5, 570.5]
# This is used to get the state variable information and thickness across all frames
        self.odb = openOdb(self.odbName, readOnly=False)
        self.frames = self.odb.steps[self.step].frames
```

b) This is the information you would be putting into each category:

i) `self.odbName:` this would be the name of your .odb

ii) `self.lamina:` this would be the deformed elements, for the brain simulation in this example, it was labeled the "CORTEX"

iii) `self.strainEnergy:` This is the state variable, so for say for strain energy that the user solved for in the example simulation, the variable is defined by "ELSE"

iv) `self.part:` this is your part, in this case the same as the .odb file name

v) `self.step:` your step name

vi) `self.elementTypeNumber` = this is the element type, likely again this information can be obtained by querying any element in the deformed area. It will be identified with an element code as you see above.

vii) `self.adjustmentLower:` [0] *LEAVE THIS AS 0*

viii) `self.adjustmentLower:` [0] *LEAVE THIS AS 0*

ix) self.adjustmentUpperCoords: [TRUE] *LEAVE THIS AS TRUE*

x) `self.connectivityLower:` [5] *LEAVE THIS AS 5* This tells the code which nodes to take the distance of from the lower elements.

xi) `self.connectivityLower:` [6] *LEAVE THIS AS 6* This tells the code which nodes to take the distance of from the upper elements

xii) `self.sectionList:` These are the element section names that you created. You can have as many as you like as long as you have associated element bounds to them. For example, if the model had 3 gyri and 3 sulci in a brain folding simulations, you would put in ['Sulci1', 'Gyri1', 'Sulci2', 'Gyri2', 'Sulci3', 'Gyri3'].

xiii) `self.xCoordLowerList:` ***REALLY IMPORTANT: THIS IS WHERE YOU PUT IN THE LOWER BOUND OF YOUR ELEMENT SETS*** So here, if your first section was a sulci, you would put the lower element bound of your sulci section, and then do a comma and add the bound of the next gyral section.

xiv) `self.xCoordUpperList:` ***REALLY IMPORTANT: THIS IS WHERE YOU PUT IN THE UPPER BOUND OF YOUR ELEMENT SETS*** Same thing as for the lower bounds, just the upper bounds this time.

xv) `self.odb:`Do not modify

xvi) `self.frames:`Do not modify

c) With all of the information above put in, you should be able to then run the code successfully, you just need to make sure to uncomment the `solve(odbname())`

```
# element_info(example_simulation())
solve(example_simulation())
```

You will end up getting two output files, ***fileName_State_Variable_Output.csv*** and ***fileName_Lamina_Thickness_Output.csv.*** With the state variable information and thickness information respectively.

Step 3: Understanding output data

a) This is the way the output files will be organized for both the state variable file and the lamina thickness file. The way the columns work are:

Frames	Cortical Section	Section Type	Layer	Lamina Thickness or State variable
This is the frame number. Going from 0-final frame.	This is the section you created. Say you had created two sulci sections and two gyri, then you would first go through the SULCI1 section for all frames, then the GYRI1 section for all frames, and so on.	This indicates if the section you are currently on is a sulci or gyri. So basically, like if you were going through the frames of your SULCI1, it will identify it as a sulci	This tells you what layer you are working on, so like in your case the only layer you are working on is the cortex, but in my simulations I went through the 6 layers.	If it is lamina thickness, it is the **average thickness of the section** (like sulci or gyri) at that specific frame. Then if you are doing the state variable, in this case the strain energy, it would be the **TOTAL strain energy** at all frames for a specific section.

This is an example of how the output file might look:

General_Geometry_Thickness_Analysis.py

User Instructions

	A	B	C	D	E	F
1	Frame	Cortical Sect	Section Type	Layer	Lamina Thickness	
2	0	SULCI1.1	SULCI	CORTEX	2	
3	1	SULCI1.1	SULCI	CORTEX	2	
4	2	SULCI1.1	SULCI	CORTEX	2	
5	3	SULCI1.1	SULCI	CORTEX	2.00000048	
6	4	SULCI1.1	SULCI	CORTEX	2.00000024	
7	5	SULCI1.1	SULCI	CORTEX	2.00000135	
8	6	SULCI1.1	SULCI	CORTEX	2.0000031	
9	7	SULCI1.1	SULCI	CORTEX	2.00000596	
10	8	SULCI1.1	SULCI	CORTEX	2.00001025	
11	9	SULCI1.1	SULCI	CORTEX	2.00001534	
12	10	SULCI1.1	SULCI	CORTEX	2.00002209	
13	11	SULCI1.1	SULCI	CORTEX	2.00003075	