’’’

ExoToComsol

BSD 3-Clause License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

’’’

Documentation for ExoToComsol (version 0.1)

Farhan Rahman, James Y. Hartley

Introduction

ExoToComsol is a program written in Python for both way conversion between EXODUS and COMSOL file formats. The program has four main utilities:

a. Convert an entire FE model in EXODUS format to COMSOL file format:

   This requires calling the function exoToComsol()

b. Convert a user defined region on a FE model in EXODUS format to COMSOL file format:

   This requires calling the function exoToComsol_with_ROI()

c. Convert an entire FE model in COMSOL format to the EXODUS file format:

   This requires calling the function comsolToExo()

d. Convert a user defined region on a FE model in COMSOL format to the EXODUS file format:

   This requires calling the function comsolToExo_with_ROI()

   The source code contains example files (with '_driver' suffix) for each of the utilities mentioned above along with sample input and output files in the input/output folders (Figure 1).

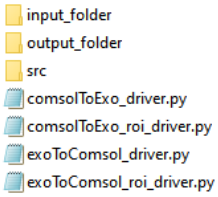

Figure 1: Source code folder for ExoToComsol program

Dependencies

Running the ExoToComsol program requires Python3.7 or above (https://www.python.org/), and SEACAS module (https://github.com/sandialabs/seacas) both of which are open source. Installation instructions of them can be found in the provided links.

Classes

ExoToCOMSOL program has two main classes 'Node' and 'Element_Tetrahedra', along with a 'util' class providing read/write functionalities for EXODUS and COMSOL file formats. COMSOL software supports multiple file formats (Table 1) from which the 'section-wise' file format was selected to work with during ExoToComsol program development. As shown in Table 1, 'section-wise' file format is not a native COMSOL format which is desired for open-source development and has most support for different features (volume mesh, surface mesh, etc.) among other alternate formats (spreadsheet and

grid). However, during ExoToComsol program development it was found that the sectionwise format does not support hex element export. Hence, currently ExoToComsol program only supports Tet elements.

Table 1: COMSOL file formats

File Format	Extension	Volume mesh	Surface mesh	Interpolation data import	Export
COMSOL native binary	.mphbin	Yes	Yes	No	Yes
COMSOL native text	.mphtxt	Yes	Yes	No	Yes
COMSOL sectionwise	.txt	Yes	Yes	Yes	Yes
COMSOL spreadsheet	.txt	No	No	Yes	Yes
COMSOL grid	.txt	Yes	No	Yes	Yes

Functions

Documentations for the functions in the different classes are shown in Figure 2, Figure 3 and Figure 4

Functions

comsolToExo(inputFolderPath, input_comsol_file_name, outputFolderPath, outputExodusFilename)
Outputs Exodus file from COMSOL file

Returns/writes an Exodus file for SIERRA code

comsolToExo_with_ROI(inputFolderPath, input_comsol_file_name, outputFolderPath, outputExodusFilename, bounds)
Outputs Exodus file from COMSOL file for user-defined region-of-interest (ROI) of the FE model

Returns/writes an Exodus file for SIERRA code

exoToComsol(inputFolderPath, inputExodusFilename, outputFolderPath, output_comsol_file_name, elem_type)
Outputs COMSOL file in section-wise format from Exodus file

Returns/writes a text file in section-wise format directly importable in COMSOL for mesh and simulation data

exoToComsol_with_ROI(inputFolderPath, inputExodusFilename, outputFolderPath, output_comsol_file_name, ouptut_file_extension, elem_type, bounds)
Outputs COMSOL file of user-defined region-of-interest (ROI) in section-wise format from Exodus file

Returns/writes a text file in section-wise format directly importable in COMSOL for mesh and simulation data of user-defined region-of-interest (ROI)

read_COMSOL_section_wise_data(inputFolderPath, input_comsol_file_name)
Reads COMSOL file in section-wise format to retrieve FE model information

Returns FE mesh and simulation data

write_sectionwise_file_for_COMSOL_input_full_mesh(path, filename, dimension, num_nodes, num_blk_elems, nodes_list, elem_type, elems_list)
writes COMSOL file in section-wise format for the full FE model

outputs text file

write_sectionwise_file_for_COMSOL_input_roi_cropped_mesh(path, filename, dimension, num_nodes, num_blk_elems, nodes_list, elem_type, elems_list)
writes COMSOL file in section-wise format for the user-defined region-of-interest (ROI) of the FE model

outputs text file

Figure 2: Functions in 'util' class:

Functions

create_elem(elem_conn, nodes_list)
Creates an element object from a list of node ids for an element, and a list of node objects

Note: This function assumes that the nodes are ordered in ascending order of node ids in the input list of node objects

Returns an element object

create_elem_from_unordered_nodes_list(elem_conn, nodes_list)
Creates an element object from a list of node ids for an element, and a list of node objects

Note: This function assumes that the nodes are not ordered in ascending order of node ids in the input list of node objects.
This function as currently implemented using filter() is slow for large list of nodes. To remedy this we can first create a list of nodes ordered by node ids.

Returns an element object

create_elems_list_from_elem_conn_list(elem_conn_entire_list, num_blk_elems, nodes_list)
Creates a list of element object from a list containing node ids in order to show element connectivity.

Returns a list of element objects

get_elem_conn_list_aft_roi_cropping(elems_list_roi_cropped)
Gets the element connectivity list which is a list of node ids after a cropping on the full model based on user-defined region-of-interest has been performed.

Note: Separte node ids before and after cropping are required to be generated since node ids must always start from 1 for COMSOL sectionwise file format.

get_elem_connectivity(lines_with_element_connectivity)
Gets element connectivity information from text

Returns list of node ids representing element connectivities.

get_element_id_array(numElems)
Creates element id array without requiring node object

Returns element id array

get_elems_list_roi_cropped(elems_list, node_ids_roi_cropped)
Creates a list of elements falling inside the user-defined region-of-interest of the model.

Returns: List of element objects

get_num_nodes_per_elem(lines_with_element_connectivity)
Gets number of nodes per element from input text

Returns number of nodes per element

Figure 3: Functions for Element_Tetrahedra class

Functions

create_nodes_list_from_coords(nodal_coords_tuple, nodal_temps_list)
Creates list of node objects from nodal coordinates and nodal simulation data

Returns list of node objects

get_coords_aft_roi_cropping(nodes_list_roi_cropped)
Gets coordinates of nodes inside user-defined region-of-interest

Returs arrays of x, y and z coordinates

get_new_node_id_after_roi_cropped(nodes_list_roi_cropped)
Gets new node ids after nodes are identifed inside ROI

Note: new node ids are needed since for COMSOL sectwise format to work node ids must start from 1 and end at the number of nodes

Returns: none

get_nodal_coords(lines_with_nodal_coords)
Gets nodal coordinates from text

Returns lists for x, y and z coordinates

get_nodal_sim_data(lines_with_nodal_sim_data)
Gets nodal simulation data

Returns list of nodal simulation data

get_nodal_sim_data_roi_cropped(nodes_list_roi_cropped)
Gets nodal simulation data over the user-defined region-of-interest

Returns list of nodal simulation data

get_node_id_array(numNodes)
Gets array of node ids from number of nodes and without using node object

Returns list of node ids

get_nodes_list_roi_cropped(x_coord_lower_bound, x_coord_upper_bound, y_coord_lower_bound, y_coord_upper_bound, z_coord_lower_bound, z_coord_upper_bound, nodes_list)
Gets list of nodes in the user-defined region-of-interest (ROI)

Returns list of nodes in the ROI

Figure 4: Functions for Node class

Acknowledgements

Funding was provided as part of the Durable Module Materials Consortium (DuraMAT), an Energy Materials Network Consortium funded by the U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy, Solar Energy Technologies Office.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.