Automation of COMSOL Multiphysics Parameter Studies using the MATLAB LiveLink

Dominik Kern[a], Nils-Henning Framke[b]

September 26, 2012

Abstract

COMSOL Multiphysics is a Finite Element Methods (FEM) software package for the numerical solution of partial differential equations. The MATLAB LiveLink combines the powerful FEM abilities of COMSOL with the versatile programming environment of MATLAB. This tutorial shows how to utilize the LiveLink for the frequent application of a parameter study. The FE analysis is invoked for several parameters by a M-File and its results are post-processed in MATLAB. As demonstration example serves the verification model *large deformation beam* from the model library. It is modified such that its geometry is parameterized. It can then be solved and post-processed automatically for several parameter combinations by a M-File.

This tutorial refers to versions: COMSOL 4.2a and MATLAB 2011b.

Contents

[a]kern@kit.edu

[b]nframke@online.de

Building the model in COMSOL GUI

The model can be opened from the model library structural mechanics – verification models – large deformation beam. It contains a cantilever beam that is clamped on the left end and highly deformed by forces on the right end. In order to perform a geometrically nonlinear

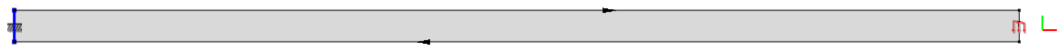
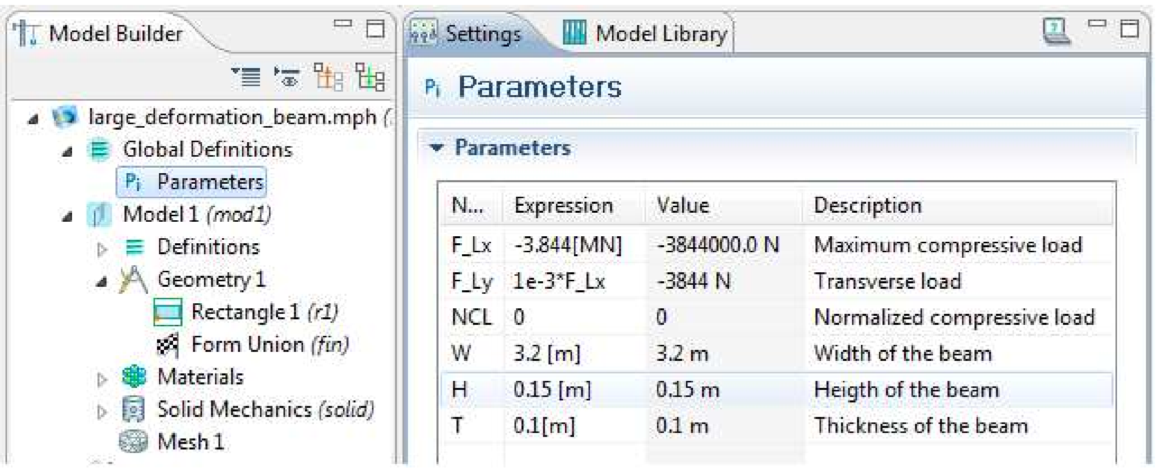


Figure 1: large deformation beam from model library – structural mechanics – verification models

deformation analysis for several lengths and the corresponding evaluations of the end point path we need a parametrized geometry.
We will now prepare the parameterized model using the COMSOL GUI.

1. Start COMSOL and open the model library. Select the large deformation beam from structural mechanics – verification models.

2. In the *Model Builder* tree goto *Parameters* in the *Global Definitions* node and enter the parameters *W, H, T* shown in Figure 2.



N...	Expression	Value	Description
F_Lx	-3.844[MN]	-3844000.0 N	Maximum compressive load
F_Ly	1e-3*F_Lx	-3844 N	Transverse load
NCL	0	0	Normalized compressive load
W	3.2 [m]	3.2 m	Width of the beam
H	0.15 [m]	0.15 m	Heigth of the beam
T	0.1[m]	0.1 m	Thickness of the beam

Figure 2: Load and geometry parameters of the beam

3. From the *Model* node select the *Geometry* node and locate the *Rectangle* entry. Overwrite the *Width* and *Height* values of the rectangle definition with the corresponding global parameters *W* and *H*, respectively.

4. From the *Model* node select the *Solid Mechanics* node and overwrite the thickness from the Thickness section with the corresponding global parameter *T*.

5. Rebuild and remesh the geometry and compute the study to check if geometry, mesh and study are built/computed correctly.

6. From the *Model* node select the *Results* node. Select *Cut Point 2D 1* from *Data Sets* and set the x-coordinate of the Cut Point to W and the y-coordinate to $T/2$.

7. From *File* select *Reset History*.

8. From *File* select *Save As Model M-File...* and save. The M-File with the COMSOL model will be `beamModel.m` for this tutorial.

By default COMSOL will save any changes you have made during the modeling process to the MATLAB M-File. Therefore the resulting M-File might be large and full of redundant information. Deleting the modeling history with *File – Reset History* will reduce the M-File-Code to what you actually see in the *Model Builder* tree at the time of saving the model.

Running the COMSOL model from MATLAB via LiveLink

Start the COMSOL – MATLAB LiveLink (Windows: *COMSOL with MATLAB*, Linux: `comsol server matlab`, alternatively using mphstart.m in the COMSOL installation directory) and check that it is set up correctly (test with autocomplete `mph<TAB>` if COMSOL commands, such as `mphplot, mpheval, ...` are available).
Next we will modify the saved Model M-File from COMSOL. After that we will create a new M-file which invokes the COMSOL model for different parameters and access the results for postprocessing.

Prepare the COMSOL model M-File for a parameter study

1. By default COMSOL is building a MATLAB function from your model if you save to a M-File in COMSOL. In the first line of the M-File you find:

   ```
   function out = model
   ```

   Replace `model` by the name of the M-File (beamModel in this tutorial)

   ```
   function out = beamModel
   ```

2. Add input variables for the width W, height H and thickness T of the beam to the MATLAB function

   ```
   function out = beamModel(W,H,T)
   ```

3. Use these input variables in the COMSOL model (`beamModel.m`).

   a) Locate the section in the COMSOL model M-File where the *global parameters* are set. It looks like:

```
model.param.set('F_Lx', '-3.844[MN]', 'Maximum␣
   compressive␣load');
model.param.set('F_Ly', '1e-3*F_Lx', 'Transverse␣load');
model.param.set('NCL', '0', 'Normalized␣compressive␣load'
   );
model.param.set('W', '3.2␣[m]', 'Width␣of␣the␣beam');
model.param.set('H', '0.15␣[m]', 'Heigth␣of␣the␣beam');
model.param.set('T', '0.1[m]', 'Thickness␣of␣the␣beam');
```

b) For the parameter study it is more convenient to use numerical input for the MATLAB model function. Use `num2str` to bulid the required strings for width W, height H and thickness T. So the modified section should then look like:

```
% 'F_Lx', 'F_Ly', 'NCL' remain unchanged in this tutorial
model.param.set('W', [num2str(W), '␣[m]'], 'Width␣of␣the␣
   beam');
model.param.set('H', [num2str(H), '␣[m]'], 'Heigth␣of␣the
   ␣beam');
model.param.set('T', [num2str(T), '␣[m]'], 'Thickness␣of␣
   the␣beam');
```

4. Call the MATLAB function `beamModel.m` building and solving the COMSOL model from the MATLAB Command Line with the desired parameters, i.e. type

```
W=3.2; H=0.15; T=0.1; model=beamModel(W,H,T)
```

As result there will be a COMSOL object “model” in the MATLAB Workspace.

You can now use the MATLAB LiveLink `mphnavigator(<model name>)` function to explore the model object. It allows you to get an insight into the models properties and methods and can give you hints how to control or access the model object.
MATLAB users might not be very familiar with the object oriented approach in MATLAB. This is not needed for getting started with the post processing of the COMSOL model in MATLAB. However, it is advisable to read the *MATLAB Classes and Object-Oriented Programming* section in the MATLAB Documentation [1] for more detailed information.

Accessing and Postprocessing the COMSOL results in MATLAB

For the following it is recommended to start building a MATLAB script instead of using the command line. We will now access the displacement data of the Point at $(W, T/2)$ to plot its trajectory. Keep in mind, how it was accomplished using the GUI: at first was defined which quantity is to be evaluated (displacement u and v of a point) and next where (point on the beam's free end). The same sequence of steps applies using a script in MATLAB.

1. Clear all variables from the MATLAB workspace, close all figures and execute the COMSOL model with your M-FILE script as previously on the command line.

2. Now we set up a point evaluation

   a) Add the command

   ```
   model.result.numerical.create('pev2', 'EvalPoint');
   ```

   to your script. The part `model.result.numerical` always refers to the numerical result data of the COMSOL model. The part `create('pev2', 'EvalPoint')` is creating the *Point Evaluation*. This part is equivalent to adding a Point Evaluation to your model in the COMSOL GUI by right-clicking the *Derived Values* node from the *Results* node in the *Model Builder* tree. The `create`-method is requiring two input strings.

      i. The first one ('pev2' in our case) is the *tag* of the Point Evaluation. It is the arbitrary name of the evaluation point in the MATLAB object hierarchy. The following is probably true in most cases: Whenever in the COMSOL GUI by right-clicking a node you are able to create more than one entity, i.e. a mesh, plot group, solver ..., it will have a tag within the MATLAB model object hierarchy. You can use the *tags()* method to see tags that are already present in sections of your model. Type `model.result.tags()` in the MATLAB command line. If you have set up the model according to this tutorial it will return something similar to

      ```
      ans =

                             java.lang.String[]:

                         'pg1'
                         'pg2'
      ```

      where 'pg1' and 'pg2' refer to the default plot groups created by COMSOL.

      ii. The second string ('EvalPoint' in our case) specifies the entity to create. As far as the authors of this tutorial know, the entity names that need to be passed to the `create` function are not documented. For now the best way to find these names is to make the corresponding modification within the COMSOL GUI, save to another M-File and compare to the actual model M-File, e.g. by using the MATLAB editor tool *Compare Against*.

b) Now specify which point of the model corresponds to the *Point Evaluation*. We already set up a *Cut Point 2D* for that while modifying the model within the COMSOL GUI (as described in the large deformation beam tutorial). Execute `model.result.dataset.tags()` in the MATLAB command line. It will return something like

```
ans =

                    java.lang.String[]:

                'dset1'
                'cpt1'
```

In case you would like to create a new Cut Point in the COMSOL GUI you would have right-clicked on the *Data Sets* node from the *Results* node in the *Model Builder* tree.
Execute `model.result.dataset('cpt1').getType()` in the MATLAB command line. The return will tell you, that 'cpt1' refers to a Cut Point. Note that we have just used the *tag* 'cpt1' to navigate to a certain entity within the model object.

c) Now we set 'cpt1' to be the geometrical position for the Point Evaluation. Add this command to the script:

```
model.result.numerical('pev2').set('data', 'cpt1');
```

The `set`-method allows to modify `properties` of the entity which can usually be seen with the `mphnavigator` or by executing `model.result.numerical('pev2').properties()` in the MATLAB command line. Actually a lot of the properties can be identified from the settings tab of an entity in the COMSOL GUI.

3. Add

```
model.result.numerical('pev2').set('expr', 'u');
```

to the script to set the expression of interest to be the displacement u (x-direction).

4. Now use the `getReal()`-method to access the numerical data of the evaluation point. Add

```
u=model.result.numerical('pev2').getReal();
```

to the script.

5. Repeat the above for the displacement v (y-direction).

6. You are now able to plot the trajectory of the point $(W, T/2)$.

In order to do a parameter study simply introduce a loop to the script and loop over the geometry parameter of interest. For the postprocessing one may use COMSOL commands, such as `mphplot(model,'pg1')`, see the LiveLink Documentation [2] for more commands. Alternatively, one may examine the COMSOL object and access it directly by its methods, such as `.getReal()`.

Example source code

Now for a parameter study modify the model M-File as follows: function name, parameter in function head, and parameter in definition (here width W) it should then look like

```
function out = beamModel(W)
...
model.param.set('F_Lx', '-3.844[MN]', 'Maximum␣compressive␣load')
   ;
model.param.set('F_Ly', '1e-3*F_Lx', 'Transverse␣load');
model.param.set('NCL', '0', 'Normalized␣compressive␣load');
model.param.set('W', [num2str(W),'[N]'], 'beam␣width');
...
```

Now create a new script that invokes the FEM-calculation passing parameters (here width W)

```
clc; clear;
W=3.2; % this could be set in a loop
model=beamModel(W); % run beam model
% post-process using COMSOL commands (cf. LiveLink Docu)
mphplot(model,'pg1');
figure;
mphplot(model,'pg2');
ui=mphinterp(model,'u','coord',[W;0.005]); % 0.005 is half height
vi=mphinterp(model,'v','coord',[W;0.005]);
% alternatively mpheval for evaluation at the nodes
% post-process accessing the COMSOL object
model.result.numerical('pev1').set('expr','u');
ue=model.result.numerical('pev1').getReal(); % end point
   displacement (x) for each load step
model.result.numerical('pev1').set('expr','v');
ve=model.result.numerical('pev1').getReal(); % end point
   displacement (y) for each load step
figure; plot(ue, ve, ui, vi, 'Linewidth',2); % plot path of end
   point trajectory
```

There are both variants included: the direct access to the COMSOL object and the access via a COMSOL command, here `mphinterp` [2]. Figure 3 shows the resulting plot.

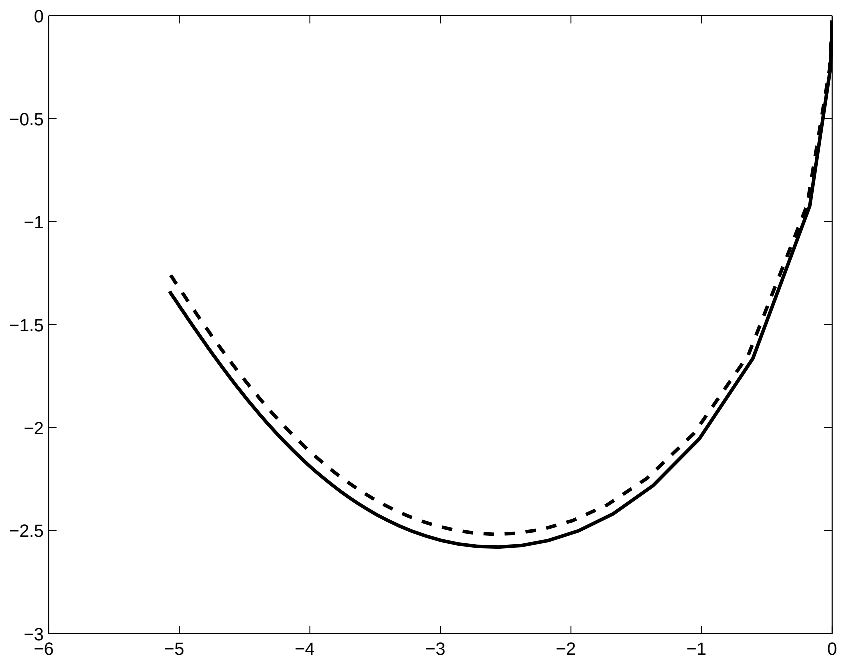


Figure 3: End point paths for the parameter values from the model library tutorial accessing the COMSOL object (solid line) and using the `mphinterp` command (dashed)

Some remarks about export to SimuLink

In COMSOL versions 3.X there was the option to save COMSOL models as SimuLink blocks. The only thing left in versions 4.X is export of constant state space matrices by right-clicking the *Solver* node choosing *Other - State Space*. There its input and output must be defined. Global parameters, such as NCL in the large deformation beam model may be used as input and operators, e.g. an integral operator `mod1.intop1(mod1.u)`, as output. However, to find the correct names is not intuitive and moreover only linear systems can be described by this formulation. If you really want to use this feature, see the documentation of `mphstate` in the LiveLink Documentation [2]. Note that MIMO systems are created by passing lists as parameters, e.g.:

```
M = mphstate(...'output', {'mod1.ppb1','mod1.ppb2'},...);
```

References

[1] The MathWorks, Inc., *MATLAB® Object-Oriented Programming R2011b*, September 2011

[2] COMSOL, Inc., *LiveLink™ for MATLAB User's Guide LiveLink™ for MATLAB® User's Guide*, October 2011