```
                 _____________      _________
                /_____  _____/     /__  ____ \
                     / /  ______     / /   / /         _____
                    / /  /_____ \   / /   / /         /____ \
                   / /    ____/ /  / /   / /          ____/ /
                  / /    /___  /  / /   / /          / ____/
      _______    / /  ______/ /  / /___/ /          / /____
     / _____ \  /_/   \______/  /_______/          /______/
    / /    / / _______    _________  _________  __________
   / /    / / / _____ \  / _______/ / _______/ / __  __  /
  / /    / / / /    / / / /____    / /____    / / / / / /
 / /    / / / /    / / / _____/   / _____/   / / /_/ / /
/ /____/ / / /____/ / / /        / /______  / /     / /
\_______/  \_______/ /_/        /________/ /_/     /_/
```

Converter from T3D to OOFEM

Version 2.4

User Guide

January 24, 2005 October 29, 2014

Daniel Rypl

Czech Technical University in Prague
Faculty of Civil Engineering, Department of Mechanics
Thákurova 7, 166 29, Prague
Czech Republic

e-mail: drypl@fsv.cvut.cz
http://mech.fsv.cvut.cz/~dr/dr.html

1 Introduction

T3d2oofem converts the output of T3d mesh generator to the input file for Oofem finite element solver. T3d2oofem processes on the input two files

- output file of T3d containing all relevant data (appropriate T3d command line options for output specification are required),
- control file containing Oofem control records and assignment of some of these control records (boundary conditions, loading, material) to individual entities of the T3d model.

T3d2oofem processes the control file at first and stores internally the Oofem control records and property assignments. After that, the T3d output file is processed (without actually storing the individual mesh entities) and the Oofem input file is produced. Optionally, Oofem domain input file (Oofem input file without certain control records) for adaptive analysis is also generated.

2 Synopsis

T3d2oofem is executed as

t3d2oofem *control_file t3d_output_file oofem_in_file* [options]

The following command line options are recognized

-din domain input file (named *oofem_in_file*.**domain**) is also generated

Usage and list of control file keywords can be obtained by executing

t3d2oofem -h

3 Format of Control File

Control file consists of two sections. The first section contains Oofem control records, this is all Oofem input file statements without the mesh (node and element statements). Note that the statement describing the number of records in the Oofem input file has to be used without textual keywords and without the first two entries corresponding to the number of nodes and elements.

An example of the first section of the control file may look like this

```
test.out
testing example
linearstatic nsteps 1
domain 3d
outputmanager tstep_all dofman_all element_all
# in the following statement, keywords and entries corresponding
# to the number of nodes and elements are omitted
1 1 4 0 1
simplecs 1 thick 0.25
```

```
isole 1 d 4850.0 e 210.0e6 n 0.3 talpha 1.2e-6
boundarycondition 1 loadtimefunction 1 prescribedvalue 0
constantsurfaceload 2 loadtimefunction 1 components 3 0 0 -10 loadtype 3 ndofs 3
nodalload 3 loadtimefunction 1 components 3 -2.5 0 0
deadweight 4 loadtimefunction 1 components 3 0 0 -1
constantfunction 1 f(t) 1.0
```

The second section contains records describing assignment of node and element properties. Each assignment consists of model entity or model property specification followed by one or more node property specification(s), element property specification(s), element type specification(s) and boundary load and/or code specification(s). In the following paragraphs, syntax of assignment specifications is given. The keywords are typed in bold, user specification in italic, braces enclose obligatory parameter(s), brackets enclose optional parameter(s), parentheses enclosed (at least once) repeated statement or parameter, | stands for logical XOR, and # represents an identification number.

Model entity specification has the form

({ **vertex** | **curve** | **surface** | **patch** | **shell** | **region** } { **all** [**except** (#)] | (#) })

and it is valid for all the subsequent node and element property assignments until a new model entity or property specification is encountered or until the keyword **cancel** is encountered. Note that curve/surface/patch/shell/region specification includes all elements on that entity but only interior nodes of that entity.

An example of the model entity specification may look like this

```
vertex all except 1 2
curve all
surface 1 2 11 12
patch 24
shell 13 14
region 1
```

Model property specification has the form

({ **property** } { **all** [**except** (#)] | (#) })

Similarly as before, model property specification is valid for all the subsequent node and element property assignments until a new model entity or property specification is encountered or until the keyword **cancel** is encountered. Note that model property specification include all nodes and elements with that property.

An example of the model entity specification may look like this

```
property 2
```

Node property assignment has the form

({ **nodeprop** } { *oofem_string* })

where *oofem_string* is proper Oofem boundary condition or load specification referring to the relevant control record(s) in the first section.
An example of the node property assignment may look like this

```
nodeprop bc 3 1 1 0 load 1 3
```

or like this

```
nodeprop bc 3 1 1 0
nodeprop load 1 3
```

Element property assignment has the form

({ **elemprop** } { *oofem_string* })

where *oofem_string* is proper Oofem material, cross-section or body load specification referring to the relevant control record(s) in the first section.

An example of the element property assignment may look like this

```
elemprop crosssect 1 mat 1 bodyloads 1 4
```

or like this

```
elemprop crosssect 1 mat 1
elemprop bodyloads 1 4
```

Element type assignment has the form

({ **edgetype** | **triatype** | **quadtype** | **tetratype** | **hexatype** } { *oofem_element_type* })

where *oofem_element_type* is proper Oofem element type according consistent with the spatial dimension, element shape, element degree, and element functionality. Note that on model entities discretized by mixed meshed (for example combination of triangular and quadrilateral elements), element type for both types of elements must be provided. Note that T3d2oofem converts the pyramidal as well as wedge elements to degenerated hexahedral elements.

An example of the element type assignment may look like this

```
tetratype LTrSpace
hexatype LSpace
```

Boundary load assignment has the form

({**boundaryload** | **bload** } {(*oofem_load_number*)})

where *oofem_load_number* is number of the boundary condition record corresponding to the boundary loading. Note that keywords **boundaryload** and **bload** are interchangeable. Note that boundary load assignment cannot be generally applied using the element property assignment because usually only some of the elements of a particular model entity are subjected to the boundary load and because the appropriate face of these elements must be detected and specified in the final specification generated in the Oofem input file.

An example of the boundary load assignment may look like this

```
boundaryload 2 5
```

Boundary code assignment has the form

({**boundarycode** | **bcode** } {(*oofem_code_number*)})

where *oofem_code_number* is number of the appropriate boundary condition. Note that keywords **boundarycode** and **bcode** are interchangeable. Similarly as for the boundary load assignment, also the boundary code assignment cannot be generally applied using the element property assignment.

An example of the boundary code assignment may look like this

```
bcode 3
```

A complete example of the second section of the control file may look like this

```
vertex all except 1 2 3 4 5 6
curve 1 2
nodeprop bc 3 1 0 0

vertex 1 2
curve 3
nodeprop bc 3 1 1 1

vertex 5 6
curve 4
nodeprop load 3

region all
elemprop crossSect 1 mat 1 bodyLoads 1 4
tetratype LTrSpace
hexatype LSpace

surface 2 4
boundaryload 2
```

Note that empty lines in the second section of the control file are ignored. Lines starting by # (hash) are treated as comments in both sections. However, while comments in the first section are copied to the generated input file, comments in the second section are discarded.

4 General Remarks

For correct run of T3d2oofem it is necessary to run T3d with **-p 8** or **-p 512** command line option. This will ensure that relevant boundary entity information are included in T3d output file. Option **-p 512** is required in the case when application of edge load to 3D elements is to be processed (otherwise the request for the application of the edge load to 3D elements is ignored).

In T3d output file, the mesh entities are classified to the entity of the lowest dimension. This implies that boundary nodes of a particular model entity are not classified to that model entity but to other model entity of possible lowest dimension. For example, boundary nodes of surface are classified to boundary curves of that surface or boundary vertices of those curves. Also note that mesh entities are classified to the top parent physical entity. This implies that if there is a physical curve c1 fixed to part of another physical curve c2 (not fixed to any physical curve), then the edges on curve c1 (if subjected to the output) will be classified to c2. As a consequence, curve c1 will appear as nonexisting and cannot be therefore subjected to boundary conditions etc. Should you need to apply these boundary conditions you must enforce curve c1 to appear in T3d output by changing the model (for example by changing curve c2 to virtual and fixing to it not only curve c1 but also curve c3 on its remaining part).

When handling surfaceload, it is always applied to all elements incident to particular face. For example, if there are two spatial regions sharing a common surface subjected to surfaceload, then surfaceload will be applied to both elements sharing the face on the common surface. Thus to enforce proper response, the magnitude of the loading should be scaled appropriately (typically by half). Since there is generally not possible that the number of elements on one side of the surface is different from that on the other side of the surface, the scaling should be always possible. This behaviour is the consequence of the current implementation generating the Oofem input file on the fly without storing any mesh data.

When handling edgeload, it is always applied to just a single element incident to particular edge. For example, if there are two planar domains sharing a common curve subjected to edgeload, then edgeload will be always applied to just one of the two elements sharing the edge on the common curve. This is important when handling the above described case of curve c1 fixed to part of curve c2, subjected to edge load. This behaviour is also crucial when handling edge load subjected to curve bounding 3D elements as there may be several elements sharing the same edge on that curve.

5 Compilation

For compilation on Linux/Unix platforms, use typically command

```
gcc -O2 -o t3d2oofem -lm t3d2oofem.c
```

Should you prefer other compiler, replace gcc by the name of your preferred compiler and follow its syntax for proper specification of command line options.

For compilation on Windows/Mac platforms, created appropriated project within your favourite compiler GUI environment.

6 Bugs, Problems and Limitations

T3d2oofem does not handle all the control lines recognized by Oofem in ordinary Oofem input file (for example line describing Vtk output). If you need to have such a control line in the input file, add it to the input file produced by T3d2oofem manually.

Note that T3d2oofem does not support all (not only the most recent) features of Oofem. For example, there is no support for variable local coordinate system and variable loading within a single model entity. Should you require these features you should apply some additional postprocessing of Oofem input file generated by T3d2oofem.

Similarly, T3d2oofem may not handle all features produced by T3d, for example, interface elements or bubble quadratic elements are not recognized.

Due to the limitations of the current Oofem element library, pyramid and wedge elements produced by T3d are handled as degenerated hexahedral elements.

T3d2oofem may produce warnings that could be but need not necessarily be an error. For example, warning

Warning: Surface 14 referenced in ctrl file is not present in t3d output file

may be caused by the fact that in T3d output there is present no element classified to surface 14 (because, surface 14 is, for example, bounding a region and explicit request for output of elements on it was not required in T3d) and there is simultaneously present no node classified to surface 14 (because, for example, there is no inside node on that surface or because the inside nodes are classified to other entities, e.g. fixed vertices). Nevertheless, the issued warnings are worth to be checked to prevent eventual problems with the Oofem analysis.

Please, report the bugs to the author together with the description of circumstances (input file, command line options, platform). Thank you.