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

Converter from T3D to OOFEM

Version 2.7

User Guide

December 13, 2018

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 may not contain entries corresponding to the number of nodes and elements. The entries in the statement can be ordered arbitrarily only if corresponding Oofem textual keywords are used. In such a case the statement can contain also specification of number of optional records. If the textual keywords are not used (for compatibility with older control files), the statement must contain only the obligatory components in predefined order (number of cross sections, number of materials, number of boundary conditions, number of initial conditions and number of load time functions).

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
ncrosssect 1 nmat 1 nbc 4 nic 0 nltf 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 features. Each assignment consists of model entity or model property specification or 1D interface specification followed by one or more of the following feature specifications

- node type specification (single),
- node property specification (multiple),
- element type specification (multiple),
- element property specification (multiple),
- boundary load specification (multiple),
- boundary code specification (single),
- local coordinate system specification (single).

The second section may also contain specification of protective spheres (around singularities), solution from which is excluded from the error assesment and mesh refinement during an adaptive analysis.
In the following paragraphs, syntax of model entity, property, 1D interface, feature and protective sphere specifications are given. The keywords are typed in **bold**, user specification in *italic*, braces {} enclose exclusive selection, brackets [ ] enclose optional parameter(s), parentheses () enclose (at least once) repeated parameter, chevrons ⟨⟩ enclose (at least once) repeated statement (each one on a new line), | stands for exclusive OR (logical XOR) and # represents an identification number.

Model entity specification has the form

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

and it is valid for all the subsequent node and element feature assignments until a new model entity or property specification or 1D interface 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 those nodes on that entity which cannot be classified to other model entity of lower dimension. Interface specification includes only the elements.

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
interface all
```

Model property specification has the form

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

Similarly as before, model property specification is valid for all the subsequent node and element feature assignments until a new model entity or property specification or 1D interface 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
```

Specification of 1D interface between pair of vertices has the following form

⟨ **interface1d vertex** # # **dir** *normal_dir* ⟩

where *normal_dir* defines the normal direction of the interface. Note that 1D interface between pairs of vertices can be also defined using 1D interface element produced by T3d (if your version of T3d supports that feature) to which the normal direction is supplied via element property specification (see the last example of element property assignment below).

Specification of 1D interface between pair of compatible curves has the following form

⟨ **interface1d curve** # # { **dir** *normal_dir* | **tan** | **nor** *normal_vector* } ⟩

The interface normal direction can be defined either directly by keyword **dir** using *normal_dir* of the interface, or indirectly by keyword **tan** (normal direction of interface is tangent to the curve) or keyword **nor** (normal direction of interface is normal to the curve and *normal_vector*). Note that curves are compatible if they are topologicaly identical and discretized by topologically identical mesh.

Specification of 1D interface between pair of compatible surfaces/patches/shells has the following form

⟨ **interface1d** { **surface** | **patch** | **shell** } # # { **dir** *normal_dir* | **tan** *plane_vector* | **nor** } ⟩

The interface normal direction can be defined either directly by keyword **dir** using *normal_dir* of the interface, or indirectly by keyword **tan** (normal direction of interface is tangent to the surface/patch/shell in plane defined by given *plane_vector* and surface/patch/shell normal) or keyword **nor** (normal direction of interface is normal to the surface/patch/shell). Note that surfaces/patches/shells are compatible if they are topologicaly identical and discretized by topologically identical mesh.

Example of 1D interface specifications may look like this

```
interface1d vertex 2 14 dir 1 0 0
interface1d curve 12 7 tan
```

```
interface1d curve 11 6 nor 0 0 1
interface1d patch 7 9 nor
interface1d surface 5 13 tan 0 1 0
```

Node type assignment has the form

nodetype *oofem_node_type*

An example of the node type assignment may look like this

```
nodetype rigidarmnode
```

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 type assignment has the form

⟨ { **edgetype** | **triatype** | **quadtype** | **tetratype** | **hexatype** } *oofem_element_type* ⟩

where *oofem_element_type* is proper Oofem element type 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
```

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
```

or like this

```
elemprop normal 1.2 0.4 0.8
```

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 type of 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
```

Assignment of local coordinate system at curve or surface/patch/shell node with respect to primary entity (curve or surface/patch/shell to which the node is classified) has the form

lcsprimary { **uvw** | **vwu** | **wuv** | **vuw** | **wvu** | **uwv** } *ref_vector*

Triads **uvw**, ..., **uwv** control which vectors are used to define local right-hand coordinate system xyz. Local x and y axes are always given by the first and second vector specified by the triad, in which u indicates tangent (at curve node) or normal (at surface/patch/shell node) read from T3d output file, v stands for orthonormalized reference vector (with respect to u and w), and w denotes a vector computed as normalized vector product of u and given *ref_vector* defining the plane uv of right-hand coordinate system uvw. Note that while for the first three triads the local z axis is oriented identically with corresponding from u, v, and w vectors, in the case of last three triads its orientation is opposite.

An example of the local nodal coordinate system specification with respect to primary entity may look like this

```
lcsprimary wvu 0 0 1
```

in which case local x axis (represented by w in the triad) is defined as normalized vector product of tangent (normal) vector to a curve (surface/patch/shell) (represented by u) and reference vector $\{0\ 0\ 1\}$, local y axis (represented by v) is defined as the reference vector orthonormalized with respect to local z (represented by u) and x (represented by w) axes (this means as a vector forming with refrence vector angle smaller than 90 degrees and being simultaneously perpendicular to tangent/normal vector and to vector product of tangent/normal vector and reference vector), and local z axis (represented by u) is defined as vector product of unit vectors on local x (represented by w) and y (represented by v) axes and has opposite orientation (in this particular case) than the tangent/normal vector.

Specification of a protective sphere around singularity has one of the forms

⟨ **singularnode** # *radius* ⟩

in which the protective sphere of *radius* is centered at given node, or

⟨ **singularpoint** *center radius* ⟩

in which the coordinates of sphere *center* and sphere *radius* are given. When using the protective sphere(s) around singular node(s) and/or point(s), the elements which are fully inside the sphere are assigned to a new region (with the same properties as the original one) which is excluded from error assessment and refinement by modifying properly the regionskipmap parameter in analysis record. The radius of the sphere should be large enough to eliminate the effect of enclosed singularity but small enough to avoid significant influence on the assessed error of the solution. A rule of thumb is to use such a radius that the density of mesh inside the sphere does not get coarser during the adaptive remeshing by T3d than the refined density specified (for the remeshing) along the the perimeter of the sphere. Note that use of protective sphere(s) is meaningless for other than adaptive analysis.

Example of protective sphere specification may lookk like this

```
singularnode 17 0.8
singularpoint 14.3 18 0.0 0.65
```

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 order to process properly the local coordinate system requirements, T3d must be run with **-p 14** or **-p 518** command line option which will include except boundary entity information also components of local tangent vectors (on inner curve nodes) and normal vectors (on inner surface/patch/shell nodes).

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 (except interface elements) incident to a 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 (not interface element) incident to a 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 the number of elements sharing the edge on the curve is generally varying along the curve. Sometimes it may be relevant that the edgeload is applied to elements classified to particular entity (e.g. for example if a curve subjected to the edgeload is shared by several surfaces or regions). This can be achieved by defining the relevant entity in the T3d input file before the other candidate entities. Note that order of the definition of the entities is relevant, not their ids. Note that this approach fails if edgeload should be applied to 3D elements rather than 2D elements sharing the curve because 2D entities are always processed before 3D entities due to the native ordering of elements in T3d output.

T3d2oofem may produce warnings concerning non-existing entities. 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). Sometimes this warning is irrelevant (for example if nodes on the missing entity are subjected to constraints), but sometimes it indicates a problem (for example if the missing entity is subjected to a loading). There are 3 ways around this problem:

- use quadratic elements - note however that this does not help in case of surface/patch/shell entity composed of just a single element, if the entity is bounding a region or interface and the entity is not explicitly designated for output).
- designate problematic entity for output and modify constitutive parameters (e.g. stiffness) for elements classified to that entity, so that they will not influence the response.
- refine locally the mesh so that there exist always at least one node classified to the entity.

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, create appropriated project within your favourite compiler GUI environment.

6 Bugs, Problems and Limitations

Note that Oofem analysis record in the first section may be followed by several other control lines expanding the analysis record. This version of T3d2oofem can handle only additional control lines describing metasteps, export modules, initialization modules and xfem managers. If you are using other additional control lines, comment them by # (hash) in control file and then uncomment them in the Oofem input file produced by T3d2oofem.

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 with respect to other than primary entity. There is also missing support for 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, bubble quadratic elements (if available in your version of T3d) 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.

Boundary load and boundary code are not applied to interface elements.

If using 1D interface specification(s), the total number of elements written to Oofem input file is invalid as it does not include the generated 1D interface elements. This is due to the single-pass processing of T3d output file which does not allow to precompute number of 1D interface elements to be generated. The user is informed about this fact by a warning.

T3d2oofem may produce warnings that could be but need not necessarily be an error. 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.