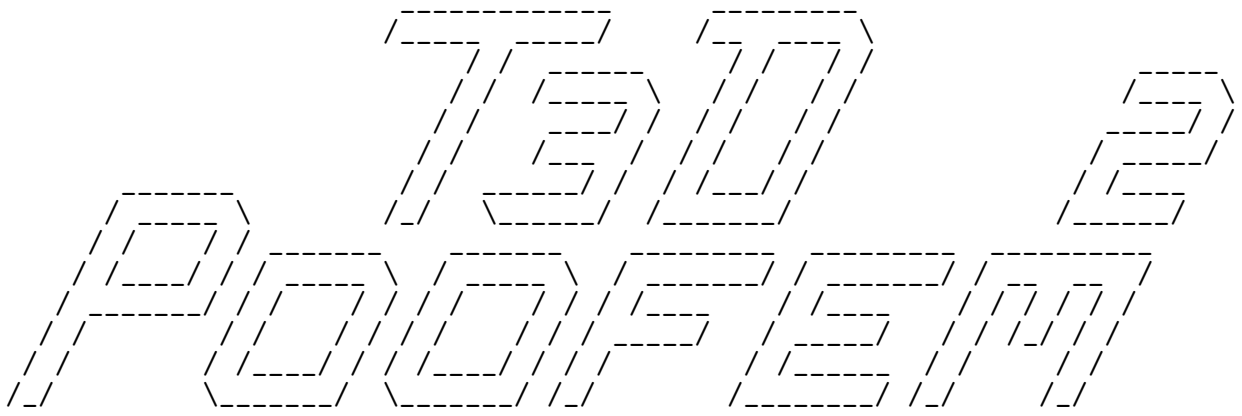

Converter from T3D to POOFEM

Version 2.0

User Guide

October 30, 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

T3d2poofem converts the output of T3d mesh generator to the input files for Poofem parallel finite element solver. T3d2poofem 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 Poofem control records, assignment of some of these control records (boundary conditions, loading, material) to individual entities of the T3d model and data controlling the decomposition to individual subdomains.

T3d2poofem processes the T3d output file at first and stores internally the mesh. Then the control file is processed and the control records, property assignments and decomposition control records are stored. After that the mesh is converted to weighted graph which is subjected to decomposition using Metis graph partitioner. The obtained subdivision is then projected back to the mesh and Poofem input files for individual partitions of the resulting subdivision are produced.

2 Synopsis

T3d2poofem is generally executed as

t3d2poofem *control_file t3d_output_file poofem_in_file* [options]

The following command line options are recognized

-np #	number of partitions (default 1)
-nc	node cut partitioning (default)
-ec	element cut partitioning
-gn	global numbering (default)
-ln	local numbering
-rn	node renumbering (for -nc and -ln only)
-last	number interface nodes last (for -nc only)
-sdd	static domain decomposition (for -nc only)
-din	generate also domain input file
-dsz	reorder partitions according to average element size (for -nc, not -sdd and pure tetra mesh only)
-X	graphic interface (for -np # > 1 only)

Note that option **-X** is available only if T3d2poofem is compiled with support for Elixir graphic interface (available for Unix/Linux only).

Poofem input files produced by T3d2poofem are named *poofem_in_file*.*part_number*. When using option **-din**, also domain input files (Poofem input files without certain control records) for parallel adaptive analysis are produced. These files are named as *poofem_in_file*.**domain.** *part_number*. Note that partitions are numbered from zero. When running T3d2poofem with **-np 1** then ordinary Oofem input files are generated and their names do not contain the suffix with the partition number.

Note that T3d2poofem can be compiled without support for Metis partitioning (sequential regime). In such a case, only options **-rn** and **-din** are available. The naming conventions of T3d2poofem output files are then the same as explained above for case with **-np 1**.

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

t3d2poofem -h

2.1 Node Cut and Element Cut Partitioning

When using node cut partitioning, the elements of the mesh are uniquely assigned to individual partitions. Nodes which are incident only to elements assigned to the same partition are local nodes. Nodes which are incident to elements assigned to different partitions are shared nodes. Isolated nodes are treated as shared by all partitions and are always numbered from 1.

When using element cut partitioning, the nodes of the mesh are uniquely assigned to individual partitions. Elements which are formed only by nodes assigned to the same partition are local elements. Elements which are formed by nodes assigned to different partitions are shared elements. Nodes incident to shared elements but not assigned to local partition are called Null nodes. Isolated nodes are assigned to partitions by the partitioner.

In output for a particular partition, there may be present also elements assigned to other partition. These are so called remote copy elements mirroring a neighbourhood of that particular partition (for analysis with nonlocal averaging). Nodes incident to the remote copy elements and not being shared nodes are also present as Null nodes.

2.2 Global and Local Numbering

When global numbering is applied, all the nodes and elements have global numbers inherited from the mesh numbering in T3d output file.

When local numbering is applied, nodes and elements on individual partitions are assigned local numbers starting from 1. Nevertheless global numbers given by the mesh numbering in T3d output file are used as special parameters of records of individual nodes and elements.

2.3 Node Renumbering

When node renumbering is required, the nodes are renumbered locally for each partition. This may be useful for the solution of the algebraic system of linear equations in Poofem. Note that when internal renumbering within Poofem is required, the renumbering by T3d2poofem is meaningless.

The node renumbering is based on Sloan's renumbering algorithm which attempts to minimize the nominal profile of connectivity matrix. Note that for regular structured meshes the renumbering can be very slow.

2.4 Numbering Interface Nodes Last

Some of the parallel solution solvers (for example Schur's complement method) require that nodes on interface between individual partitions (shared nodes) are numbered as last. The request for numbering interface nodes as last is respected even if node renumbering is required.

2.5 Static Domain Decomposition

When static domain decomposition is required, the partitioning is performed according to the specification in the control file on the model entity basis. Note that partition assignment must be done for all model entities elements of which appear in the mesh.

2.6 Reordering Partitions According to Average Element Size

The produced partitions are reordered with respect to increasing average element size computed as partition volume divided by the number of elements assigned to the partition.

3 Format of Control File

Control file consists of two sections. The first section contains Poofem control records, this is all Poofem 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 Poofem 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 0.5
isole 1 d 2650.0 e 15.0e6 n 0.25 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 property specification (multiple),
- element type specification (multiple),
- element property specification (multiple),
- boundary load specification (multiple),
- boundary code specification (single),
- element computational weight specification (multiple),
- partition assignment specification (single),
- width of averaging zone specification (single),
- master node specification (multiple).

In the following paragraphs, syntax of model, property and feature 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 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 those nodes on that entity which cannot be classified to other model entity of lower dimension.

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** *poofem_string* ⟩

where *poofem_string* is proper Poofem 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** } *poofem_element_type* ⟩

where *poofem_element_type* is proper Poofem 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 all types of elements must be provided. Note that T3d2poofem 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** *poofem_string* ⟩

where *poofem_string* is proper Poofem 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
```

Boundary load assignment has the form

⟨ { **boundaryload** | **bload** } (*poofem_load_number*) ⟩

where *poofem_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 Poofem input file.

An example of the boundary load assignment may look like this

```
boundaryload 2 5
```

Boundary code assignment has the form

⟨ { **boundarycode** | **bcode** } *poofem_code_number* ⟩

where *poofem_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
```

Element computational weight assignment has the form

⟨ { **edgeweight** | **triaweight** | **quadweight** |
tetraweight | **pyramweight** | **wedgeweight** | **hexaweight** } *rel_weight* ⟩

where *rel_weight* is the relative computational weight of elements of given type within the active model entity and/or model property specification. Note that on model entities discretized by mixed meshed (for example combination of triangular and quadrilateral elements), element type for all types of elements must be provided. Note that within the context of mesh partitioning, pyramidal and wedge elements have to be distinguished from hexahedral elements although pyramidal as well as wedge elements are treated as degenerated hexahedral elements by the Poofem.

An example of the element computational weight assignment may look like this

```
triaweight 1.5
quadweight 2.0
```

Partition assignment has the form

partition *part_number*

where *part_number* is the number of partition to which elements corresponding to active

model entity and/or model property specification should be assigned. Note that the partition number should range from 1 to the number of required partitions.

An example of the partition assignment may look like this

```
partition 3
```

Specification of the width of averaging zone has the form

width *band_width*

where *band_width* is the width of the band of mirrored elements. This generally corresponds to the radius of the averaging zone.

An example of the width assignment may look like this

```
width 4.5
```

Master node specification has the form

⟨ **master** (*master_node_id*) ⟩

where *master_node_id* is the number of master node of any slave node. Master nodes are added as shared to all partitions on which their slaves appear if directive MASTER_SLAVE is defined (in the source code), or to all partitions if it is undefined. Adding master nodes as shared to other partitions should prevent parallel solution failure if some sort of diagonal precondition (for example Jacobi preconditioning) is adopted. However, using master nodes will not prevent the failure if only some of DOFs of the master node are used on particular partition while the others are not used.

An example of the master node specification may look like this

```
master 1 11 24
```

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
tetraweight 1.0
pyramweight 8.0
wedgeweight 8.0
hexaweight 8.0

surface 2 4
boundaryload 2

property 1
width 4.5

master 1 4
```

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 T3d2poofem 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 Poofem 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 Unix/Linux platforms, use the supplied Makefile, modify it according to your needs and preferences and run command

make -k all

which should result in executable file t3d2poofem.

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

Make sure that the prerequisite Metis library are installed on your computer prior you start to compile T3d2poofem. Metis library is not part of T3d2poofem distribution and the user have to download it from its home page.

Similarly, should you be interested in the graphic interface, make sure that prerequisite Ckit and Elixir libraries are installed on your computer before you start to compile T3d2poofem. Neither Ckit nor Elixir are part of T3d2poofem distribution. The locally maintained version of these libraries can be downloaded from http://mech.fsv.cvut.cz/web/?page=software. Note that Elixir library is available only for Unix/Linux platform.

6 Graphic Interface

When running T3d2poofem with graphic interface, the bottom menu palette provides you with Custom button with some facilities to explore the actual domain partitioning. There are few useful key and mouse bindings

General bindings

Ctrl a	fit all (all drawing windows will be affected)
Ctrl p	proceed
Ctrl s	stop
Ctrl x	exit

Bindings for fast navigation

B1	zoom window
Ctrl B1	pan view
Ctrl B2	zoom view
Shift B2	fit all (only active drawing window will be affected)
Ctrl Shift B1	rotate
B3	done

Bindings for selection

B1	select
Ctrl B1	select window
Shift B1	select nearest point (confirm by B1 or select next one by Shift B1)
B2	accept
B3	reject

Bindings to suspend and resume an action

Ctrl B3	suspend action
B3	resume action

In the above, B1, B2, and B3 stand for left, middle, and right mouse buttons.

7 Bugs, Problems and Limitations

In the current implementation, T3d2poofem can handle only meshes composed of linear elements.

Note that Poofem analysis record in the first section may be followed by several other control lines expanding the analysis record. This version of T3d2poofem 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 Poofem input file produced by T3d2poofem.

Note that T3d2poofem does not support all (not only the most recent) features of Poofem. 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 Poofem input file generated by T3d2poofem.

Similarly, T3d2poofem 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 Poofem element library, pyramids and wedge elements produced by T3d are handled as degenerated hexahedral elements.

T3d2poofem may produce warnings that could be but need not necessarily be an error. For example, warning

Warning: Surface 14 does not exist ==> created

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 Poofem analysis.

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