OOFEM User Manual

Release 2.5

Apr 08, 2021

CONTENTS:

CHAPTER

ONE

INTRODUCTION

OOFEM is a general purpose, object-oriented Finite Element code for solving mechanical. transport and fluid mechanics problems, primarily focused on academic/research use. It has been continuously developed since 1997. At present it consist of more than 230 thousands lines of source C++ code. It has acive development team and large user base around the world. The code has been successfully applied to solution of several industrial problems.

OOFEM is trying to be one of the best FEM solvers. To make this a reality, we focus on this and do not try to provide extensive pre and post processing capabilities. Here we rely on third party external tools.

OOFEM as a FEM solver solves the problems described by a set of partial differential equations. On input it expects the discretized domain and corresponding problem parameters, initial and boundary conditions. On output, it provides the primary unknown fields as well as other secondary fields of interest. The results can be exported in many ways to facilitate post-processing.

OOFEM is free software, distributed under GNU LGPL license.

1.1 General Features

- Structural, heat & mass transfer, and CFD modules
- Support for parallel processing on shared and massively parallel machines, computer clusters, Dynamic load balancing
- Direct & iterative solvers (IML, Spooles, PETSc, SuperLU, Pardisso)
- Full restart capability, support for adaptive and staggered analyses
- Postprocessing: X-Windows post-processing, export to VTK
- Python bindings allowing to script oofem and implement new components in Python

1.2 Documentation

The extensive documentation is available on OOFEM web pages:

- Input manual, available in [html] and [PDF].
- Element Library Manual, available in [html] and [PDF]
- Material Model Library Manual, available in [html] and [PDF]
- Programmer's manual, available in [html] and [PDF].
- The [Reference manual]
- Python bindings documentation , available in [html] and [PDF]

1.3 OOFEM Ecosystem

- OOFEM forum is the best place to ask questions and get support from developpers and user community.
- OOFEM wiki contains many useful resources, gallery of results and tutorials.
- OOFEM gitHub repository.

CHAPTER

TWO

INSTALLATION

2.1 Installation options

- **Official stable releases (http://www.oofem.org/en/download)**
  - Usually 12M release cycle
  - Binary packages for Windows (AMD64 version only) and Linux (x86_64 version, Debian package)
  - Source package (requires compilation)
- **Development version**
  - Manual installation from OOFEM GitHub repository (https://github.com/oofem/oofem.git)

2.2 Binary package installation - Windows

- Extract downloaded zip archive (oofem_2.5_AMD64.zip) into any directory
- The extraction should create oofem_2.5_AMD64 directory
- Modify PATH variable to include oofem_2.5_AMD64/lib dir

```
set PATH=C:\Users\user\Documents\oofem_2.5_AMD64\lib;%PATH%
```

- Test run

```
C:\Users\user\Documents\oofem_2.5_AMD64\bin\oofem -v
OOFEM version 2.5 (AMD64-Windows, fm;tm;sm;IML++) of Dec 30 2017 on JAJA
Copyright (C) 1994-2017 Borek Patzak
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR␣
↪PURPOSE.
```

2.3 Binary installation – Unix/Linux

You can either choose generic Linux binary or Debian package.

2.3.1 Linux – binary

- Extract downloaded zip archive (oofem_2.5_x86.tar.gz) .. code-block:: bash

  tr xzvf oofem_2.5_x86.tar.gz
- This creates oofem_2.5_x86_64/bin and oofem_2.5_x86_64/bin directories
- **Define where to find libraries**

  ```
  $ export LDD_LIBRARY_PATH=../lib:$LDD_LIBRARY_PATH
  ```
- **Run oofem**

  ```
  $ cd bin; ./oofem -v
  ```

2.3.2 Linux – Debian package

```
$ sudo apt install oofem_2.5_x86_64.deb
```

2.4 Installation from source

Requirements

- C++ compiler (g++, Visual Studio, MinGW, . . .)
- CMake build tools (https://cmake.org/)
- Controls software compilation process by using platform independent configuration and generate native make-files or projects
- To generate platform makefile or project configuration, use `cmake` (or any user friendly based GUI, such as ccmake or cmake-gui)

2.4.1 Instalation from source - Linux

In the following, we assume the sources are in /home/user/oofem.src and build directory in /home/user/oofem.build

```
$ cd /home/user/oofem.build
$ ccmake /home/user/oofem.src
$ make
```

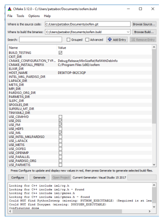
2.4.2 Installation from source - Windows

Requirements:

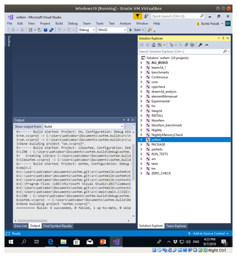
- C++ compiler (Visual Studio, MinGW)
- CMake build tools (https://cmake.org/)

Procedure:

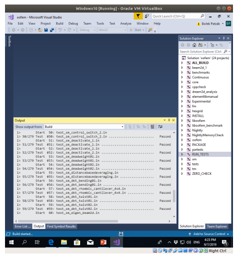
- Clone oofem git repository



- **Use cmake to generate VS solution**
  - Select compiler
  - Set source directory
  - Set build directory
  - Generate project/solution



- **Use compiler to build project targets (oofem and RUN_TESTS)**



CHAPTER

THREE

GETTING STARTED

To get OOFEM installed on yout system, please follow *Installation* instructions first.

OOFEM is console application, it should be executed from command line with arguments specifying the path to input file. When no arguments are privided, the help is printed on output.

```
$ ./oofem

Options:

    -v  prints oofem version
    -f  (string) input file name
    -r  (int) restarts analysis from given step
    -ar (int) restarts adaptive analysis from given step
    -l  (int) sets treshold for log messages (Errors=0, Warnings=1,
        Relevant=2, Info=3, Debug=4)
    -rn turns on renumbering
    -qo (string) redirects the standard output stream to given file
    -qe (string) redirects the standard error stream to given file
    -c  creates context file for each solution step

Copyright (C) 1994-2017 Borek Patzak
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

The problem to be solved is fully described in input file. The structure of input is explained in *Understanding input file* sectiona and fully documented in OOFEM Input manual.

To run oofem with specific input, use `-f` option followed by a system path to input file. For illustration, we demonstrate the execution using `beam2d_1.in` test which is a part of OOFEM test suite and is located in `tests/sm` directory.

```
$ ./oofem -f /home/user/oofem.git/tests/sm/beam2d_1.in
    ____________________________________________________
                OOFEM - Finite Element Solver
            Copyright (C) 1994-2017 Borek Patzak
    ____________________________________________________
    Computing initial guess

    StaticStructural :: solveYourselfAt - Solving step 1, metastep 1, (neq = 15)

    ...

    ANALYSIS FINISHED

    Real time consumed: 000h:00m:00s
```

(continues on next page)

(continued from previous page)

```
    User time consumed: 000h:00m:00s
    Total 0 error(s) and 0 warning(s) reported
```

By default, the text output file with results is created (as specified in input file). In this case, the `beam2d_1.out` file is created in the current working directory.

CHAPTER

FOUR

UNDERSTANDING INPUT FILE

The oofem input file fully describes the problem under consideration. Input file is a text file with specific structure, outlined below and fully documented in OOFEM Input manual.

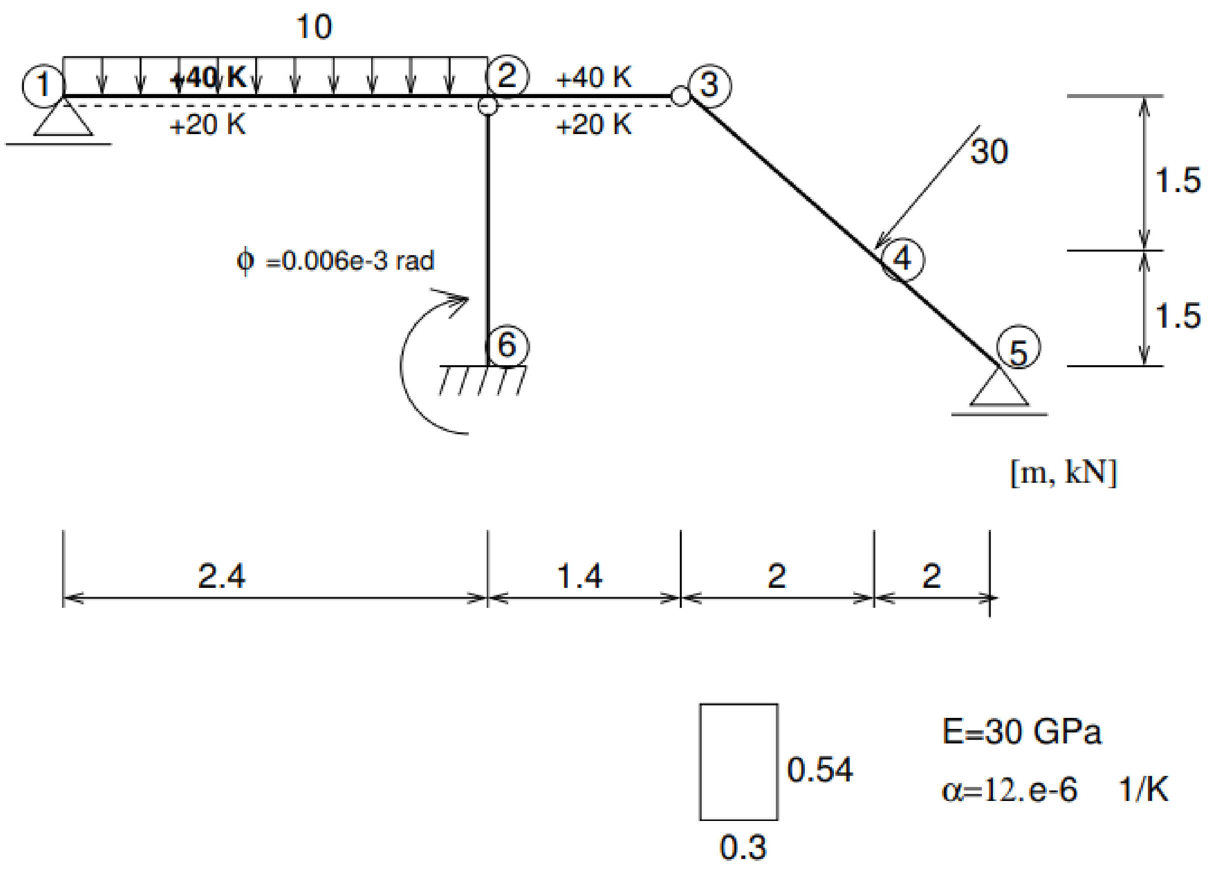
The input file can be prepared manually by any text editor or generated by special purpose code, preprocessor or converter.

Input file consists of several sections

- Ordering of sections is compulsory
- Each section can consist of one or more records (corresponding to lines)
- Each record starts with a keyword followed by compulsory or optional parameters
- Input file can contain comments (lines beginning with '#' character)
- Long Input records (lines) can be splitted using continuation character at the end of each line '\'
- Content is not case sensitive

4.1 An Example

Consider the linear elastic analysis of beam structure, as illustrated on



The full input file listing is following:

4.1.1 The details explained

The input file starts with `output file record` represented by a line containing the path to output file

Next, the `job description record` follows represented by a line with free text describing the problem

Next record, called `analysis record` determines the problem to be solved. In our case we solve linear elastic problem represented by `StaticStructural` keyword. The analysis keyword has compulsory parameter `nsteps` determining in this case the number of time steps representing load cases in our example.

Next, the domain type is described. At present version, the domain type is no longer relevant, but it is maintained for compatibility reasons. So we declare our domain type as 2dbeam. Note that oofem allows to combine different element types with different degrees of freedom per node.

The next record determines what the default output should contain. In our case, we require output for all time steps and all elements

The `components size record` determines the number of nodes, elements, cross-section and material models and other components describing the discretization and problem.

Here the `ndofman` determines the number of dof managers (nodes), the `nelem` determines the number of elements, `ncrosssect` determines the number of cross section models (oofem decouples the geometrical model of cross section from matterial model). The number of material models used is determined by `nmat` keyword. Number of boundary and initial conditions is determined by `nbc` and `nic` keywords, respectively. Number of functions, determining time variation or activity of certain component, is described using `nltf` keyword. Finally, number of sets, which are used to group entities into groups to assign for example material or boundary conditions, is determined by `nset` keyword.

Next, the nodal records follow. The number of nodes has been determined in the `components size record` and so the input contains in our case 6 records (lines) for individual nodes. The nodal records start with `node` keyword followed by its label, which is later used to refer to specific node. Labels should be unique integer numbers. A number of optional or compulsory parameters follows. Compulsory parameters determine the nodal coordinates, for example.

Next, the element records follow. The number of elements has been specified in `components size record` and so the input contains in our case 5 records (lines) for individual elements. The element type is determined by element keyword at the beginning of each record. In our case, we use `beam2d` element type (Available element types are documented in OOFEM Element manual.

The element type keyword is followed by a compuslory parameters (determining the element nodes, for example) and optional parameters (in our case determining which element degrees of freedom to condense out to represent hinge type of connection).

After element records, the records describing cross-section models follow. In our example, the cross section consists of single cross section (same dimensions) made of single material, so we use integral cross section model, represented by `SimpleCS` keyword, which is followed by cross section model label (unique integer number) followed by parameters describing cross section dimensions and associated material model (`material` keyword). The cross section model is associated to all elements in given set (‘‘set keyword).

Similar to cross section models, we follow with material models. IN our example we have a single linear isotropic material model, represented by `isole` keyword. This keyword is followed by a material model label (unique integer number) followed by a number of parameters determining material model parameters.

Next, we proceed with boundary conditions. In our example, we have 6 boundary conditions in total. The first one is Dirichlet type boundary condition, corresponding to fixed vertical displacement (prescribed is one nodal DOF corresponding to vertical displacement in z-direction: `dofs 1 3`) enforced at nodes 1 and 5 (specified using `set 4`)

The second boundary condition is fixed rotation in node 3. Yes, in node 3 we need to fix rotation, as both connected elements (2 and 3) have independent, condensed rotations, so that there is no rotation stiffness in node 3 and thus the nodal rotation has to be fixed.

The third boundary condition is used to enforce the clamped displacements and prescribed rotation at node 6.

The remaining three boundary condition represent applied constant edge load on element 1, concentrated loading at node 4, and temperature loading on element 1.

What follows is `function section` defining the functions. They are attributes of several components (boundary conditions, for example) and define, how the things evolve in time or space. In this example, the functions are only functions of time, determining the time evolution of boundary condition values. The example is divided into three load cases, in the first load case only force loading is taken into account, so all force loads (labels 4 and 5) have the first function associated (using `loadTimeFunction` keyword) which is nonzero in the first time increment and zero otherwise. Similarly, the prescribed rotation is applied exclusively in second loading step and temperature loading in the third.

The input is finally concluded with defining the sets, that are used to define group of nodes, elements, element edges, etc, on which the boundary conditions or cross sections are applied.

For more information about the input file structure, please follow OOFEM Input manual.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search