MECHPREP
User guide

Tomáš Koudelka

Department of Mechanics
Czech Technical University in Prague
Faculty of Civil Engineering

email: `tomas.koudelka@fsv.cvut.cz`

August 28, 2018

Contents

List of Tables

Part I

MECHPREP - Description

Chapter 1

Introduction

This guide describes MECHPREP command line tool which is intended for the support of input file preparation for MEFEL. The MEFEL is part of SIFEL (SImple Finite ELements) system dealing with mechanical problems. It is supposed that user gives a correct input file for MEFEL because it does not support keywords by default and the program performs few checks of the input file validity rather. On the contrary, the preprocessor input file supports and uses keywords in its input file and thus it allows for more thorough checks of input commands. The user has to prepare two input files at least and optionally, two additional files:

- file with FE mesh in SIFEL or T3D format (compulsory input file)
- file with preprocessor commands (compulsory input file)
- file with description of materials used (optional input file)
- file with description of cross sections used (optional input file)

All input files are plain text files that can be prepared in arbitrary text editor. Syntax highlighting template for Emacs/XEmacs editors can be provided for the preprocessor file format.

The guide organization is described in the following text. Installation of MECHPREP, brief description of particular source files, compilation and running commands are topics of chapter 3. Chapter 4 deals with two supported mesh formats and there is also involved a selection mechanism of element and nodal groups within the mesh format. The following two chapters 5 and 6 are dedicated to the description of optional material and cross section files respectively but the principles and formats used in these files are also exploited in the preprocessor input file. Chapter 7 deals with the format of general purpose functions defined in class `gfunct` that are often used in commands of preprocessor file. The format of preprocessor files is described in chapter 9 which describes the various commands for assignment of materials, cross sections or boundary/initial conditions to groups of elements and nodes defined in the mesh file. Chapter 8 deals specification of hanging node file format which can be useful in case of analysis of reinforced concrete structures. The second part of the manual contains several examples of the mechanical problems whose analyses were prepared with the help of MECHPREP.

Chapter 2

Notation used

In the further text, the format of file or commands are described in the following notation:

- The C format specifiers denote the value type of the given record - `%ld` denotes integer value and `%le` denotes real value.

- The list of alternative values for the given record are given in braces separated by pipes, e.g. {`1 | 5 | 8`}. In such the case, the user has to specify only one selected values from the list, e.g. optional value 5. Many optional values can be specified by keywords as well as corresponding integer value. For example, the type of general function can be given by the following options {`stat | pars | tab | pars_set | itab`} or corresponding integers {`0 | 1 | 2 | 3 | 20`}. If the user wants to specify constant function type then `stat` keyword could be used or 0 value.

- There can be also optional parts of format that should be used only on certain conditions specified in the given description. These parts are enclosed in square brackets, e.g. `[`**`slc_id`**` slc]` is an optional part of several preprocessor commands (defines subloadcase identifier) which should be used in time dependent problems only.

- In particular format records, the keywords used for the separation of particular values are written by bold mono-space font, e.g. **`slc_id`**, while the value identifier is written by normal mono-space font, e.g. `slc`.

Chapter 3

Installation of MECHPREP

All source files of SIFEL software are maintained by SVN system located on [1] where the list of versions can be found. The MECHPREP source files are involved in every version of SIFEL and the version can be downloaded on [2]. Details about the version downloading and unpacking can be found on SIFEL web pages [3] in section *Getting started - installation* together with details about the folder structure of SIFEL. The source files of MECHPREP are located in the MEFEL/PREP folder including appropriate `Makefile`.

3.1 Source files

This section contains a brief description of source files related with MECHPREP directly (the program exploits also source files of MEFEL and GEFEL) and these information can be useful for experienced users that want to extend or debug the MECHPREP code. The program is written in C++ as a plain console application whose `main` function is placed in file `mechprep.cpp`. Remaining source files are described in the the list below:

`bocon.cpp` - class with description of Dirichlet boundary conditions at nodes (see section 9.6)

`dbcrs.cpp` - class for input of cross section parameters (see chapter 6 or section 9.5)

`dbmat.cpp` - class for input of material parameters (see chapter 5 or section 9.4)

`descrip.cpp` - class that handles the input files to MECHPREP (see section 9.1)

`entityload.cpp` - class with description of continuous load on edges, surfaces and volume load (see sections 9.7 and 9.7)

`entitytdload.cpp` - class with description of time dependent continuous load on edges, surfaces and volume load

`hangnode.cpp` - class for the definition of hanging nodes in the problem (see chapter 8)

`input.cpp` - contains functions called for corresponding preprocessor commands and there is also function `input` that controls the whole reading of preprocessor input file.

`output.cpp` - contains functions that write individual parts of the resulting MEFEL input file and there is also function `output` that controls the whole output phase and creation of MEFEL input file.

`pointset.cpp` - class for set of user defined points on elements

`tempload.cpp` - class for temperature load (see section 9.6)

3.2 Compilation

In order to be able to compile MECHPREP, the user has to install and compile GEFEL and MEFEL libraries too. They can be obtained from the source files in folders GEFEL and MEFEL/SRC where corresponding `Makefile`s can be found. On Linux systems, the compilation of these dependent libraries are driven automatically by invoking `make` command in the MECHPREP folder MEFEL/PREP. For example, having user `usr` logged on computer `comp` and SIFEL root folder installed in home folder, the compilation can be run in terminal by:

```
usr@comp:~/SIFEL/MEFEL/PREP$ make
```

If compilation ran well, the `mechprep` executable file will be created and placed in two folders - SIFEL/MEFEL and SIFEL/MEFEL/_DBG. The compilation process can be adjusted by the specification of different targets in `make` command. More details about this adjustment can be found in [3] in section *Getting started - installation*.

3.3 Running of MECHPREP

The program is built as a command line tool which requires two arguments to be specified on the command line. The program can be run by the following command:

```
mechprep file.pr file.in
```

where `file.pr` is the preprocessor input file while the `file.in` is the name of the MECHPREP output, i.e. generated input file to MEFEL. Resulting `file.in` can be used for the running of MEFEL analysis by the command:

```
mefel file.in
```

Chapter 4

Formats of mesh input files

The user have to prepare FE mesh of the problem solved in his preferred mesh generator. SIFEL does not contain its own mesh generator for general 2D/3D problems. For the testing purposes, there are several simple generators of structured FE meshes for rectangular/prism domains where the dimensions of the domain and mesh density in particular directions can be specified. Additionally, these generators denotes group of nodes on individual domain edges/surfaces by unique numbers that are referenced as property numbers in further text. There are several versions of these generators depending on the type of FE element created. There is also semiautomatic mesh generator of structured mesh that was intended for the discretization of girder bridge as 3D domain originally but it can be exploited for general domains with some limitations. More details can be found on SIFEL web page [3] in section *Getting started - Preparation of input files*. Should be noted that all these generators creates mesh file in SIFEL format naturally.

Another possibility how to create FE mesh and import it to MECHPREP represents GiD software [4]. The GiD preprocessing environment contains mesh generator for arbitrary 2D/3D domains. It produces mesh file in its own format but these files are supported by MeshEditor tool where the required groups of nodes and elements can be associated with unique property number and resulting mesh file can be saved in SIFEL mesh format. For more details about this tool, see SIFEL web pages [3] in section *Getting started - Preparation of input files* or download MeshEditor from [3] in section *Tools*.

4.1 T3D format

T3D is very robust and fast generator developed by D. Rypl. It can produce 2D or 3D meshes for almost arbitrary geometry [5]. The geometry is described in plain input text file according to the following format [6]. In this file, the domain geometry is described with help of set of entities such as vertices, curves, patches, shells and regions. For every entity, the unique property number can be specified which results to assignment of this property number to group of nodes and elements generated on such entity. The output file of T3D generator can be referenced directly from the preprocessor file having the proper keyword specified in section `files` (see section 9.1)

4.2 SIFEL format

This mesh format is the default one which is used in MECHPREP. This format is also used in MeshEditor tool and simple generators mentioned at the beginning of this chapter. Should be noted that in format description, the optional keyword entries are enclosed in square brackets, e.g.

```
[opt_kwd]
```

By default, the preprocessor does not use keywords in the mesh files due to performance in case of large meshes. Additionally, the # character is used for the commenting out of all characters that follows until the end of line. The FE mesh can be represented by a plain text file with the following format given below:

```
 # Section of nodes
[num_nodes] nn
[node_id] id_1 [x] x_1 [y] y_1 [z] z_1 [numprop] npr_1 [prop] et_1 prop_1 ... [prop] et_npr_1 prop_npr_1
.
.
.
[node_id] id_nn [x] x_nn [y] y_nn [z] z_nn [numprop] npr_nn [prop] et_1 prop_1 ... [prop] et_npr_nn prop_npr_nn

 # Section of elements
[num_elements] ne
[elem_id] 1 [eltype] type_1 [enodes] n1 n2 ... ni [eprop] p [[[propedg] e1 e2 ... ej] [[propsurf] s1 s2 ... sk]]
.
.
.
[elem_id] ne [eltype] type_ne [enodes] n1 n2 ... ni [eprop] p [[[propedg] e1 e2 ... ej] [[propsurf] s1 s2 ... sk]]

 # Section of global node numbers (used only in parallel computations)
[node_id] 1 [glob_id] gnn_1
.
.
.
[node_id] nn [glob_id] gnn_nn

 # Optional section with surface property numbers (generated by MeshEditor)
faces nf
nfn_1 n_1 n_2 . . . n_nfn_1 prop_1
.
.
.
nfn_nf n_1 n_2 . . . n_nfn_nf prop_nf

 # Optional section with edge property numbers (generated by MeshEditor)
edges ned
n1_1 n2_1 prop_1
.
.
.
n1_ned n2_ned prop_ned
```

The format contains two compulsory sections, i.e. section of nodes and section of elements. Then the section of global node numbers follows in case that the mesh file represents subdomain for computations based on parallel algorithms such as domain decomposition. Additionally, there are two sections for extended selection of element groups belonging to certain surface or edge. They are added by MeshEditor tool normally in case that the user select some surface or edge and assign them a property number but they can be also provided manually by user.

4.2.1 Section of nodes

The section with description of nodes uses the following notation:

`nn` - total number of nodes in the mesh (`%ld`)

`id_i` - number of the i-th node (`%ld`) `x_i` - x coordinate of i-th node (`%le`)

`y_i` - y coordinate of i-th node (`%le`)

`z_i` - z coordinate of i-th node (`%le`)

`npr_i` - number of property identifiers assigned to i-th node (`%ld`)

`et_i` - type of entity of i-th property identifier of the given node ({`1 | 2 | 3 | 4`} see Tab. 4.1)

`prop_i` - the property identifier assigned to given entity type and node (`%ld`)

Having the total number of nodes specified, the records of individual nodes follow. Each record of node consists of node number id, three spatial coordinates x, y, z and record of property identifiers assigned to the given node. This record starts with the number of assigned property identifiers `npr` and `npr` pairs of values that describes type of entity `et`,which is associated with the given node, and property identifier of the given entity. Basically, this system of property identifiers selects group of nodes being part of some entity such as surface, edge, etc. In a preprocessor input file, these identifiers can be used in commands assigning for example boundary conditions to selected group of nodes on the entity with the given property identifier.

Entity type id	Entity shape
1	vertex / point
2	curve / edge
3	2D region / patch
4	3D region / volume

Table 4.1: Table of entity types

4.2.2 Section of elements

The section with description of elements uses the following notation:

`ne` - total number of elements in the mesh (`%ld`)

`id_i` - number of the i-th element (`%ld`)

`type_i` - type of i-th element ({`1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14`} see Tab. 4.2)

`n1...ni` - node numbers defining the given element connectivity (%ld)

`p` - property identifier of region(volume) part of which the given element is being (`%ld`)

`e1...ej` - edge property identifiers for particular edges of the given element (`%ld`)

Element type id	Element shape	Number of nodes	Number of edges	Number of surfaces
1	bar	2	1	0
2	bar	3	1	0
3	triangular	3	3	1
4	triangular	6	3	1
5	quadrilateral	4	4	1
6	quadrilateral	8	4	1
7	tetrahedron	4	6	4
8	tetrahedron	10	6	4
9	pyramid	5	8	5
10	pyramid	13	8	5
11	wedge	6	9	5
12	wedge	15	9	5
13	hexahedron	8	12	6
14	hexahedron	20	12	6

Table 4.2: Table of element types and their number of edges and surfaces

`s1...ek` - surface property identifiers for particular surfaces of the given element (`%ld`)

Having the total number of elements specified, the records of particular elements follow. Each element record consists of element id, element type, set of node identifiers defining element connectivity (depending on element type) and record of property identifiers. Each property identifier record contains region(volume) property identifier optionally followed by property identifiers of element edges and surfaces. If the given element is not associated with any region surface or edge then zeros can be specified. The identifier `p` associates the given element to region(volume) with region of the same property identifier and thus group of elements with the same region identifier can be selected. Group of selected elements can be used consequently in the preprocessor input file and the same material parameters can be assigned to them for example.

Similarly, the optional property identifiers of element edges and surfaces allow for the selection of elements associated with edge or surface having the given property identifier. If the edge or surface property identifiers are required to be given then they must be given for all elements and on all edges and all surfaces of individual elements. If some property identifier is not given to edge or surface then zero may be specified. The number of edge or surface property identifiers is given by the number of edges or surfaces of the corresponding element type (see [7]). Only one property identifier is allowed per each element edge or surface and if more is required then the additional edge or surface and its property identifier can be specified either by MeshEditor tool or manually in the optional mesh file sections (see sections 4.2.5 and 4.2.4).

Similarly to region property id, edge identifiers `e1...ej` associate the given element with the edge(curve) of the same property identifier and thus group of elements and their edges with the same edge identifier can be selected. This selection can be used consequently in the preprocessor input file and the uniform load can be assigned to these element edges for example.

The surface identifiers `s1...sk` associate the given element with the surface of the same property identifier and thus group of elements and their surfaces with the same surface identifier can be selected. This selection can be used consequently in the preprocessor input file and the uniform load can be assigned to these element surfaces for example.

4.2.3 Section of global node numbers

This section is optional and it contains the global node numbers in case that the mesh file represents subdomain for computations based on parallel algorithms such as domain decomposition. For each node, the global node number must be specified and thus `nn` pairs of integer numbers is the content of this section. The first number is the node identifier related to the nodes given the mesh file and `gnn_i` is the corresponding global node number related to the whole domain. More details can be found in [8]. This section is created automatically by SIFEL simple parallel generators or can be obtained from the mesh decomposer.

4.2.4 Section with surface property numbers

This optional section is created by MeshEditor by default but it can be also added manually by the user. The section allows for the definition of arbitrary surface with given property identifier by a list of nodes associated with that surface. The section is opened by the keyword `faces` followed by number of records with nodes used for the definition of surfaces. Each record consists of number of nodes `nfn_i`, node identifiers `n_1...n_nfn_i` and the property identifier of the i-th surface property specifier `prop_i`.

4.2.5 Section with edge property numbers

This optional section is created by MeshEditor by default but it can be also added manually by the user. The section allows for the definition of arbitrary curve with given property identifier by a list of nodes associated with that curve. The section is opened by the keyword `edges` followed by number of records with segments used for the definition of edges. Record of the i-th segment consists of two nodes identifiers `n1_i` and `n2_i` and the property identifier of the edge `prop_i` associated with the i-th segment.

4.2.6 Example of 2D mesh on rectangular domain

The mesh was generated by `gensifquad` generator which can be found in SIFEL package in folder SIFEL/PREP/SEQMESHGEN. It creates structured mesh on rectangular domain including automatic generation of edge property identifiers on elements. In this example, a rectangular domain 5.0×3.0 m (length×height) was discretized by regular mesh of quadrilateral four node elements. The mesh was generated in the *xy* plane such that the length in x direction was divided by 5 elements and height in y direction was divide by 3 elements. The mesh can be generated by the following command:

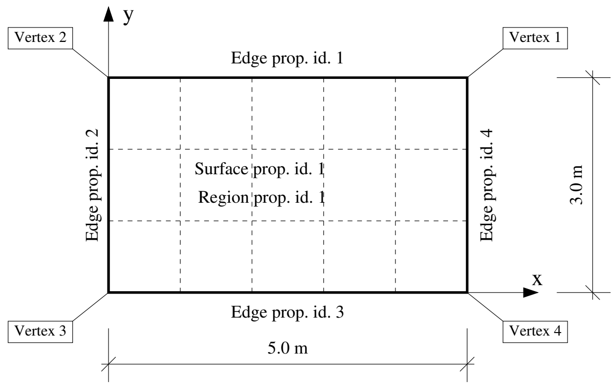


Figure 4.1: Property identifiers generated by `gensifquad` on the rectangular domain

```
gensifquad file.top 5.0 3.0 5 3 1
```

where the first argument represents the name of the output file with topology in SIFEL format, the second and third arguments stand for dimension in x and y directions, the fourth and fifth arguments represent mesh density in x and y directions and the last argument switches on the generation of edge property identifiers. The resulting mesh has the property identifiers generated as follows (see also Fig. 4.1):

- All nodes have assigned surface property identifier 1 and region property identifier 1 - see pairs of nodal property record `3 1` and `4 1` respectively.
- Ordinary inner nodes are denoted by vertex property identifier 0 - see pair `1 0` in the nodal property record.
- Corner nodes are denoted by vertex property identifiers 1, 2, 3 and 4 - see pairs `1 1`, `1 2`, `1 3` and `1 4` in the nodal property record.
- Nodes lying on edges are marked by edge property identifiers 1, 2, 3 and 4 - see pairs `2 1`, `2 2`, `2 3` and `2 4` in the nodal property record.
- All elements are denoted by region property identifier 1 and surface property identifier 1 - see the first and last value in the element property record.
- Edges of elements corresponding to domain edges are denoted by edge property identifiers 1, 2, 3 and 4 - see nonzero values in the middle quaternion values of element property record.

The content of generated file follows where the resulting mesh contains 15 elements and 24 nodes:

```
24
  1  0.00000000000e+00  0.00000000000e+00 0.0     5  1 3  2 2  2 3  3 1  4 1
  2  0.00000000000e+00  1.00000000000e+00 0.0     4  1 0  2 2  3 1  4 1
  3  0.00000000000e+00  2.00000000000e+00 0.0     4  1 0  2 2  3 1  4 1
  4  0.00000000000e+00  3.00000000000e+00 0.0     5  1 2  2 1  2 2  3 1  4 1
  5  1.00000000000e+00  0.00000000000e+00 0.0     4  1 0  2 3  3 1  4 1
  6  1.00000000000e+00  1.00000000000e+00 0.0     3  1 0  3 1  4 1
```

```
    7  1.00000000000e+00  2.00000000000e+00 0.0     3  1 0  3 1  4 1
    8  1.00000000000e+00  3.00000000000e+00 0.0     4  1 0  2 1  3 1  4 1
    9  2.00000000000e+00  0.00000000000e+00 0.0     4  1 0  2 3  3 1  4 1
   10  2.00000000000e+00  1.00000000000e+00 0.0     3  1 0  3 1  4 1
   11  2.00000000000e+00  2.00000000000e+00 0.0     3  1 0  3 1  4 1
   12  2.00000000000e+00  3.00000000000e+00 0.0     4  1 0  2 1  3 1  4 1
   13  3.00000000000e+00  0.00000000000e+00 0.0     4  1 0  2 3  3 1  4 1
   14  3.00000000000e+00  1.00000000000e+00 0.0     3  1 0  3 1  4 1
   15  3.00000000000e+00  2.00000000000e+00 0.0     3  1 0  3 1  4 1
   16  3.00000000000e+00  3.00000000000e+00 0.0     4  1 0  2 1  3 1  4 1
   17  4.00000000000e+00  0.00000000000e+00 0.0     4  1 0  2 3  3 1  4 1
   18  4.00000000000e+00  1.00000000000e+00 0.0     3  1 0  3 1  4 1
   19  4.00000000000e+00  2.00000000000e+00 0.0     3  1 0  3 1  4 1
   20  4.00000000000e+00  3.00000000000e+00 0.0     4  1 0  2 1  3 1  4 1
   21  5.00000000000e+00  0.00000000000e+00 0.0     5  1 4  2 3  2 4  3 1  4 1
   22  5.00000000000e+00  1.00000000000e+00 0.0     4  1 0  2 4  3 1  4 1
   23  5.00000000000e+00  2.00000000000e+00 0.0     4  1 0  2 4  3 1  4 1
   24  5.00000000000e+00  3.00000000000e+00 0.0     5  1 1  2 1  2 4  3 1  4 1

15
    1 5      1      5      6      2     1  3 0 0 2  1
    2 5      2      6      7      3     1  0 0 0 2  1
    3 5      3      7      8      4     1  0 0 1 2  1
    4 5      5      9     10      6     1  3 0 0 0  1
    5 5      6     10     11      7     1  0 0 0 0  1
    6 5      7     11     12      8     1  0 0 1 0  1
    7 5      9     13     14     10     1  3 0 0 0  1
    8 5     10     14     15     11     1  0 0 0 0  1
    9 5     11     15     16     12     1  0 0 1 0  1
   10 5     13     17     18     14     1  3 0 0 0  1
   11 5     14     18     19     15     1  0 0 0 0  1
   12 5     15     19     20     16     1  0 0 1 0  1
   13 5     17     21     22     18     1  3 4 0 0  1
   14 5     18     22     23     19     1  0 4 0 0  1
   15 5     19     23     24     20     1  0 4 1 0  1
```

Chapter 5

Format of a material input file

This chapter describes the format of a material input file which is optional but the same format is used in the section of preprocessor input file which has to be given if no material file has been prepared. The material input file is intended for users who prefer to manage all their material parameters from one file which is reused in several preprocessor input files.

By default in preprocessor, the keywords are used in the material files and the `#` character can be used for the commenting out of all characters that follows until the end of line. The material file may be prepared in arbitrary text editor as a plain text file and the content of such file has the following format where the bold face font denotes keywords used:

num_mat_types nmt # *number of material types*
mattype mt_1 **num_inst** nmi_1
1 mtrec_1_1
2 mtrec_1_2
.
.
nmi_1 mtrec_1_nmi_1

mattype mt_2 **num_inst** nmi_2
1 mtrec_2_1
2 mtrec_2_2
.
.
nmi_2 mtrec_2_nmi_2

.
.
.

mattype mt_{nmt} **num_inst** nmi_{nmt}
1 mtrec_nmt_1

```
2  mtrec_nmt_2
.
.
```
$\texttt{nmi}_{nmt}$ $\texttt{mtrec\_nmt\_nmi}_{nmt}$

The following notation is used in the above format:

`nmt` - the total number of different material types used(`%ld`)

$\texttt{mt}_i$ - specification of i-th material type (`{1 | 2 | 3 | 10 | 11 | 12 | 24 | 25 | 26 | 27 | 30 | 31 | 42 | 45 | 50 | 51 | 52 | 71 | 80 | 100 | 104 | 106 | 108 | 150 | 160 | 310 | 320 | 340 | 400 | 420 | 502 | 503 | 504 | 550 | 900 | 951 | 1000 | 1010 | 1030 | 1040 | 1050 | 1060}`). The material type specification can be given either by the mentioned integer values or by keywords - see Tab. 5.1 for the complete description.

$\texttt{nmi}_i$ - the number of instances of material parameter sets of the given material type (`%ld`)

`mtrec_i_j` - record of material parameter set for i-th material type and j-th instance of parameters, see section 5.1.

The number of material types and instances of parameter set depends on the problem solved. Should be noted that unused material types and their instances are simply ignored in course of the preprocessing. Every finite element in the problem must have a material assigned and pairs of material type and index of instance of material parameter set are used in such assignment.

5.1 Material parameter set record

Record of a material parameter set depends on the given material type of course. Each material type is implemented as a standalone class in MEFEL where the input of material parameters from the file is controlled by the function mattype::read(XFILE *in). In this function, the order of reading and types of material parameters are given by the calls of xfscanf function which uses similar format string as standard C function fscanf with some extensions described in [7] chapter *Iotools*. By default in material records, the keyword reading is not permitted but it can be permitted by adding of appropriate switch in the section `files` of the preprocessor input file - see 9.1.

There are two possibilities of the handling with material parameter set record. In the case of default (older) handling, there is no additional processing of the record which is simply copied as a string terminated by the newline character. The record length is limited to 1000 characters by default and if the user requires longer or multi line string then `@` character must be placed at the end of each line except of the last one. Of course, this handling performs no additional check of material records. Another approach represents the handling with the record in the same way as it is processed in MEFEL, i.e there is direct call of appropriate material function read. In such the case, material record is

Material type id	Material type keyword	Source file in MEFEL/SRC	Description
1	elisomat	elastisomat.cpp	elastic isotropic material
2	elgmat3d	elastgmat3d.cpp	elastic fully anisotropic material
3	elortomat	elastortomat.cpp	elastic orthotropic material
10	simplas1d	splas1d.cpp	simple plasticity for 1D
11	jflow	j2flow.cpp	J2 flow plasticity
24	mohrcoul	mohrc.cpp	Mohr-Coulomb plasticity
25	boermaterial	boermat.cpp	Boer's plasticity
26	druckerprager	drparg.cpp	Drucker-Prager plasticity
27	doubledrprager	doubdp.cpp	Double Drucker-Prager plasticity
30	modcamclaymat	camclay.cpp	Cam-Clay plasticity
31	modcamclaycoupmat	camclaycoup.cpp	Barcelona Cam-Clay model for coupling with the moisture transport
42	chenplast	chen.cpp	Chen plasticity material
45	hissplasmat	hissplas.cpp	HISS plasticity model (Dessai)
50	microplaneM4	microM4.cpp	Microplane model M4
51	microsimp	microSIM.cpp	simplified microplane model
52	microfibro	microfiber.cpp	
71	simvisc	simviscous.cpp	simple viscous model
80	layerplate	layplate.cpp	layered model for plates
100	scaldamage	scaldam.cpp	scalar isotropic damage
104	anisodamag	anisodam.cpp	anisotropic damage (Papa)
106	ortodamage	ortodam.cpp	orthotropic damage for concrete
108	fixortodamage	fixortodam.cpp	orthotropic damage with given orthotropy directions
150	contmat	contactmat.cpp	simple plasticity model for 2D interface elements
160	cebfipcontmat	cebfipcontactmat.cpp	CEB FIP model for 2D interface elements
310	nonlocplastmat	nonlocplast.cpp	nonlocal plasticity models
320	nonlocdamgmat	nonlocdamg.cpp	nonlocal damage models
340	nonlocalmicroM4	nonlocmicroM4.cpp	nonlocal microplane M4
400	graphm	graphmat.cpp	prescribed working diagram (1D)
420	hypoplastmat	hypoplast.cpp	hypoplastic model (Mašín)
502	creepb3	creep_b3.cpp	Bazant's B3 creep
503	creepdpl	creep_dpl.cpp	double power law creep
504	creeprs	creep_rspec.cpp	B3 creep model with continuous ret. spectrum
550	winklerpasternak	winpast.cpp	Winkler-Pasternak subsoil model for plates or beams
900	therisodilat	therisomat.cpp	simple thermal isotropic expansion model
951	relaxationeuro	relaxeuroc.cpp	EC model for tendon relaxation
1000	damage_plasticity	damplast.cpp	combination of damage-plasticity
1010	viscoplasticity	visplast.cpp	combination of viscous-plasticity
1030	creep_damage	creepdam.cpp	combination of creep and damage
1040	time_switchmat	timeswmat.cpp	change of material models in time
1050	effective_stress	effstress.cpp	effective stress concept
1060	shrinkagemat	shrinkmat.cpp	isotropic shrinkage for coupling with moisture transport

checked according to system implemented in the given material type but keywords are not used by default. If the checking of the material records would be improved then the keyword reading must be enabled in the section `files` of the preprocessor input file - see 9.1. In the MEFEL approach, the material parameters have to be written in the order given in the function `read` but there is no limitation about the number of record lines.

5.2 Example of material input file without keywords

This section contains example of the material file without keywords which can be processed either with help of default material record handling or by the MEFEL handling with keyword reading switched off. There are defined four different material types in this example:

`elisomat` - elastic isotropic material where three instances of the material parameter set are given - the first one with the Young's modulus 20 GPa and the Poisson's ratio 0.2, the second one with the Young's modulus 210 GPa and the Poisson's ratio 0.3 and the third one with the Young's modulus 8 MPa and the Poisson's ratio 0.3.

`jflow` - material model of J2 plasticity with isotropic hardening where only single instance of material parameter set is given with the yield stress f_y=200 MPa and the zero hardening modulus, the cutting plane algorithm is used for the stress return (1), the number of iterations in the cutting plane algorithm is set to 50 and the relative residuum of the yield function is 10^{-6}. For more details about the model, see `j2flow.cpp` source file in the MEFEL/SRC folder.

`druckerprager` - material model of plasticity with the Drucker-Prager yield criterion where only single instance of material parameter set is given - friction angle φ=30° (0.523599 rad), cohesion c=5 kPa, dilation angle ψ=10° (0.174533 rad), hardening parameter θ=0.35, limit cohesion c_{lim}=8 kPa, the cutting plane algorithm is used for the stress return (1), the number of iterations in the cutting plan algorithm is set to 50 and the relative residuum of the yield function is 10^{-6}.For more details about the model, see `drprag.cpp` source file in the MEFEL/SRC folder.

`scaldamage` - scalar isotropic damage model where only single instance of material parameter set is given - tensile strength f_t=3.0 MPa, softening slope u_f=2.55$\cdot10^{-5}$ m, Mazars equivalent strain norm used (7), correction of dissipated energy is on (1), general algorithm for correction of dissipated energy is used (10) which can perform up to 50 iterations in the damage parameter calculation and the relative error of residuum is set to 10^{-6}. For more details about the model, see `scaldam.cpp` source file in the MEFEL/SRC folder.

The corresponding material file is listed below and should be noted that the order of individual material types does not matter.

```
num_mat_types 4 # number of material types
mattype elisomat  num_inst 3
```

```
1  20.0e9  0.20 # elasticity parameters for concrete
2  210.0e9 0.30 # elasticity parameters for steel
3  8.0e6   0.35 # elasticity parameters for soil
mattype jflow  num_inst 1
# perfect J2 plasticity parameters for steel
1 200.0e6 0.0 1 50 1.0e-6
mattype druckerprager  num_inst 1
# Drucker-Prager plasticity with isotropic hardening for soil
1 0.523599 5.0e3 0.174533 0.35 8.0e3 1 50 1.0e-6
mattype scaldamage  num_inst 1
# scalar isotropic damage model for concrete
1 3.0e6 2.55e-5 7 1 10 50 1.0e-6
```

5.3 Example of material input file with keywords

This section contains the same example of the material file as in the section 5.2 but the keywords are used in this case. The file can be processed only with help of the MEFEL approach and the keyword reading must be switched on. The corresponding material file is listed below, the order of individual material types does not matter but the order of keywords in the particular material records is compulsory with no limitation about the number of record lines.

```
num_mat_types 4 # number of material types
mattype elisomat  num_inst 3
1  e 20.0e9  nu 0.20 # elasticity parameters for concrete
2  e 210.0e9 nu 0.30 # elasticity parameters for steel
3  e 8.0e6   nu 0.35 # elasticity parameters for soil
mattype jflow  num_inst 1
# perfect J2 plasticity parameters for steel
1 fs 200.0e6 k 0.0 cp ni 50 err 1.0e-6
mattype druckerprager  num_inst 1
# Drucker-Prager plasticity with isotropic hardening for soil
1 phi 0.523599 coh 5.0e3 psi 0.174533
  theta 0.35 clim 8.0e3 cp ni 50 err 1.0e-6
mattype scaldamage  num_inst 1
# scalar isotropic damage model for concrete
1 ft 3.0e6 uf 2.55e-5 normazar corr_on gsra ni 50 err 1.0e-6
```

Chapter 6

Format of a cross section input file

This chapter describes the format of a cross section input file which is optional but the same format is used in the section of preprocessor input file which has to be given if no cross section file has been prepared. The file format and system of processing is almost the same as for the material input file. The cross section input file is intended for users who prefer to manage all their cross section parameters from one file which is reused in several preprocessor input files.

By default in preprocessor, the keywords are used in the cross section files and the `#` character can be used for the commenting out of all characters that follows until the end of line. The cross section file may be prepared in arbitrary text editor as a plain text file and the content of such file has the following format where the bold face font denotes keywords used:

num_crsec_types ncst # *number of cross section types*
crstype cst_1 **num_inst** $ncsi_1$
1 cstrec_1_1
2 cstrec_1_2
.
.
$ncsi_1$ cstrec_1_$ncsi_1$

crstype cst_2 **num_inst** $ncsi_2$
1 cstrec_2_1
2 cstrec_2_2
.
.
$ncsi_2$ cstrec_2_$ncsi_2$

.
.
.

crstype cst_{ncst} **num_inst** $ncsi_{ncst}$

```
1  cstrec_ncst_1
2  cstrec_ncst_2
.
.
```
ncsi$_{ncst}$ cstrec_ncst_ncsi$_{ncst}$

The following notation is used in the above format:

`ncst` - the total number of different cross section types used(`%ld`)

`cst`$_i$ - specification of i-th cross section type ({`0 | 1 | 2 | 4 | 10 | 20 | 40 | 50`}). The cross section type specification can be given either by the mentioned integer values or by keywords - see Tab. 6.1 for the complete description.

`ncsi`$_i$ - the number of instances of cross section parameter sets of the given cross section type (`%ld`)

`cstrec_i_j` - record of cross section parameter set for i-th cross section type and j-th instance of parameters, see section 6.1 and Tab. 6.1.

Cross section type id	Cross section type keyword	Type of elements	Record of cross section parameter set
1	csbar2d	2D bar	area (%le) [density (%le)]
2	csbeam2d	2D beams	area (%le) moment_of_inertia_Iy (%le) shear_coefficient (%le) [density (%le)]
4	csbeam3d	3D beams	area (%le) moment_of_inertia_Ix (%le) moment_of_inertia_Iy (%le) moment_of_inertia_Iz (%le) shear_coefficient_y (%le) shear_coefficient_z (%le) local_z_base_vector (%le %le %le) [density (%le)]
10	csplanestr	2D plane elements	thickness (%le) [density (%le) concentrated_mass (%le)]
20	cs3dprob	3D space elements	[density (%le)]
40	csnodal	Layered problems def. by nodes	thickness (%le) [concentrated_mass (%le)]
50	cslayer	Layered plates	num_layers (%ld) {layer_thickness (%le)}×num_layers

Table 6.1: Table of cross section types, corresponding keywords, type of elements that can be connected with the given cross section and brief description of cross section parameters

The number of cross section types and instances of parameter set depends on the problem solved. Should be noted that unused cross section types and their instances are simply ignored in course of the preprocessing. If a finite element or node in the problem must have a cross section assigned then a pair of cross section type and index of instance of cross section parameter set is used in such assignment. In Tab. 6.1, the last column contains record of cross section parameter set where the optional parameters are given in the square brackets and they are used only in the dynamics. The cross section type is set to `nocrosssection` automatically always in the case of axisymmetric problems and 3D space problems except of dynamics where the density parameter is required and `cs3dprob` must be used.

6.1 Cross section parameter set record

Record of a cross section parameter set depends on the given cross section type and its description is given in Tab. 6.1. By default in cross section records, the keyword reading is not permitted but it can be enabled by adding of appropriate switch in the section `files` of the preprocessor input file - see 9.1.

There are two possibilities of the handling with cross section parameter set record. In the case of default (older) handling, there is no additional processing of the record which is simply copied as a string terminated by the newline character. The record length is limited to 1000 characters by default and if the user requires longer or multi line string then `@` character must be placed at the end of each line except of the last one. Of course, this handling performs no additional check of the cross section records. Another approach represents the handling with the record in the same way as it is processed in MEFEL, i.e there is direct call of appropriate cross section function `read`. In such the case, cross section record is checked according to system implemented in the given cross section type but keywords are not used by default. If the checking of the cross section records would be improved then the keyword reading must be enabled in the section `files` of the preprocessor input file - see 9.1. In the MEFEL approach, the cross section parameters have to be written in the order given in the function `read` but there is no limitation about the number of record lines.

6.2 Example of cross section input file without keywords

This section contains example of the cross section file without keywords which can be processed either with help of default cross section record handling or by the MEFEL handling with the keyword reading switched off. There are defined four different cross section types in this example:

`csbar2d` - cross section type for 2D bar elements where two instances of the cross section parameter set are given the first one with cross section area A=0.2 m^2 and the second one with the cross section area A=0.45 m^2.

`csbeam3d` - cross section type for 2D beam element where only single instance of cross section parameter set is given - cross section area A=0.06 m^2, moments of inertia to particular local beam axes $I_x=5.538\cdot10^{-4}$ m^4, $I_y=4.5\cdot10^{-4}$ m^4 and $I_z=2\cdot10^{-4}$ m^4; shear coefficients are given as $\kappa_y=\kappa_z=0.6667$ and base vector of local z axis is given as $z_l(0.0;-0.6;0.8)$.

`csplanestr` - cross section type for 2D plane elements where only single instance of cross section parameter set is given - thickness t=0.15 m.

`cs3dprob` - cross section type for 3D space elements used in eigendynamics or forced dynamics problems density is given as ρ=2500 kg/m^3.

The corresponding cross section file is listed below and should be noted that the order of individual cross section types does not matter.

```
num_crsec_types 4 # number of cross section types
crstype csbar2d  num_inst 2
 # 2D bar cross section parameters
1  0.2
2  0.45
crstype csbeam3d  num_inst 1
# 3D beam cross section parameters for rectangle 0.2x0.3 m
1 0.06 5.538e-4 4.5e-4 2.0e-4 0.6667 0.6667 0.0 -0.6 0.8
crstype csplanestr  num_inst 1
# cross section of plane problem
1 0.15
crstype cs3dprob  num_inst 1
# cross section of 3D elements used in dynamics
1 2500.0
```

6.3 Example of cross section input file with keywords

This section contains the same example of the cross section file as in the section 6.2 but the keywords are used in this case. The file can be processed only with help of the MEFEL approach and the keyword reading must be switched on. The corresponding cross section file is listed below, the order of individual cross section types does not matter but the order of keywords in the particular cross section records is compulsory with no limitation about the number of record lines.

```
num_crsec_types 4 # number of cross section types
crstype elisomat  num_inst 2
 # 2D bar cross section parameters
1  a 0.2
2  a 0.45
```

```
crstype csbeam3d  num_inst 1
# 3D beam cross section parameters for retangle 0.2x0.3 m
1 a 0.06 ix 5.538e-4 iy 4.5e-4 iz 2.0e-4
  kappa_y 0.6667 kappa_z 0.6667 loc_z 0.0 -0.6 0.8
crstype csplanestr  num_inst 1
# cross section of plane problem
1 thickness 0.15
crstype cs3dprob  num_inst 1
# cross section of 3D elements used in dynamics
1 rho 2500.0
```

Chapter 7

Format of general functions gfunct

This chapter describes format of general function record used in sections of the preprocessor input file. General functions are used for various purposes such as definitions of time dependent load, spatial dependent load, time step control, the element and nodal switching on/off, etc. The record defines a scalar function of single or multiple arguments which returns real or integer values depending on the setup. The general function record has the following format:

funct_type ft frec

where the parameters have the following meaning:

ft - the function type specifier which can be one of the following options defined either by keywords {stat | pars | tab | pars_set | itab} or by corresponding integers {0 | 1 | 2 | 3 | 20}. The meaning of type specifier is given in Tab 7.1.

frec - record of function definition for the given function type.

Function type keyword	Function type id	Description
stat	0	Constant function
pars	1	Function is defined by string with math expression which is processed by parser
tab	2	Function is defined by real values in table with various interpolation on the intervals
pars_set	3	Function is defined by table of parsed expressions
itab	20	Function is defined by integer values in table

Table 7.1: Table of general function types

Format of frec for particular function types is described in the following sections. Should noted that the general function is defined as class gfunct whose main source files gfunct.h and gfunct.cpp are placed in GEFEL folder.

7.1 Function type of `stat`

General function of type `stat` returns a real constant value regardless of argument passed. The function definition `frec` has the following format:

`const_val` `val`

where `val` represents the real value returned (%le). This type of general function can be used whenever the general function record is required except of time functions for nodal DOFs and elements where integer value is expected to be returned. Example of general function which returns constant value 0.5 follows.

`funct_type` `stat` **`const_val`** `0.5`

7.2 Function type of `pars`

General function of type `pars` defines function with help math expression written as string of characters. The string is being parsed and the function definition may contain up to four arguments. This type of function is used in the preprocessor commands for element and nodal load especially. The function definition `frec` has the following format:

`func_formula` `expr`

where `expr` represents a string with math expression (%s). The expression may define arguments of the function with help of identifiers `t`, `x`, `y` and `z` which will be evaluated as actual time (`t`) or spatial coordinates (`x`, `y`, `z`). The string with expression must not contain any whitespace characters (space, tabulator, newline, etc.) otherwise it will not be read correctly. The math expression may contain:

- basic math operators `+, -, *, /` used in C++,
- power operator `^`,
- parentheses `(,)`,
- variables `t`, `x`, `y` and `z`,
- math functions of single argument `sin`, `cos`, `tan`, `log`, `log10`, `exp`, `sec`, `cosec`, `cot`, `arcsin`, `arccos`, `arctan`, `sinh`, `cosh`, `tanh`, `arsinh`, `arcosh`, `artanh` and `abs`,
- Ludolf's number defined by sequence `pi`.

Should noted that goniometric functions expect their arguments to be in radians. In the following example, the function $f(t) = \sin(\frac{\pi}{4}t + 2.5)$ is being defined with help of parsed math expression:

`funct_type` `pars` **`func_formula`** `sin(0.25*pi*t+2.5)`

Next example defines parabolic function with single argument of spatial coordinate x $f(x) = (x - 0.7)^2 - 3.5$:

`funct_type` `pars` **`func_formula`** `(x-0.7)^2-3.5`

7.3 Function type of `tab`

General function of type `tab` defines single argument function defined on several intervals with help table of real values. The first table column contains values defining interval bounds of the function argument and the second one defines values returned at the boundaries. Additionally, the type of interpolation between bound values must be specified in the function definition record. This type of function can be used in the preprocessor commands for the time dependent load or time step control especially. The function definition `frec` has the following format:

`approx_type` `itype` **`ntab_items`** `ni`
`{`$\mathtt{x}_i$ $\mathtt{y}_i$`}×ni`

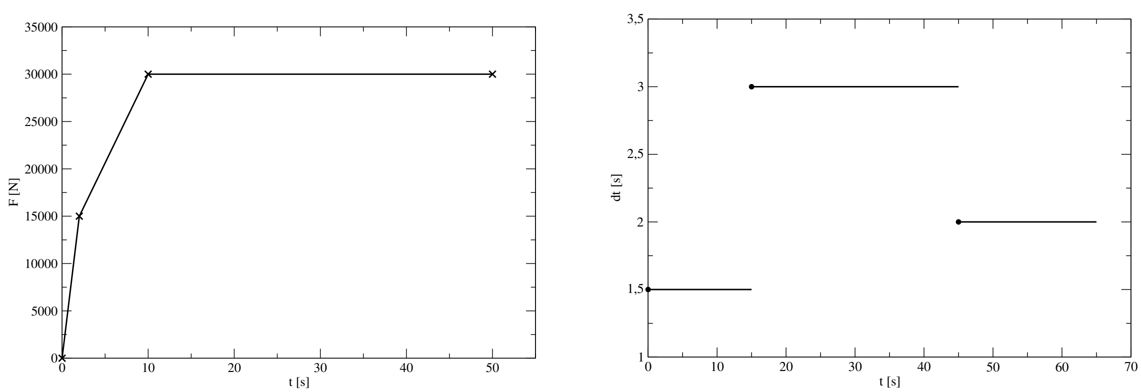
where the parameters have the following meaning:

`itype` - the interpolation type used for interpolation of values on particular intervals which can be one of the following options defined by keywords {`linear | piecewiseconst`} or by corresponding integer values {`1 | 2`}. The approximation type `linear` should be used for the definition piecewise linear function while the type `piecewiseconst` should be used for the definition of piecewise constant function (see Fig. 7.1).

`ni` - the number of rows in the table, i.e. the number of bound values defined (%ld).

$\mathtt{x}_i$ - bound value of the i-th interval (%le).

$\mathtt{y}_i$ - function value returned for the argument equaled to $\mathtt{x}_i$ (%le).



(a) (b)

Figure 7.1: General function defined by table with piecewise linear approximation (a) and piecewise constant approximations (b)

In the following example, the general function is defined according to Fig. 7.1a where a piecewise linear function is defined with help of three intervals. It is supposed that the general function would be used for the time dependent load definition and therefore the first column of the table would represent values of times where the slope of linear function changes. The first interval is defined as [0.0, 2.0), the second one is [2.0, 10.0) and the last one is [10.0, 50.0). The function values have to be defined for the lower bounds of each interval and the upper bound of the last interval.

```
funct_type tab approx_type linear
ntab_items 4
0.0   0.0
2.0   15.0e3
10.0  30.0e3
50.0  30.0e3
```

In the second example, the general function is defined according to Fig. 7.1b where a piecewise constant function is defined with help of three intervals. It is supposed that the general function would be used for the control of time step length dt and therefore the first column of the table would represent values of times where the step length changes. The first interval is defined as [0.0, 15.0) and the time step length is 1.5 s. The second interval is [15.0, 45.0) and the time step length is 3.0 s while the last one is [45.0, 65.0) and the time step length is 2.0 s. The function values have to be defined for the lower bounds of each interval and the upper bound of the last interval.

```
funct_type tab approx_type piecewiseconst
ntab_items 4
0.0   1.5
15.0  3.0
45.0  2.0
65.0  2.0
```

7.4 Function type of `pars_set`

General function of type `pars_set` defines single argument function defined on several intervals with help parsed math expressions and returns real value. It is defined with help of table where the first table column contains values defining interval bounds of the function argument and the second one defines math expressions whose evaluation results to the function value on the given interval. This type of function can be used in the preprocessor commands for the time dependent load. The function definition `frec` has the following format:

num_funct nf
{**limval** $\mathtt{lv}_i$ **func_formula** $\mathtt{expr}_i$}×nf

where the parameters have the following meaning:

`nf` - number of prescribed bounds (%ld).

$\mathtt{lv}_i$ - upper bound of i-th interval (%le).

$\mathtt{expr}_i$ - string with math expression (%s) - see section 7.2 for the format of $\mathtt{expr}_i$.

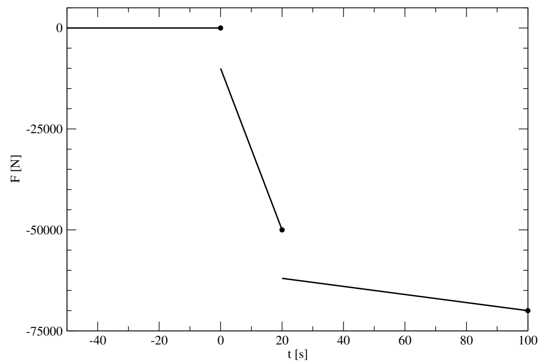


Figure 7.2: General function defined by set of parsed expressions

In the following example, the general function is defined according to Fig. 7.2 where a discontinuous piecewise linear function is defined with help of three intervals. It is supposed that the general function would be used for the time dependent load definition and therefore the first column of the table would represent values of times where the slope and values of linear function changes. The first interval is defined as $(-\infty, 0.0]$ where zero load would be applied. The second interval is (0.0 20.0] where the load increases rapidly and the last interval is $(20.0, \infty)$ where is a jump in the load intensity but it changes slowly with respect to time than in the previous interval. For each interval, a separate math expression with given linear function has to be given.

```
funct_type pars_set
num_funct  3
limval   0.0 func_formula  0.0
limval  20.0 func_formula -2.0e3*t-1.0e4
limval 100.0 func_formula -1.0e2*t-6.0e4
```

7.5 Function type of `itab`

This type of general function is accepted as a time functions that control nodal DOFs and element addition and withdrawing in case of growing mechanical problems. It is represented by a table containing times and corresponding integer values returned by time function. The format of **frec** follows:

nitab_items nit
{t_i val_i}×nit

where the parameters have the following meaning:

nit - the number of intervals on which the time function will be defined (%ld).

$\mathbf{t}_i$ - initial time of i-th interval (%le).

$\mathbf{val}_i$ - the value returned by function for time ranging in the i-th interval (%ld).

Should be noted that i-th time interval is defined as half-closed [$\mathtt{t}_i$,$\mathtt{t}_{i+1}$) and on that interval the function returns $\mathtt{val}_i$. If time argument is less than t_1 then $\mathtt{val}_1$ is being returned. If time argument is greater than $\mathtt{t}_{nit}$ then $\mathtt{val}_{nit}$ is being returned. This approach corresponds to the general function of type `tab` except of the handling with extrapolation out of bounds of the first and last interval.

In the following example, the time function is defined which withdraw elements from the computation beginning until the time 3600 s when elements are switched on for the remaining time of computation.

```
funct_type itab
nitab_items 2
0.0    0
3.6e3  1
```

Chapter 8

Format of a hanging node file

Hanging nodes can be defined as nodes that do not generate additional DOFs but the unknown displacements at hanging nodes are calculated with help of adjacent nodes of element to which the given node is connected, i.e. it hangs on the given element. This approach is useful in case of the connection of two independent meshes where nodes on mesh interfaces do not coincident. In such the case, nodes of one interface can be defined as hanging nodes and connected to the interface elements of the second mesh. Another example of typical hanging node usage represents the involving of reinforcement to concrete structure. The reinforcement is defined with help of bar elements while concrete structures are modelled by 2D or 3D elements. In the first stage, reinforcement bar elements can be defined independently on concrete elements and then intersections of bar and concrete elements are calculated and bar elements are divided according to these intersections. Basically, the intersections become nodes on the newly defined bar elements and they should be defined as hanging nodes. Each hanging node is defined by its identifier, spatial coordinates, number of nodes and their identifiers to which is connected and natural coordinates of hanging node with respect to adjacent element.

The file with hanging nodes can be prepared either manually or with help of MIDAS tool [11]. It contains a list of hanging nodes and their additional data that are added to the mesh defined by topology file (see chapter 4). In the topology file, the hanging nodes are defined as ordinary nodes with help of identifier, spatial coordinates and possible list of property numbers. In the hanging node file, the remaining data must be specified involving number of nodes and their identifiers to which is the hanging node connected and natural coordinates of hanging node with respect to adjacent element. The format of of the hanging node file is given below:

$\texttt{nhn}$
$\{\texttt{idhn}_i\ \texttt{-nadn}_i\ \{\texttt{idn}_{ij}\}\times\texttt{nadn}_i\ \texttt{ksi}_i\ \texttt{eta}_i\ \texttt{dzeta}_i\ \texttt{et}_i\}\times\texttt{nhn}$

where the meaning of particular parameters is the following:

$\texttt{nhn}$ - the total number of hanging nodes in the list (%ld),

$\texttt{idhn}_i$ - identifier of the hanging node (%ld).

$\texttt{nadn}_i$ - the number of adjacent nodes which is the given hanging node connected to (%ld).

`idn`$_{ij}$ - identifier of the j-th adjacent node of the i-th hanging node (%ld).

`ksi`$_i$`,` `eta`$_i$`,` `dzeta`$_i$ - natural coordinates in the element defined by list of adjacent nodes (%le).

`et`$_i$ - type of i-th element ({`1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14`} see Tab. 4.2).

Should b noted that in all cases, three natural coordinates `ksi`$_i$`,` `eta`$_i$`,` `dzeta`$_i$ must be given even though the element type `et`$_i$ requires lesser natural coordinate components. In such the case, the redundant components are simply ignored and therefor arbitrary values can be given instead of them.

Chapter 9

Format of a preprocessor input file

The preprocessor input file contains several sections denoted by keywords which control generation of individual parts of the resulting MEFEL input file. The preprocessor input file may be prepared in arbitrary text editor as a plain text file which contains the compulsory sections in the arbitrary order and there are also some optional sections whose order also does not matter. Each section begins with the keyword `begsec_XXX` where `XXX` is the name of the section and similarly, the end of section `XXX` is marked with the keyword `endsec_XXX`. The list of section names follows:

`files` - compulsory section which defines names of topology file, material file, cross section file and there is also setup of format and the keyword handling in these files, see 9.1.

`probdesc` - compulsory section with the description of problem solved, see 9.2

`loadcase` - compulsory section where the number of load cases and their types are given, see 9.3

`mater` - optional section with the material parameters, see 9.4

`crsec` - optional section with the cross section parameters, see 9.5

`nodvertpr` - optional section where the user can place commands for assignment of nodal properties or boundary/initial conditions related to nodal vertex property id, see 9.6

`nodedgpr` - optional section where the user can place commands for assignment of nodal properties or boundary/initial conditions related to nodal edge property id, see 9.6

`nodsurfpr` - optional section where the user can place commands for assignment of nodal properties or boundary/initial conditions related to nodal surface property id, see 9.6

`nodvolpr` - optional section where the user can place commands for assignment of nodal properties or boundary/initial conditions related to nodal region/volume property id, see 9.6

`eledgpr` - optional section where the user can place commands for assignment of element properties or boundary/initial conditions related to element edge property id, see 9.7

`elsurfpr` - optional section where the user can place commands for assignment of element properties or boundary/initial conditions related to element surface property id, see 9.7

`elvolpr` - optional section where the user can place commands for assignment of element properties or boundary conditions related to element region/volume property id, see 9.7

`outdrv` - compulsory section with setup of the MEFEL output, see 9.8

`gfunct` - compulsory section for growing mechanical problems which contains time functions of nodes or elements, see 9.9

Should be noted that each preprocessor input file must contain one optional section for nodal commands and one optional section for element commands at least. The order of sections in the file is arbitrary.

9.1 Section `files`

The section beginning is marked with the keyword `begsec_files` and the section is closed by the keyword `endsec_files`. The section contains topology file name obligatorily and material, cross section and hanging node file names optionally. There are also some mesh format specifiers and options for the handling with material and cross section files/sections. The format of the section is listed below:

```
topology_file_name
[material_file_name]
[cross_section_file_name]
mesh_format meshfmt
edge_numbering redgn
[hanging_nodes_file hangnf]
[read_mat_strings matstr]
[read_mat_kwd matkwd]
[read_crs_strings crsstr]
[read_crs_kwd crskwd]
```

where the meaning of particular identifiers follows:

`topology_file_name` - a string with the name of topology input file. The initial whitespaces are ignored while the internal spaces of the file name are not ignored. The string is terminated by the new line character and its length can be up to 1024 characters. From these limitations follows that the topology file name cannot start with whitespace characters while trailing whitespace characters are not ignored and the file name string must be given on a single line.

`material_file_name` - string with the material file name must not be specified in case that the preprocessor input file contains section `mater` whose materials are taken in such the case. If the section `mater` is not detected in the preprocessor file then the material file name must be given. The initial whitespaces are ignored while the internal spaces of the file name are not ignored. The string is terminated by the new line character and its length can be up to 1024 characters. From these limitations follows that the topology file name cannot start with whitespace characters while trailing whitespace characters are not ignored and the file name string must be given on a single line.

`cross_section_file_name` - string with the cross section file name must not be specified in case that the preprocessor input file contains section `crsec` whose cross sections are taken in such the case. If the section `crsec` is not detected in the preprocessor file then the cross section file name must be given. The initial whitespaces are ignored while the internal spaces of the file name are not ignored. The string is terminated by the new line character and its length can be up to 1024 characters. From these limitations follows that the topology file name cannot start with whitespace characters while trailing space characters are not ignored and the file name string must be given on a single line.

`meshfmt` - mesh file format specifier with optional values {`0 | 1`} or corresponding keywords {`sifel | t3d`}. See chapter 4 for more details about the format supported.

`edge_numbering` - specifier whether the edge and surface property identifiers on elements are involved in the mesh file or not. The values may be given by {`0 | 1`} where 0 means that no edge or surface property identifiers are read on elements and the 1 represents the opposite case. In cases that the problem can be defined only with help of boundary/initial conditions at nodes and element volume property identifiers then the edge and surface property identifiers need not to be involved in the mesh file (see corresponding command line options of mesh generators) and thus its size is reduced.

`hangnf` - if the keyword `hanging_nodes_file` is detected then a string with the hanging node file name must follow. The format of the hanging node file is given in the chapter 8. The initial whitespaces of the string are ignored while the internal spaces of the file name are not ignored. The string is terminated by the new line character and its length can be up to 1024 characters. From these limitations follows that the topology file name cannot start with whitespace characters while trailing space characters are not ignored and the file name string must be given on a single line.

`matstr` - optional specifier for the handling with material parameter records. If the keyword `read_mat_strings` is specified then the optional values {`0 | 1`} or corresponding keywords {`yes | no`} must be given. If the keyword `read_mat_strings` is not given then the default option `yes` is used and the material parameter records are handled as a plain strings with no parameter checking. If `no` option is given then the material parameter records are handled by `read` functions of MEFEL materials and additional parameter checking may be performed.

matkwd - optional specifier controls the keyword usage in the material parameter records. If the keyword read_mat_strings is specified then the keyword read_mat_kwd must be given followed by optional values {0 | 1} or corresponding keywords {yes | no}. If the keyword read_mat_strings is not given then the material parameter records are handled as plain strings and no keywords must be used in such the case.

crsstr - optional specifier for the handling with cross section parameter records. If the keyword read_crs_strings is specified then the optional values {0 | 1} or corresponding keywords {yes | no} must be given. If the keyword read_crs_strings is not given then the default option yes is used and the cross section parameter records are handled as a plain strings with no parameter checking. If no option is given then the cross section parameter records are handled by read functions of MEFEL cross section classes and additional parameter checking may be performed.

crskwd - optional specifier controls the keyword usage in the cross section parameter records. If the keyword read_crs_strings is specified then the keyword read_crs_kwd must be given followed by optional values {0 | 1} or corresponding keywords {yes | no}. If the keyword read_crs_strings is not given then the cross section parameter records are handled as plain strings and no keywords must be used in such the case.

It should be noted that the order of keywords in the section format is compulsory and only optional values or optional keywords and their values can be omitted.

9.2 Section probdesc

The section beginning is marked with the keyword begsec_probdesc and the section is closed by the keyword endsec_probdesc. It contains the description of the essential details about the problem solved, e.g. problem type, solver type, time step control, solution backup, type of storage of system matrices and many other similar options. All these parameters of the problem are controlled by class probdesc in MEFEL. The class contains function read which is being called for the processing of this section. Contrary to MEFEL, the keywords usage is switched on in this case. The format of this section is quite complex and it is described in details in [10].

9.3 Section loadcase

The section beginning is marked with the keyword begsec_loadcase and the section is closed by the keyword endsec_loadcase. The section contains the number of load cases and their general description. The section content strongly depends on the problem type given in the section probdesc. The formats of the section for particular problem types follows:

linear_statics problem type:

num_loadcases nlc
{**lc_id** lcid_i **temp_load_type** tlt_i}×nlc
[**num_macro_stress_comp** nmstrec {**macro_stress_comp** mcstrec}×nmstrec]
[**num_macro_strain_comp** nmstrac {**macro_strain_comp** mcstrac}×nmstrac]

mat_nonlinear_statics problem type:

num_loadcases nlc
lc_id lcid_i **temp_load_type** tlt_i×nlc
[**num_macro_stress_comp** nmstrec {**macro_stress_comp** mcstrec}×nmstrec]
[**num_macro_strain_comp** nmstrac {**macro_strain_comp** mcstrac}×nmstrac]

eigen_dynamics problem type:

num_loadcases 0

forced_dynamics problem type - there are two alternatives:

num_loadcases nlc
dload_type **timeindload**
{**lc_id** lcid_i **num_sublc** nslc_i}×nlc
{**tfunc_lc_id** lcid_i **tfunc_slc_id** slcid_j gfunct_j}×tnslc
{**tempr_type_lc_id** lcid_i **tempr_type_slc_id** slcid_j
temp_load_type tlt_j}×tnslc

or

num_loadcases nlc
dload_type **timedepload**
{**lc_id** lcid_i **temp_load_type** tlt_i}×nlc

mech_timedependent_prob problem type:

num_loadcases nlc
{**lc_id** lcid **num_sublc** nslc_i}×nlc
{**tfunc_lc_id** lcid_i **tfunc_slc_id** slcid_j gfunct_j}×tnslc
{**tempr_type_lc_id** lcid_i **tempr_type_slc_id** slcid_j
temp_load_type tlt_j}×tnslc

growing_mech_structure problem type:

num_loadcases nlc
dload_type **timeindload**
{**lc_id** lcid **num_sublc** nslc_i}×nlc
{**tfunc_lc_id** lcid_i **tfunc_slc_id** slcid_j gfunct_j}×tnslc

{**tempr_type_lc_id** lcid$_i$ **tempr_type_slc_id** slcid$_j$
temp_load_type tlt$_j$}×tnslc
{**num_pres_displ_lc_id** lcid$_i$ **num_pres_displ_slc_id** slcid$_j$
num_presc_displ npd$_j$ **presc_displ_val**
val_1 val_2 ... val$_{npd_j}$}×tnslc

where the meaning of particular identifiers follows:

nlc - the number of load cases (%ld). It must be even for nonlinear statics problems where even load cases are proportional while odd are constant ones.

lcid$_i$ - id of the i-th load case, i.e. lcid$_i$=i (%ld)

tlt$_i$ - type of the temperature load has to be given by one of the optional values {0 | 1 | 2 | 3} where 0 means no temperature load, 1 represents absolute temperature increments defined at nodes, 2 represents temperature increments defined at nodes that will be scaled by time function of the given subloadcase and 3 stands for the automatic import of temperatures values from TRFEL in case of coupled mechanical-transport analysis.

nmstrec - number of macrostress components (%ld). It is used only for the homogenization problems type of 3 (probdesc::homog=3).

mstrec - the value of macrostress component (%le). It is used only for the homogenization problems type of 3 (probdesc::homog=3).

nmstrac - number of macrostrain components (%ld). It is used only for the homogenization problems type of 4 (probdesc::homog=4).

mstrac - the value of macrostrain component (%le). It is used only for the homogenization problems type of 4 (probdesc::homog=4).

nslc$_i$ - number of subloadcases in the i-th load case (%ld). It is used only in the time dependent problems, see notes below the identifier description for more details about the subloadcase concept.

slcid$_j$ - id of the j-th subloadcase in the given i-th load case, i.e. slcid$_j$ = j) (%ld). It is used only in the time dependent problems, see notes below the identifier description for more details about the subloadcase concept.

tnslc - the total number of subloadcases in all load cases, i.e tnslc=$\sum_1^{nlc}$nslc$_i$

gfunct$_j$ - record of a time function for the j-th subloadcase of the i-th load case. The time function should return a magnitude used for the scaling of all values in the given subloadcase. The time function record is described in the section 9.9 or in [7]. It is used only in the time dependent problems, see notes below the identifier description for more details about the subloadcase concept.

`npd`$_j$ - the number of prescribed displacement values at nodes of the j-th subloadcase in the i-th load case (%ld). It is used only in the case of growing mechanical problems.

`val`$_k$ - a value of k-th prescribed value at nodes in the j-th subloadcase and i-th load case (%le). It is used only in the case of growing mechanical problems.

In the above description, a subloadcase concept is referenced. The concept is used in the forced dynamics if the `timeindload` load type is specified. The same concept is also used in time dependent problems and growing mechanical problem if the number of subloadcases `nslc` is set to nonzero. In such cases, the user can specify several load cases as usual but each load case can be subdivided into several load cases called *subloadcases*. Each subloadcase has the same format as ordinary load cases used in the linear statics problems where the constant values of load are specified. Additionally, there must be given a time function `gfunct` for each subloadcase which returns for the given time a subloadcase scale, i.e. all components of load vector assembled from the given subloadcase are multiplied by the actual `gfunct` value.

In case of growing mechanical problem, there must be also given values of prescribed displacements at nodes. These values are referenced by integer identifiers returned from the time functions used for the definition of boundary conditions in a such problem type. Basically, the number of prescribed values is given by the maximum number returned from that time functions. The values of prescribed displacements are also scaled by appropriate time function `gfunct` of the given subloadcase.

There is also another concept of the time dependent load definition which can be used in forced dynamics, time dependent problems and growing mechanical problems. The load type can be specified with `timedepload` option in forced dynamics or by settinf `nslc` to zero in time dependent problems and growing mechanical problems. In such case, all load components must be specified by individual time functions.

Both concepts cannot be used together in one preprocessor/MEFEL input file. In both concepts, arbitrary complex load case can defined but concepts differs in the size of the MEFEL input file produced. If the load components of particular load cases are indepenent and varies in time too much, it would be better to choose `timedepload` option probably but if the magnitude of most of load componets varies in time according to the same time function it is better to use `timeindload` option.

9.4 Section `mater`

The section beginning is marked with the keyword `begsec_mater` and the section is closed by the keyword `endsec_mater`. The section format is the same as the format of a material input file - see chapter 5 for more details. Should be noted that if the section `mater` is involved in the preprocessor input file then all material parameters are taken from this section and no material input file name must be specified in the `files` section. See also section 9.1 for options which controls the format of `mater` section.

9.5 Section `crsec`

The section beginning is marked with the keyword `begsec_crsec` and the section is closed by the keyword `endsec_crsec`. The section format is the same as the format of a cross section input file - see chapter 6 for more details. Should be noted that if the section `crsec` is involved in the preprocessor input file then all cross section parameters are taken from this section and no cross section input file name must be specified in the `files` section. See also section 9.1 for options which controls the format of `crsec` section.

9.6 Section `nodvertpr`, `nodedgpr`, `nodsurfpr`, `nodvolpr`

The section beginnings are marked with the keywords

`begsec_nodvertpr`, `begsec_nodedgpr`, `begsec_nodsurfpr` or `begsec_nodvolpr`

and sections are closed by keywords

`endsec_nodvertpr`, `endsec_nodedgpr`, `endsec_nodsurfpr` or `endsec_nodvolpr`.

The naming convention for these sections consists in prefix `nod` and suffix `pr`. The root of a section name represents entity type to which the section commands will be applied to, i.e. `vert` means vertex, `edg` means edges, `surf` means surfaces and `vol` means volumes or regions. Should be noted that group of nodes are defined/selected on the level of mesh file with help of two specifiers where one specifier represents the entity type while the second one denotes entity identifier so called *property id*. Such a nodal group are formed from nodes that were generated on/in the specified entity. See chapter 4 for more details about the relation between mesh and selection of nodal groups. Thus the group of selected nodes are referenced by section name which defines the entity type and property id which is involved in every command in nodal sections.

The following commands are available in nodal sections:

`ndofn` - defines the number of degrees of freedom (DOF) at nodes

`bocon` - defines Dirichlet's boundary conditions, i.e. prescribed displacements at nodes (except of growing mechanical problem)

`dof_coupl` - defines coupled DOFs at nodes (except of growing mechanical problem)

`nod_tfunc` - defines the time function identifiers in case of growing mechanical problems. The Dirichlet's boundary conditions and DOF coupling can be defined with help of these functions

`nod_crsec` - defines cross section parameters at nodes

`nod_spring` - defines spring elements, i.e. spring supports, connected to the selected nodes.

`nod_lcs` - defines nodal local coordinate systems in which the nodal equilibrium equations will be assembled `nod_load` - defines nodal concentrated load, i.e. nodal forces, independent on time

`nod_tdload` - defines time dependent concentrated load at nodes, i.e. nodal force components as independent time functions

`nod_inicond` - defines initial conditions at nodes (strains, stresses, internal variables)

`nod_temper` - defines temperature changes at nodes

Sections may be ordered arbitrarily in the input file and there are no restrictions on the command order or number of commands in particular sections. MECHPREP processes particular nodal sections in the order `nodvolpr`, `nodsurfpr`, `nodedgpr` and `nodvertpr`. If a command is being applied to the same node several times then the following operations may be performed:

merging - given assigned properties are merged together if possible (load, boundary conditions, initial conditions, coupled DOFs). For example, this operation is being performed on condition that the given property is assigned to the different direction or DOF than the ones that were assigned formerly.

comparing - given assigned property is compared to the one assigned formerly and error is reported if they differs or warning. is written to the log file if they are the same (boundary conditions, cross section, local coordinate systems, coupled DOFs).

rewriting - the properties assigned formerly are rewritten by new values (time functions, temperature changes), see the order of section processing described above. Commands in the later processed sections rewrites values assigned in sections processed former.

A detailed description of nodal commands is provided in the following subsections. For each command, the corresponding subsection contains the purpose of the command, syntax, parameter description, operations performed for multiple assignment and example of usage. Should be noted that all nodes in the mesh must have an assigned number of DOFs and usually, there must be some Dirichlet boundary conditions defined at certain nodes in order to avoid the system matrix singularity.

9.6.1 `ndofn` command

This command defines the number of DOFs at particular nodes. The command has the following syntax:

`ndofn` `ndof` **`propid`** `prop`

where the parameters have the following meaning:

`ndof` - the number of degrees of freedom at nodes (%ld).

`prop` - the property id of the given entity (%ld).

The number of DOFs assigned formerly are not rewritten and the error is reported if the different `ndof` would be assigned to the same node. The following command placed in the `nodvolpr` section defines 2 DOFs per node for nodes involved in the region with the property id equaled to 1.

ndofn 2 **propid** 1

9.6.2 `bocon` command

This command defines Dirichlet's boundary conditions at particular nodes. These conditions are represented by prescribed nodal displacements that can be either 0.0 at nodes with supports or they can be nonzero at nodes where the prescribed displacements should be applied. The command has the following syntax:

bocon propid prop num_bc nbc
{**dir** d_i **cond** val$_i$ [**lc_id** nlc$_i$ [**slc_id** slc$_i$]]}×nbc

or

bocon propid prop num_bc nbc
{**dir** d_i **gcond** gf$_i$ [**lc_id** nlc$_i$ [**slc_id** slc$_i$]]}×nbc

or

bocon propid prop num_bc nbc
{**dir** d_i **tdcond** expr$_i$ [**lc_id** nlc$_i$ [**slc_id** slc$_i$]]}×nbc

where the parameters have the following meaning:

`prop` - the property id of the given entity (%ld).

`nbc` - number of boundary conditions being assigned by the command (%ld). It corresponds to the number of DOFs constrained by this command.

d$_i$ - local DOF number (%ld) of i-th condition. This number defines direction in which the condition will be applied to. The values range from 1 to `ndof` of the given nodes.

val$_i$ - the static value of prescribed displacement in `d`-th DOF

gf$_i$ - static value of prescribed displacement in `d`-th DOF given by general function (`gfunct` record - see chapter 7),

expr$_i$ - string expression with time function description whose value defines the given prescribed displacement with respect to actual time (%s). The more details about string expressions can be found in section 7.2. This parameter may used only in case of time dependent problems or forced dynamics problems.

The following parameters are optional (including of corresponding keywords) and they must be specified only if the val$_i$≠0 and also in the case of time dependent problems:

`nlc`$_i$ - load case identifier of i-th condition in which the prescribed displacements will be involved (%ld).

`slc`$_i$ - subloadcase identifier of i-th condition in which the prescribed displacements will be involved (%ld). It must be given only for the time dependent/forced dynamics problems for time independent load scaled by time dependent factor.

Boundary conditions assigned formerly are merged with the new ones and the error is reported if the different values of prescribed displacements would be assigned to the same node and the same local DOF. The following example placed in the `nodedgpr` section fix all DOFs at nodes involved in the edge with the property id equaled to 2. The linear static plane stress problem is assumed in this case.

```
bocon propid 2 num_bc 2 dir 1 cond 0.0  dir 2 cond 0.0
```

The following command placed in `nodsurfpr` section defines prescribed displacements for the proportional load case in nonlinear static 3D problem. The displacement $1.0{\cdot}10^{-5}$ m is prescribed in the z direction to nodes on surface with property 4.

```
bocon propid 4 num_bc 1 dir 1 cond 1.0e-5 lc_id 1
```

The following command placed in `nodvertpr` section defines prescribed displacement for the time dependent load case in forced dynamics 2D problem. The displacement $2.0{\cdot}10^{-4}\sin(5t)$ m is prescribed in the y direction to node with vertex property 5.

```
bocon propid 5 num_bc 1 dir 2 tdcond 2.0e-4*sin(5.0*t) lc_id 2
```

The following command placed in `nodsurfpr` section defines prescribed displacements for the the constant load case in nonlinear static 3D problem. The prescribed displacement value in x direction varies linearly in dependence on z coordinate of the nodes on surface with property 2.

```
bocon propid 2 num_bc 1 dir 1 gcond
               funct_type pars func_formula 1.0e-5-2.0e-5*z lc_id 2
```

Another example of prescribed displacements at vertices with property 3 for time dependent 3D problem. The nodes at these vertices will be constrained in x direction where the displacement $3.0{\cdot}10^{-5}$m will be prescribed and also in z direction where the displacement $-1.0{\cdot}10^{-4}$m will be given. All these values will be defined in the the second subloadcase of the first load case. Should be noted that the command must be placed in `nodvertpr` section.

```
bocon propid 3 num_bc 2 dir 1 cond 3.0e-5 lc_id 1 slc_id 2
                        dir 3 cond -1.0e-4 lc_id 1 slc_id 2
```

9.6.3 `dof_coupl` command

This command prescribes boundary condition such that particular nodes have assigned the same DOF number in the given direction which results to the same values of displacements calculated at these nodes. The command has the following syntax:

`dof_coupl propid` `prop` **`ndir`** `nd` {**`dir`** $\mathtt{d}_i$}×`nd`

where the parameters have the following meaning:

`prop` - the property id of the given entity (%ld).

`nd` - number of directions in which the DOFs will be coupled by the command (%ld), i.e. number of coupling conditions.

$\mathtt{d}_i$ - local DOF number (%ld) of i-th condition. This number defines direction in which the condition will be applied to. The values range from 1 to `ndof` of the given nodes.

The following command placed in the `nodedgpr` section fix all DOFs in the x direction at nodes involved in the edge with the property id 8.

`dof_coupl propid` `8` **`ndir`** `1` **`dir`** `1`

9.6.4 `nod_tfunc` command

This command prescribes Dirichlet's boundary condition in case of growing mechanical problems. In such the problem type, DOFs at particular nodes must be controlled by time functions which return 0 if the given DOFs should not involved in the problem solved, i.e. they are constrained or 1 if the given DOFs are free. The positive integer value greater than 1 should be returned if the given DOFs with the same time function value are coupled. The negative integer value should be returned if there are nonzero prescribed displacements at given time. In such the case, the function value represents the the negative value of index k of prescribed values $\mathtt{val}_k$ defined in section `loadcase`, see section 9.3. All these time functions must be defined in the section `gfunct` and referenced by their identifiers. By default, the nodal DOFs are controlled by time function of adjacent elements and only supports or other special cases must be defined by this command whose syntax follows:

`nod_tfunc propid` `prop` **`ndir`** `nd` {**`dir`** $\mathtt{d}_i$ **`tfunc_id`** $\mathtt{id}_i$}×`nd`

where the parameters have the following meaning:

`prop` - the property id of the given entity (%ld).

`nd` - number of directions in which the time functions will be assigned by the command (%ld), i.e. number of boundary conditions.

$\mathtt{d}_i$ - local DOF number (%ld) of i-th condition. This number defines direction in which the condition will be applied to. The values range from 1 to `ndof` of the given nodes.

$\mathtt{id}_i$ - identifier of time function from the section **gfunct** (%ld) of i-th condition. The values range from 1 to ngf where ngf is defined in the section gfunct.

The following command placed in the **nodsurfpr** section fix all DOFs in the x and y directions at nodes involved in the surface with the property id 6. It is supposed that growing mechanical problem with plane stress elements would be solve in this case.

```
begsec_nodsurfpr
.
.
nod_tfunc propid 6 ndir 1 dir 1 tfunc_id 1 dir 2 tfunc_id 1
.
.
endsec_nodsurfpr
.
.
begsec_gfunct
time_functions
num_gfunct 5
gf_id 1 funct_type itab
nitab_items 2
0.0    0
1.0e3  0
.
.
endsec_gfunct
```

9.6.5 nod_crsec command

The command assigns cross section parameters to the selected nodes. Usually, the nodal cross section command is used in case of plane problems where the thickness should be prescribed at nodes which leads to the approximation of thickness on adjacent elements with help of element shape functions. Another possibility represents the cross section defined on elements (see section 9.7.3) where the cross section parameters are assumed to be constant on element and for example, jumps in thickness can be defined. The command has the following syntax:

```
nod_crsec propid prop type t type_id id
```

where the parameters have the following meaning:

prop - the property id of the given entity (%ld).

t - the cross section type specifier according to Tab 6.1 (keyword or %ld).

id - the cross section parameters set identifier (%ld). The identifier refers to the cross section parameter set of the given cross section type $\mathtt{t}_i$ which is defined in the section crsec or in the cross section input file. See chapter 6 for more details about the cross section specification.

The following command placed in the `nodvolpr` section defines uniform thickness 0.25 m at nodes involved in the region with the property id 0. The cross section parameters are taken from the section `crsec` included in the preprocessor input file. It is supposed that plane stress mechanical problem would be solve in this case, `crsec` section has the parameter keywords switched on by defining appropriate options in section `files`. There are also two different thicknesses (0.1 and 0.25) defined in section `crsec`.

```
begsec_files
.
.
read_crs_strings no
read_crs_kwd     yes
endsec_files
.
.
begsec_nodvolpr
.
.
nod_crsec propid 0 type csplanestr type_id 2
.
.
endsec_nodvolpr
.
.
begsec_crsec
num_crsec_types 1 # number of cross section types
crstype csplanestr  num_inst 2
# cross section of plane problem
1 thickness 0.10
2 thickness 0.25
endsec_crsec
```

9.6.6 `nod_spring` command

The command assigns spring support to the selected nodes. The spring supports are represented by spring elements in the given direction whose stiffness is governed by the material model specified. The material model is defined by the sequence of material types and corresponding identifiers of material parameter set. The number of material types in the sequence and their order depends on the first material type used. For example, elastic isotropic material is defined with help of single type specifier and corresponding parameter set id. Plasticity materials are defined by plasticity material type, where the yield stress and hardening parameters are given, and type of elastic material where the coefficients of elasticity are given. Single material type `graphm` represents another useful choice for spring elements where the nonlinear stiffness is calculated in terms of working diagram defined by `gfunct` (see chapter 7). For more details about material types and the definition of materials on elements see section 9.7.2 and chapter 5. The command has the following syntax:

nod_spring propid prop **dir** d **num_mat** nm {**type** t_i **type_id** id_i}×nm

where the parameters have the following meaning:

- `prop` - the property id of the given entity (%ld).
- `d` - the number which defines direction in which the spring support will be applied to (%ld). The values range from 1 to `ndof` of the given nodes.
- `nm` - the number of material types used in the material model description
- t_i - the material type specifier according to Tab 5.1 (keyword or %ld).
- id_i - the material parameters set identifier (%ld). The identifier refers to the material parameter set of the given material type t_i which is defined in the section `mater` or in the material input file. See chapter 5 for more details about the material specification.

The following command placed in the `nodsurfpr` section applies spring supports in the z direction (3D problem is assumed in this case) at nodes involved in the surface with the property id 8. The stiffness of these supports is set to 8.0 MN/m with help of Young's modulus of elastic isotropic model. The value of Poisson's ratio is ignored in this case.

```
begsec_nodsurfpr
.
.
nod_spring propid 8 dir 3 num_mat 1 type elisomat type_id 1
.
.
endsec_nodsurfpr
.
.
begsec_mater
num_mat_types 3 # number of material types
mattype elisomat  num_inst 1
1 e 8.0e6  nu 0.0
.
.
endsec_mater
```

The following command placed in the `nodsurfpr` section applies spring supports in the y direction at nodes involved in the surface with the property id 2. The stiffness of these spring supports will be defined with help of `graphm` material type.

```
begsec_nodsurfpr
.
.
nod_spring propid 2 dir 2 num_mat 1 type graphm type_id 1
.
.
endsec_nodsurfpr
.
.
begsec_mater
num_mat_types 1 # number of material types
# material with defined working diagram for 1D
mattype graphm  num_inst 1
 # working diagram is defined by piecewise linear function
1 gtable approx_type linear ntab_items 4

# table with four values follows
# displacement  force
-1.0     0.0
 0.0     0.0
 1.0e-3  2.5e6
 1.0     2.5e6
endsec_mater
```

In the following example, the spring supports in x and y directions are defined at nodes on the edge with the property id 1. Spring behaviour is defined by J2 plasticity model with the yield stress f_s=3 MPa and no hardening coupled with elastic isotropic model with Young's modulus E=20 MPa. Poison's ratio ν=0.0 is ignored for spring elements.

```
begsec_nodedgpr
.
.
nod_spring propid 1 dir 1 num_mat 2 type jflow type_id 1
                                    type elisomat type_id 1
nod_spring propid 1 dir 2 num_mat 2 type jflow type_id 1
                                    type elisomat type_id 1
.
.
endsec_nodedgpr
.
.
begsec_mater
num_mat_types 1 # number of material types
# J2 plasticity material type
mattype jflow  num_inst 1
fs 3.0e6 k 0.0
```

```
# elastic isotropic material
mattype elisomat  num_inst 1
e 20.0e6  nu 0.0
endsec_mater
```

9.6.7 nod_lcs command

The command defines a local coordinate system in the selected nodes which the equilibrium equations are calculated in. It is useful for the definition of rotated supports where the user can specify support, i.e. zero prescribed displacements, with help of `bocon` command and consequently, if the local coordinate system is defined in the given node then the boundary conditions are assumed to be defined in such local system. Similarly, components of nodal load are assumed to be defined in such local system. The command has the following syntax:

nod_lcs propid prop **dim** nd {**basevec** {comp$_{ij}$}×nd}×nd

where the parameters have the following meaning:

`prop` - the property id of the given entity (%ld).

`nd` - number of base vector components (%ld). It represents the dimension of the problem, i.e. `nd`=2 for 2D problems and `nd`=3 for 3D problems.

comp$_{ij}$ - j-th component of i-th base vector (%le).

The following command placed in the `nodvertpr` section applies local coordinate system $x_l y_l$ rotated 30° anticlockwise (2D problem in xy plane is assumed in this case) at node with the vertex property id 1.

```
nod_lcs propid 1 dim 2 basevec 0.8660254 0.5
                       basevec -0.5 0.8660254
```

9.6.8 nod_load command

The command defines a constant concentrated load in selected nodes. Generally, the load is represented by force components whose meaning is defined in the problem solved. The command has the following syntax:

nod_load propid prop **lc_id** nlc [**slc_id** slc] **load_comp** {v$_i$}×ndof

where the parameters have the following meaning:

`prop` - the property id of the given entity (%ld).

`nlc` - load case id which the load will be involved into (%ld).

`slc_id` - subloadcase id of the given `nlc`-th load case (%ld) which the load will be involved into. It is used only in the time dependent problems, see notes in the section 9.3.

`v`$_i$ - i-th load component (%le). The number of nodal load components is given by the number of DOFs `ndof` assigned to the nodes by the command `ndofn`.

Should be noted that if one node has got assigned load in several `nod_load` commands then the merging is performed and values of corresponding particular load components are added. In such a case, a message is written to the log file.

The following command placed in the `nodedgpr` section applies vertical forces 15 kN to nodes involved in the edge with property id 4. The forces are assigned to the second load case of 2D plane stress linear statics problem.

nod_load propid 4 **lc_id** 2 **load_comp** 0.0 1.5e4

9.6.9 `nod_tdload` command

The command defines a time dependent concentrated load in selected nodes. Generally, the load is represented by force components whose meaning is defined in the problem solved. Every load component is defined by a general time function and the command should be used only in case that `dload_type` value defined in the `loadcase` section is set to `timedepload`. The command has the following syntax:

nod_tdload propid prop **lc_id** nlc **load_comp** {vf$_i$}×ndof

where the parameters have the following meaning:

`prop` - the property id of the given entity (%ld).

`nlc` - load case id which the load will be involved into (%ld).

`vf`$_i$ - i-th load component (`gfunct` record - see chapter 7). The number of nodal load components is given by the number of DOFs `ndof` assigned to the nodes by the command `ndofn`.

Should be noted that if one node has got assigned load in several `nod_tdload` commands then the merging is performed and values of corresponding particular load components are added if it is allowed in `gfunct`. In such a case, a message is written to the log file.

The following command placed in the `nodvertpr` section applies vertical time dependent force with amplitude 10 kN and period π to nodes involved in the vertex with property id 3. The forces are assigned to the first load case of 2D plane stress forced dynamics problem.

nod_tdload propid 3 **lc_id** 1 **load_comp funct_type** stat **const_val** 0.0
funct_type pars **func_formula** 1.0e4*sin(2

9.6.10 `nod_inicond` command

The command defines initial conditions for the specific load case. The command can be used for `mat_nonlinear_statics`, `forced_dynamics`, `mech_timedependent_prob` and `growing_mech_structure` problem types. The command has the following syntax:

nod_inicond propid prop **lc_id** nlc **cond ini_cd_type** ict **nval** nv $\{v_i\}\times$nv

where the parameters have the following meaning:

- `prop` - the property id of the given entity (%ld).
- `nlc` - load case id for which the initial conditions will be defined (%ld). Should be noted that for nonlinear statics problems, the constant load case id must be specified, i.e. `nlc` must be even number.
- `ict` - initial condition type which can be specified either by one of the keywords {**none | inidisp | inistrain | inistress | iniother | inicond**} or equivalent integer identifier {0 | 1 | 2 | 4 | 8 | 16}. Details about particular options is given in the text below this list.
- `nv` - number of initial values defined in the selected nodes (%ld). The number of values depends on `ict` specified, see notes below this list for more details.
- v_i - i-th initial value (%le). The meaning of the initial values is given by the type of initial conditions `ict`, material model used and problem type solved.

Several different initial condition types can specified by `ict` whose description follows:

- `none` - initial condition is not set.
- `inidispl` - initial displacements, displacement components must be given in v_i. The number of components is given by the number of DOFs at the given nodes.
- `inistrain` - initial strain components must be given in v_i. The number of components is given by the type of problem solved.
- `inistress` - initial stress components must be given in v_i. The number of components is given by the type of problem solved.
- `iniother` - initial values of internal variables must be given in v_i. The number of components is given by the material model used.
- `inicond` - various initial condition values must be given in v_i. The number of components is given by the material model used.

The following command placed in the **nodvolpr** section applies initial conditions for modified Cam-Clay model to all nodes of region with property id 0. The initial conditions are assigned to the first load case of axisymmetric nonlinear statics problem.

```
nod_inicond propid 0 lc_id 2 cond ini_cd_type inicond nval 7
# v_kappa_ref, p_ref, pc_ini
2.67  -1.0  -50.0
# eps_x_ini, eps_y_ini, eps_r_ini, gamma_xz_ini
-1.6667e-03  -1.6667e-03  -1.6667e-03  0.0
```

9.6.11 `nod_temper` command

The command defines heat load represented by changes of temperature at nodes in the specific load case. The command should be used in accordance of load case setup given in the section `loadcase`. In case of heat load, the `temp_load_type` must be set to value 1 or 2. In later case, the values of nodal temperature changes for time dependent problems will be scaled by subloadcase time function. The command has the following syntax:

`nod_temper propid` `prop` **`lc_id`** `nlc` `[`**`slc_id`** `slc]` **`temperature`** `t`

where the parameters have the following meaning:

`prop` - the property id of the given entity (%ld).

`nlc` - load case id which the heat load will be involved into (%ld).

`slc_id` - subloadcase id of the given `nlc`-th load case (%ld) which the heat load will be involved into. It is used only in the time dependent problems, see notes in the section 9.3.

`t` - the temperature change value (%le).

The following command placed in the `nodvolpr` section applies heat load represented by change of temperature 15 K to nodes of region with property id 5. The heat load is assigned to the second subloadcase of the first load case in the time dependent mechanical problem.

`nod_temper propid` `5` **`lc_id`** `1` **`slc_id`** `2` **`temperature`** `15.0`

9.7 Section `eledgpr`, `elsurfpr`, `elvolpr`

The section beginnings are marked with the keywords

`begsec_eledgpr`, `begsec_elsurfpr` or `begsec_elvolpr`

and sections are closed by keywords

`endsec_eledgpr`, `endsec_elsurfpr` or `endsec_elvolpr`.

The naming convention for these sections consists in prefix `el` and suffix `pr`. The root of a section name represents enity type to which the section commands will be applied to, i.e. `edg` means edges, `surf` means surfaces and `vol` means volumes or regions. Should be noted that group of elements are defined/selected on the level of mesh file with help of specifiers following the element connectivity where entity identifiers, so called *property id*, are specified for volume/region in which the element is being involved. Additionally, for each element edge and surface, the property id is also given. Such a nodal group are formed from elements that were generated on/in the specified entity or whose boundaries are connected with the given entity. See chapter 4 for more details about the relation between mesh and selection of element groups. Thus the group of selected elements are referenced by section name which defines the entity type and property id which is involved in every command in element sections.

The following commands are available in element sections:

`el_type` - defines type of the element, e.g. quadrilateral element with linear shape functions. For plane elements, there is also given if the stress/strain state is plane stress or plane strain.

`el_mat` - defines material model used on the given elements.

`el_crsec` - defines cross section type for the given elements.

`el_load` - defines load on elements with help of the read_prep function of loadel class used in MEFEL. Definition of arbitrary load type - edge, surface or volume - can be accomplished with single command `el_load` on selected elements.

`edge_load` - defines traction forces on edge [N/m]. It allows for definition of load in dependence on spatial coordinates - $f(x, y, z)$. The load is applied on all adjacent elements of edge with the given property id.

`surf_load` - defines traction forces on surface [N/m^2]. It allows for definition of load in dependence on spatial coordinates - $f(x, y, z)$. The load is applied on all adjacent elements of surface with the given property id.

`volume_load` - defines a volume load [N/m^3]. It allows for definition of load in dependence on spatial coordinates - $f(x, y, z)$. The load is applied on all elements with the given volume/region property id.

`edge_tdload` - defines time dependent traction forces on edge [N/m]. It allows for definition of load in dependence on spatial coordinates too - $f(x, y, z, t)$. The time dependent load is applied on all adjacent elements of edge with the given property id.

`surf_tdload` - defines time dependent traction forces on surface [N/m^2]. It allows for definition of load in dependence on spatial coordinates too - $f(x, y, z, t)$. The time dependent load is applied on all adjacent elements of surface with the given property id.

`volume_tdload` - defines a time dependent volume load [N/m^3]. It allows for definition of load in dependence on spatial coordinates too - $f(x, y, z, t)$. The time dependent load is applied on all elements with the given volume/region property id.

`el_tfunc` - defines the time function identifiers in case of growing mechanical problems. The elements are added or withdrawn according to the returned values from this time function (element birth and death is controlled by this function).

Sections may be ordered arbitrarily in the input file and there are no restrictions on the command order or number of commands in particular sections. MECHPREP processes particular element sections in the order `elvolpr`, `elsurfpr` and `eledgpr`. If a command is being applied to the same element several times then the following operations may be performed:

merging - given assigned properties are merged together if possible (load). For example, this operation is being performed on condition that the given property is assigned to the different direction or DOF than the ones that were assigned formerly.

comparing - given assigned property is compared to the one assigned formerly and error is reported if they differs or warning. is written to the log file if they are the same (element type, materials, cross sections, local coordinate systems).

rewriting - the properties assigned formerly are rewritten by new values (time functions), see the order of section processing described above. Commands in the later processed sections rewrites values assigned in sections processed former.

A detailed description of element commands is provided in the following subsections. For each command, the corresponding subsection contains the purpose of the command, syntax, parameter description, operations performed for multiple assignment and example of usage. Should be noted that all element in the mesh must have an assigned number element type and material model. For 1D and plane elements, the cross section must be also given.

9.7.1 `el_type` command

This command defines the type of element and consequently the type of problem solved which cannot be determined on the topological basis. For example, a triangular plane element with three nodes can represent either plane stress problem, plane strain problem, axisymmetric problem, or plate problem. The command has the following syntax:

`el_type propid` `prop t [`**`strastrestate`**` s]`

where the parameters have the following meaning:

`prop` - the property id of the given entity (%ld).

`t` - the type specifier of the finite element according to Tab 9.1 (keyword or %ld).

`s` - stress/strain state specifier which can be one of the following options defined by keywords {`planestress | planestrain`} or by integers {`10 | 11`}. It is used only for the following element types specified by `t`: `planeelementlt`, `planeelementqt`, `planeelementrotlt`, `planeelementlq`, `planeelementqq`, `planeelementrotlq`.

Should be noted that elements used for 2D and axisymmetric problems must be defined in the *xy* plane. More details about the element shape functions and node ordering can be found in [9]. If the element type is assigned to element having the type assigned formerly then the types are checked and if they differs then the error is reported.

9.7.2 `el_mat` command

This command defines material models used on elements which can have assigned several material models even. The command has the following syntax:

Element type id	Element type keyword	Source file in MEFEL/SRC	Description
1	bar2d	barel2d.cpp	1D linear bar element for 2D
2	beam2d	beamel2d.cpp	beam element for 2D
3	bar3d	barel3d.cpp	1D linear bar element for 3D
4	beam3d	beamel3d.cpp	beam element for 3D
6	barq2d	barelq2d.cpp	1D quadratic bar element for 2D
7	barq3d	barelq3d.cpp	1D quadratic bar element for 3D
8	subsoilbeam	soilbeam.cpp	subsoil element for beams in 3D
10 . . . 15	spring_1 . . . spring_6	springel.cpp	spring in 1. local DOF direction . . . spring in 6. local DOF direction
20	planeelementlt	plelemlt.cpp	triangle linear plane element for 2D prob.
21	planeelementqt	plelemqt.cpp	triangle quadratic plane element for 2D prob.
22	planeelementrotlt	plelemlt.cpp	triangle linear plane element for 2D prob. with additional rotations
23	planeelementlq	plelemlq.cpp	quadrilateral linear plane element for 2D prob.
24	planeelementqq	plelemqq.cpp	quadrilateral quadratic plane element for 2D prob.
25	planeelementrotlq	plelemrotlq.cpp	quadrilateral linear plane element for 2D prob. with additional rotations
35	planequadcontact	plquadconact.cpp	four node interface element for 2D
41	cctel	cct.cpp	CCT triangle Mindlin plate element
42	dktel	dkt.cpp	DKT triangle Kirchoff plate element
43	dstel	dst.cpp	DST triangle plate element
45	q4plateel	q4plate.cpp	quadrilateral Q4 plate element
46	argyristr	argyrisplate.cpp	triangle Argyris plate element
50	subsoilplatetr	soilplatetr.cpp	triangle subsoil for plate elems.
51	subsoilplateq	soilplateq.cpp	quadrilateral subsoil for plate elems.
60	axisymmlt	axisymlt.cpp	linear triangle element for axisymmtric prob.
63	axisymmlq	axisymlq.cpp	linear quarilateral element for axisymmetric prob.
64	axisymmqq	axisymqq.cpp	quadratic quarilateral element for axisymmetric prob.
80	shelltrelem	shelltr.cpp	triangle Kirchoff shell element
81	shellqelem	shellq.cpp	quarilateral shell element
100	lineartet	lintet.cpp	linear tetrahedron element
101	quadrtet	quadtet.cpp	quadratic tetrahedron element
102	linearhex	linhex.cpp	linear brick element
103	quadrhex	quadhex.cpp	quadratic brick element

Table 9.1: Table of element types, corresponding keywords and brief description

el_mat propid prop **num_mat** nm {**type** $\mathtt{t}_i$ **type_id** $\mathtt{id}_i$}×nm

where the parameters have the following meaning:

`prop` - the property id of the given entity (%ld).

`nm` - the number of material models defined (%ld).

$\mathtt{t}_i$ - the type specifier of the i-th material model according to to Tab 5.1 (keyword or %ld).

$\mathtt{id}_i$ - the identifier of parameter set for the i-th material model (%ld). This number refers to the `id`-th parameter set of the given material type which is defined either in the `mater` section or material input file.

MEFEL uses system of material model chains where the first model specified is responsible for general material behaviour. For example, if the plasticity model with J2 criterion should be exploited then it is necessary to specify two material models on the elements. The first model must be `jflow` which establish the plasticity model on element. This model contains only parameters connected with the yield criterion directly, i.e. value of the yield stress k, hardening modulus H and setup of stress return algorithm. The elastic properties of the material must be specified by the second material type where user can specify one of the elastic material models, e.g. `elisomat`. The same principle is used in case of damage models. Usually, the last material type in the material model chain is the elastic one. There are also some exceptional material models such as `graphm`, `hypoplastmat` or `winklerpasternak` that can be used as stand alone models and do not need elastic model to be the last in model chain. Additionally, there is a special model for isotropic thermal dilatancy `therisodilat` that can be included to the any model chain at the last position.

Examples of the material model assigning follows where, for the sake of simplicity, only one type of elastic material will be assumed. All these commands are placed in section `elvolpr` usually, because different material models are connected with volume/regions of elements, and thus the various property id refers to different regions of elements.
Example of material model definition for isotropic elastic model for elements involved in region with property id 1. The elastic model takes the first parameter set defined in the `mater` section, i.e. Young's modulus E=20.0 GPa and Poisson's ratio ν=0.20 Should be noted that `mater` section has the parameter keywords switched on by defining appropriate options in section `files`:

```
begsec_elvolpr
.
.
# elastic isotropic material, the first parameter set
el_mat propid 1 num_mat 1 type elisomat type_id
.
.
endsec_elvolpr
```

```
begsec_mater
num_mat_types 4 # number of material types
mattype elisomat  num_inst 2
1  e 20.0e9  nu 0.20 # elasticity parameters for concrete
2  e 5.0e6   nu 0.3 # elasticity parameters for soil
.
.
endsec_mater
```

Example of material model definition for the same isotropic elastic model but with thermal dilatancy now. The thermal dilatancy coefficient is taken from the second parameter set, i.e. α=1.0·10^{-5}. The definition of this chain can be used also instead of simple elastic model definition if thermal dilatancy is required in more advanced material models:

```
begsec_elvolpr
.
.
# elastic isotropic material with thermal dilatancy,
# the first parameter set
el_mat propid 1 num_mat 2 type elisomat type_id 1
                          type therisodilat type_id 2
.
.
endsec_elvolpr
begsec_mater
num_mat_types 4 # number of material types
mattype elisomat  num_inst 2
1  e 20.0e9  nu 0.20 # elasticity parameters for concrete
2  e 5.0e6   nu 0.3 # elasticity parameters for soil

mattype therisodilat  num_inst 2
1  alpha 1.2e5 # dilatancy parameter for steel
2  alpha 1.0e5 # dilatancy parameter for concrete
.
.
endsec_mater
```

Example of material model definitions for plasticity models:

```
# j2 flow plasticity material, the second parameter set
el_mat propid 1 num_mat 2 type jflow type_id 2
                          type elisomat type_id 1

# Mohr-Coulomb plasticity material, the first parameter set
el_mat propid 8 num_mat 2 type mohcoul type_id 1
                          type elisomat type_id 1
```

```
# Modified Cam-Clay plasticity material, the third parameter set
el_mat propid 11 num_mat 2 type modcamclaymat type_id 3
                           type elisomat type_id 1
```

Example of material model definitions for damage models:

```
# scalar isotropic damage material, the first parameter set
el_mat propid 3 num_mat 2 type scaldamage type_id 1
                          type elisomat type_id 1

# orthotropic damage material, the second parameter set
el_mat propid 1 num_mat 2 type ortodamage type_id 2
                          type elisomat type_id 1
```

If the nonlocal approach should be exploited then the material model definitions should be given as follows:

```
# nonlocal scalar isotropic damage material, the first parameter set
el_mat propid 4 num_mat 3 type nonlocdamgmat type_id 1
                          type scaldamage type_id 3
                          type elisomat type_id 1

# nonlocal J2 flow plasticity material, the third parameter set
el_mat propid 5 num_mat 3 type nonlocplastmat type_id 3
                          type jflow type_id 2
                          type elisomat type_id 1
```

A simple visco-plastic material model with J2 criterion can be assigned by the following command:

```
# simple visco-plastic material, the first parameter set
el_mat propid 5 num_mat 4 type viscoplasticity type_id 1
                          type simvisc type_id 2
                          type jflow type_id 2
                          type elisomat type_id 1
```

Combination of damage and plasticity models can be arrived at:

```
# combination of Drucker-Prager plasticity and
# scalar isotropic damage material
el_mat propid 1 num_mat 4 type damage_plasticity type_id 1
                          type druckerprager type_id 4
                          type scaldamage type_id 3
                          type elisomat type_id 1
```

Combination of B3 creep model damage plasticity models can be arrived at:

```
# combination of B3 creep and and scalar isotropic damage material
el_mat propid 6 num_mat 4 type creep_damage type_id 1
                          type creepb3 type_id 2
                          type scaldamage type_id 1
                          type elisomat type_id 1
```

9.7.3 el_crsec command

The command assigns cross section parameters to the selected elements. Usually, the element cross section command is used in case of plane problems where the thickness should be prescribed to be constant on all elements. The second possibility represents the cross section defined at nodes (see section 9.6.5) where the cross section parameters are assumed to be given at nodes and approximated over the elements with help of shape functions. Another class of elements that requires the setting of cross section type is represented by 1D elements such as bars and beams. The command has the following syntax:

```
el_crsec propid prop type t type_id id
```

where the parameters have the following meaning:

prop - the property id of the given entity (%ld).

t - the cross section type specifier according to Tab 6.1 (keyword or %ld).

id - the cross section parameters set identifier (%ld). The identifier refers to the cross section parameter set of the given cross section type $\mathtt{t}_i$ which is defined in the section `crsec` or in the cross section input file. See chapater 6 for more details about the cross section specification.

The following command placed in the `elvolpr` section defines uniform section area 0.25 m^2 on elements involved in the region with the property id 5. The cross section parameters are taken from the section `crsec` included in the preprocessor input file. It is supposed that 2D truss beam problem would be solve in this case, `crsec` section has the parameter keywords switched on by defining appropriate options in section `files`. There are also two different section areas (0.1 and 0.25) defined in section `crsec`.

```
begsec_files
.
.
read_crs_strings no
read_crs_kwd     yes
endsec_files
.
.
begsec_elvolpr
.
.
```

```
el_crsec propid 0 type csbar2d type_id 2
.
.
endsec_elvolpr
.
.
begsec_crsec
num_crsec_types 1 # number of cross section types
crstype csbar2d  num_inst 2
# cross section of 2D bar
1 a 0.10
2 a 0.25
endsec_crsec
```

9.7.4 `el_load` command

The command defines load on elements with help of the read_prep function of loadel class (MEFEL/SRC/loadel.cpp) used in MEFEL directly. Definition of arbitrary load type - edge, surface or volume - can be accomplished with single command `el_load` on selected elements. This command can be useful on structured meshes where all elements are generated with the same orientation and order of edges and surfaces. For example, the command can be used for the prescribing of the compacting load in the problem of growing structure made from layers of soil. The surface of the topmost soil layer should be compacted by load but this surface is shared also by elements from layer added consequently which leads to application of surface load with double intensity in such cases. If the region of appropriate elements is defined with different property id on each surface compacted then this command can be applied on these regions and the correct compacting can be accomplished. The command has the following syntax:

el_load propid prop **lc_id** nlc [**slc_id** slc] **load_type** tl eloadrec

where the parameters have the following meaning:

`prop` - the property id of the given entity (%ld).

`nlc` - load case id which the load will be involved into (%ld).

`slc_id` - subloadcase id of the given `nlc`-th load case (%ld) which the load will be involved into. It is used only in the time dependent problems, see notes in the section 9.3.

`tl` - type of applied load. It can be one of the following options {`volume | edge | surface`} or corresponding integers {`1 | 2 | 3`}. The option `volume` corresponds to the application of volume load [N/m^3], option `edge` corresponds to the application of edge load [N/m] and option `surface` corresponds to the application of surface load [N/m^2].

`eloadrec` - record of element load.

The content of `eloadrec` is specific for particular load types `tl`. The format for `tl=volume` reads:

ncomp nc **load_comp** {val$_i$}×nc

where the parameters have the following meaning:

`nc` - the number of volume load components (%ld).

`val`$_i$ - the value of i-th load component in [N/m^3] (%le).

In the case that `tl=edge`, the format reads:

nedge ned **ncomp** nc {**coord_sys** lcs$_i$ [**load_comp** {val$_{ij}$}×nc]}×ned

where the parameters have the following meaning:

`ned` - the number of edges on elements where the edge load will be applied to (%ld)

`nc` - the number of edge load components (%ld).

`lcs`$_i$ - specifier of a coordinate system of the i-th element edge in which the edge load components will be given. There are available the following options {`0 | 1 | 2`} where 0 means no load applied on the given edge, 1 means edge load components given in the global coordinate system and 2 means edge load components given in the local coordinate system of the given i-th edge.

`val`$_{ij}$ - the value of j-th load component in [N/m] on the i-th element edge (%le). These values including prefix keyword `load_comp` must be specified only if the `lcs`$_i$ is set to nonzero value.

In the case that `tl=surface`, the format reads:

nsurf nsf **ncomp** nc {**coord_sys** lcs$_i$ [**load_comp** {val$_{ij}$}×nc]}×nsf

where the parameters have the following meaning:

`nsf` - the number of surfaces on elements where the surface load will be applied to (%ld)

`nc` - the number of surface load components (%ld).

`lcs`$_i$ - specifier of a coordinate system of the i-th element surface in which the surface load components will be given. There are available the following options {`0 | 1 | 2`} where 0 means no load applied on the given surface, 1 means surface load components given in the global coordinate system and 2 means surface load components given in the local coordinate system of the given i-th surface.

`val`$_{ij}$ - the value of j-th load component in [N/m^2] on the i-th element surface (%le). These values including prefix keyword `load_comp` must be specified only if the `lcs`$_i$ is set to nonzero value.

The load is merged in the case of multiple assignment for the same element and the new values of load are added to the previous ones. If there is conflict in the direction of the load component (global x local coordinate systems) the error is reported otherwise a message about the successful merging is written to the log file.

In case of forced dynamics, time dependent and growing mechanical problems, it should be noted that the command cannot be used in one preprocessor file with `el_tdload, edge_tdload, surface_tdload and volume_tdload` commands where the *timedepload* concept is used. See 9.3 for more details about different load case concepts.

The following command placed in the section `elvolpr` assigns uniform surface load 6 kN/m^2 in the z global direction on the third surface of each element involved in the region with property id 5. It is assumed that a 3D linear statics problem is being solved where the mesh consists of linear tetrahedron elements and load is included to the first load case.

```
el_load propid 5 lc_id 1 load_type surface nsurf 4  ncomp 3
coord_sys 0
coord_sys 0
coord_sys 1 load_comp 0.0 0.0 6.0e3
coord_sys 0
```

The following command placed in the section `eledgpr` assigns uniform edge load 2 kN/m in the local x direction on the second edge of each adjacent element of the edge with property id 3. It is assumed that a 2D time dependent mechanical problem is being solved where the mesh consists of linear quadrilateral elements and load is included to the second subloadcase of the first load case.

```
el_load propid 3 lc_id 1 slc_id 2 load_type edge nsurf 4  ncomp 2
coord_sys 0
coord_sys 2
load_comp 2.0e3 0.0
coord_sys 0
coord_sys 0
```

The following command placed in the section `elvolpr` assigns constant volume load 18 kN/m^3 in the z global direction for each element involved in the region with property id 1. It is assumed that a 3D linear statics problem is being solved and load is included to the first load case.

```
el_load propid 1 lc_id 1 load_type volume ncomp 3
load_comp 0.0 0.0 18.0e3
```

9.7.5 `edge_load` command

The command assigns edge load to element edges with the given edge property id. It must be placed only in the `eledgpr` section otherwise it is ignored. The load components may be either constant or the intensity of the load components may vary with respect to spatial coordinates according to the expression specified. The command has the following syntax:

```
edge_load propid prop lc_id nlc [slc_id slc] ncomp nc
func_type ft coord_sys lcs load_comp {val_i}×nc
```

where the parameters have the following meaning:

`prop` - the property id of the edge where the load will be applied to (%ld).

`nlc` - load case id which the load will be involved into (%ld).

`slc_id` - subloadcase id of the given `nlc`-th load case (%ld) which the load will be involved into. It is used only in the time dependent problems, see notes in the section 9.3.

`nc` - the number of edge load components (%ld).

`ft` - type of definition of load component values. There are available the following options {`stat | pars`} or corresponding integer values {`0 | 1`} where `stat` means constant load components and `pars` means components defined with help of expression string.

`lcs` - specifier of a coordinate system on element edges in which the edge load components will be given. There are available the following options {`1 | 2`} where 1 means edge load components given in the global coordinate system and 2 means edge load components given in the local coordinate system of the given element edge.

$\texttt{val}_i$ - the value of i-th edge load component in [N/m]. If `ft=stat` then $\texttt{val}_i$ is represented by real value (%le). If `ft=pars` then string expression is expected (%s). The expression may be composed from standard math operators (+,-,/,*), constant values or standard functions (sin, cos, exp, tan, pow, log). The expression is used for calculation of load intensity at nodes on the given edge according to their spatial coordinates which must be denoted by x, y or z in the expression string. The maximum expression string length is 1000 characters and it must not contain any space characters. See section 7.2 for more details about option `pars` and the expression strings.

The load is merged in the case of multiple assignment for the same element and the new values of load are added to the previous ones. If there is conflict in the direction of the load component (global x local coordinate systems) the error is reported otherwise a message about the successful merging is written to the log file. In case of forced dynamics, time dependent and growing mechanical problems, it should be noted that the command cannot be used in one preprocessor file with **`el_tdload, edge_tdload, surface_tdload and volume_tdload`** commands where the *timedepload* concept is used. See 9.3 for more details about different load case concepts.

The following command placed in the section `eledgpr` assigns constant uniform edge load 4 kN/m in the z global direction on each element edge which is involved in edge with property id 2. It is assumed that a 2D plane stress linear statics problem is being solved and load is included to the first load case.

```
edge_load propid 2 lc_id 1 ncomp 2
func_type stat coord_sys 1 load_comp 0.0 4.0e3
```

The following command placed in the section `eledgpr` assigns edge load with linear course along the edge with property id 3 which is parallel with the global x axis. The edge begins at node with x=0.0 m where the load intensity will be 2 kN/m while the end of the edge is at node with x=4.0 m and there is load intensity 14 kN/m. It is assumed that a 2D plane stress time dependent problem is being solved, the load acts in the z global direction and load is included to the first subloadcase of the first load case.

```
edge_load propid 3 lc_id 1 slc_id 1 ncomp 2
func_type pars coord_sys 1 load_comp 0.0  3.0e3*x+2.0e3
```

9.7.6 `surf_load` command

The command assigns surface load to element surfaces with the given surface property id. It must be placed only in the `elsurfpr` section otherwise it is ignored. The load components may be either constant or the intensity of the load components may vary with respect to spatial coordinates according to the expression specified. The command has the following syntax:

surf_load propid prop **lc_id** nlc [**slc_id** slc] **ncomp** nc
func_type ft **coord_sys** lcs **load_comp** {val$_i$}×nc

where the parameters have the following meaning:

`prop` - the property id of the surface where the load will be applied to (%ld).

`nlc` - load case id which the load will be involved into (%ld).

`slc_id` - subloadcase id of the given `nlc`-th load case (%ld) which the load will be involved into. It is used only in the time dependent problems, see notes in the section 9.3.

`nc` - the number of surface load components (%ld).

`ft` - type of definition of load component values. There are available the following options {`stat | pars`} or corresponding integer values {`0 | 1`} where `stat` means constant load components and `pars` means components defined with help of expression string.

`lcs` - specifier of a coordinate system on element surfaces in which the surface load components will be given. There are available the following options {`1 | 2`} where 1 means surface load components given in the global coordinate system and 2 means surface load components given in the local coordinate system of the given element surface.

val$_i$ - the value of i-th surface load component in [N/m^2]. If `ft`=`stat` then val$_i$ is represented by real value (%le). If `ft`=`pars` then expression string is expected (%s). The expression may be composed from standard math operators (+,-,/,*), constant values or standard functions (sin, cos, exp, tan, pow, log). The expression is used for calculation of load intensity at nodes on the given surface according to their spatial coordinates which must be denoted by x, y or z in the expression string. The maximum expression string length is 1000 characters and it must not contain any space characters. See section 7.2 for more details about option `pars` and the expression strings.

The load is merged in the case of multiple assignment for the same element and the new values of load are added to the previous ones. If there is conflict in the direction of the load component (global x local coordinate systems) the error is reported otherwise a message about the successful merging is written to the log file. In case of forced dynamics, time dependent and growing mechanical problems, it should be noted that the command cannot be used in one preprocessor file with **`el_tdload, edge_tdload, surface_tdload and volume_tdload`** commands where the *timedepload* concept is used. See 9.3 for more details about different load case concepts.

The following command placed in the section `elsurfpr` assigns constant uniform surface load 1.5 kN/m^2 in the x global direction on each element surface which is involved in surface with property id 2. It is assumed that a 3D linear statics problem is being solved and load is included to the first load case.

```
surf_load propid 2 lc_id 1 ncomp 3
func_type stat coord_sys 1 load_comp 1.5e3 0.0 0.0
```

The following command placed in the section `elsurfpr` assigns surface load with linear course across the surface with property id 4 which is parallel with the global xy plane. It is assumed that a 3D time dependent problem is being solved, the load acts in the z global direction and load is included to the first subloadcase of the first load case.

```
surf_load propid 4 lc_id 1 slc_id 1 ncomp 3
func_type pars coord_sys 1 load_comp 0.0  0.0 6.0e3*x+3.5e3*y+2.0e3
```

9.7.7 `volume_load` command

The command assigns volume load to elements involved in the region/volume with the given volume property id. It must be placed only in the `elvolpr` section otherwise it is ignored. The load components may be either constant or the intensity of the load components may vary with respect to spatial coordinates according to the expression specified. The command has the following syntax:

`volume_load propid` `prop` **`lc_id`** `nlc [`**`slc_id`** `slc]` **`ncomp`** `nc`
`func_type` `ft` **`load_comp`** `{val`$_i$`}`$\times$`nc`

where the parameters have the following meaning:

`prop` - the property id of the region/volume where the load will be applied to (%ld).

`nlc` - load case id which the load will be involved into (%ld).

`slc_id` - subloadcase id of the given `nlc`-th load case (%ld) which the load will be involved into. It is used only in the time dependent problems, see notes in the section 9.3.

`nc` - the number of volume load components (%ld).

`ft` - type of definition of load component values. There are available the following options {`stat | pars`} or corresponding integer values {`0 | 1`} where `stat` means constant load components and `pars` means components defined with help of expression string.

$\mathtt{val}_i$ - the value of i-th volume load component in [N/m^3]. If `ft`=`stat` then $\mathtt{val}_i$ is represented by real value (%le). If `ft`=`pars` then expression string is expected (%s). The expression may be composed from standard math operators (+,-,/,*), constant values or standard functions (sin, cos, exp, tan, pow, log). The expression is used for calculation of load intensity at nodes of the given region/volume according to their spatial coordinates which must be denoted by x, y or z in the expression string. The maximum expression string length is 1000 characters and it must not contain any space characters. See section 7.2 for more details about option `pars` and the expression strings.

The load is merged in the case of multiple assignment for the same element and the new values of load are added to the previous ones. A message about the successful merging is written to the log file. In case of forced dynamics, time dependent and growing mechanical problems, it should be noted that the command cannot be used in one preprocessor file with `el_tdload, edge_tdload, surface_tdload and volume_tdload` commands where the *timedepload* concept is used. See 9.3 for more details about different load case concepts.

The following command placed in the section `elvolpr` assigns constant volume load 23 kN/m^3 in the z global direction on each element which is involved in region with property id 2. It is assumed that a 3D linear statics problem is being solved and load is included to the first load case.

```
volume_load propid 2 lc_id 1 ncomp 3
func_type stat load_comp 0.0 0.0 2.3e4
```

The following command placed in the section `elvolpr` assigns volume load, with proportional change of intensity with respect to the x axis, to all elements involved in region with property id 4. It is assumed that a 3D time dependent problem is being solved, the load acts in the z global direction and load is included to the first subloadcase of the first load case.

```
volume_load propid 4 lc_id 1 slc_id 1 ncomp 3
func_type pars load_comp 0.0  0.0 1.5e4+0.3e3*x
```

9.7.8 `el_tdload` command

The command defines time dependent load on elements with the help of the read_prep function of loadel class (MEFEL/SRC/loadel.cpp) used in MEFEL directly. Definition of arbitrary load type - edge, surface or volume - can be accomplished with single command `el_load` on selected elements. This command can be useful on structured meshes where all elements are generated with the same orientation and order of edges and surfaces.

For example, the command can be used for the prescribing of the compacting load in the problem of growing structure made from layers of soil. The surface of the topmost soil layer should be compacted by load but this surface is shared also by elements from layer added consequently which leads to application of surface load with double intensity in such cases. If the region of appropriate elements is defined with different property id on each surface compacted then this command can be applied on these regions and the correct compacting can be accomplished. The command has the following syntax:

el_load **propid** prop **lc_id** nlc **load_type** tl eloadrec

where the parameters have the following meaning:

prop - the property id of the given entity (%ld).

nlc - load case id which the load will be involved into (%ld).

tl - type of applied load. It can be one of the following options {volume | edge | surface} or corresponding integers {1 | 2 | 3}. The option volume corresponds to the application of volume load [N/m^3], option edge corresponds to the application of edge load [N/m] and option surface corresponds to the application of surface load [N/m^2].

eloadrec - record of element load.

The content of eloadrec is specific for particular load types tl. The format for tl=volume reads:

ncomp nc **load_comp** {gf$_i$}×nc

where the parameters have the following meaning:

nc - the number of volume load components (%ld).

gf$_i$ - the gfunct record of i-th load component in [N/m^3]. The function type may be one of the following options {stat | pars | tab | pars_set }. If function type is {pars | pars_set } then the expression is used for calculation of load intensity at nodes of the given region/volume according to their spatial coordinates which must be denoted by x, y or z in the expression string and t is considered as time coordinate. If function type is {tab} then time dependence is considered only. See section 7 for more details about gfunct record.

In the case that tl=edge, the format reads:

nedge ned **ncomp** nc {**coord_sys** lcs$_i$ [**load_comp** {gf$_{ij}$}×nc]}×ned

where the parameters have the following meaning:

ned - the number of edges on elements where the edge load will be applied to (%ld)

nc - the number of edge load components (%ld).

$\texttt{lcs}_i$ - specifier of a coordinate system of the i-th element edge in which the edge load components will be given. There are available the following options {`0 | 1 | 2`} where 0 means no load applied on the given edge, 1 means edge load components given in the global coordinate system and 2 means edge load components given in the local coordinate system of the given i-th edge.

$\texttt{gf}_{ij}$ - the `gfunct` record of j-th load component in [N/m] on i-th element edge. The function type may be one of the following options {`stat | pars | tab | pars_set` }. If function type is {`pars | pars_set` } then the expression is used for calculation of load intensity at nodes of the selected elements according to their spatial coordinates which must be denoted by x, y or z in the expression string and t is considered as time coordinate. If function type is {`tab`} then time dependence is considered only. See section 7 for more details about `gfunct` record. These records including prefix keyword `load_comp` must be specified only if the $\texttt{lcs}_i$ is set to nonzero value.

In the case that `tl=surface`, the format reads:

nsurf nsf **ncomp** nc {**coord_sys** $\texttt{lcs}_i$ [**load_comp** {$\texttt{gf}_{ij}$}×nc]}×nsf

where the parameters have the following meaning:

`nsf` - the number of surfaces on elements where the surface load will be applied to (%ld)

`nc` - the number of surface load components (%ld).

$\texttt{lcs}_i$ - specifier of a coordinate system of the i-th element surface in which the surface load components will be given. There are available the following options {`0 | 1 | 2`} where 0 means no load applied on the given surface, 1 means surface load components given in the global coordinate system and 2 means surface load components given in the local coordinate system of the given i-th surface.

$\texttt{gf}_{ij}$ - the `gfunct` record of j-th load component in [N/m^2] on i-th element surface. The function type may be one of the following options {`stat | pars | tab | pars_set` }. If function type is {`pars | pars_set` } then the expression is used for calculation of load intensity at nodes of the selected elements according to their spatial coordinates which must be denoted by x, y or z in the expression string and t is considered as time coordinate. If function type is {`tab`} then time dependence is considered only. See section 7 for more details about `gfunct` record. These records including prefix keyword `load_comp` must be specified only if the $\texttt{lcs}_i$ is set to nonzero value.

The load is merged in the case of multiple assignment for the same element and the new values of load are added to the previous ones. If there is conflict in the direction of the load component (global x local coordinate systems) the error is reported otherwise a message about the successful merging is written to the log file.

The command is intended for use in forced dynamics, time dependent and growing mechanical problems and it should be noted that the command cannot be used in one preprocessor file with `el_load, edge_load, surface_load and volume_load` commands where the subloadcase concept is used. See 9.3 for more details about different load case concepts.

The following command placed in the section **elvolpr** assigns uniform surface load 6 kN/m^2 in the z global direction on the third surface of each element involved in the region with property id 5. It is assumed that a 3D time dependent problem is being solved where the mesh consists of linear tetrahedron elements and load is included to the first load case.

```
el_load propid 5 lc_id 1 load_type surface nsurf 4  ncomp 3
coord_sys 0
coord_sys 0
coord_sys 1 load_comp
funct_type stat const_val 0.0
funct_type stat const_val 0.0
funct_type stat const_val 6.0e3
coord_sys 0
```

The following command placed in the section **eledgpr** assigns uniform edge load 2+0.1t kN/m increasing linearly in time in the local x direction on the second edge of each adjacent element of the edge with property id 3. It is assumed that a 2D time dependent mechanical problem is being solved where the mesh consists of linear quadrilateral elements and load is included to the first load case.

```
el_load propid 3 lc_id 1 load_type edge nsurf 4  ncomp 2
coord_sys 0
coord_sys 2
load_comp
funct_type pars 2.0e3+0.1e3*t
funct_type stat const_val 0.0
coord_sys 0
coord_sys 0
```

The following command placed in the section elvolpr assigns linearly increasing volume load from zero up to 18 kN/m^3 at time 100 s and then is kept constant until 200 s. The load is defined in the z global direction for each element involved in the region with property id 1. It is assumed that a 3D linear statics problem is being solved and load is included to the first load case.

```
el_load propid 1 lc_id 1 load_type volume ncomp 3
load_comp
funct_type stat const_val 0.0
funct_type stat const_val 0.0
funct_type tab
approx_type linear
ntab_items 3
  0.0   0.0
100.0  18.0e3
200.0  18.0e3
```

9.7.9 `edge_tdload` command

The command assigns time dependent edge load to element edges with the given edge property id. It must be placed only in the `eledgpr` section otherwise it is ignored. The load components are prescribed with the help of general time dependent functions and for each particular component, the different function has to be specified. The load components may be either constant or the intensity of the load components may vary with respect to spatial coordinates and time according to the `gfunct` specified. The command has the following syntax:

`edge_load` **`propid`** `prop` **`lc_id`** `nlc` **`ncomp`** `nc`
`coord_sys` `lcs` **`load_comp`** {val$_i$}×nc

where the parameters have the following meaning:

`prop` - the property id of the edge where the load will be applied to (%ld).

`nlc` - load case id which the load will be involved into (%ld).

`nc` - the number of edge load components (%ld).

`lcs` - specifier of a coordinate system on element edges in which the edge load components will be given. There are available the following options {`1 | 2`} where 1 means edge load components given in the global coordinate system and 2 means edge load components given in the local coordinate system of the given element edge.

gf$_i$ - the `gfunct` record of i-th volume load component in [N/m]. The function type may be one of the following options {`stat | pars | tab | pars_set` }. If function type is {`pars | pars_set` } then the expression is used for calculation of load intensity at nodes of the given region/volume according to their spatial coordinates which must be denoted by x, y or z in the expression string and t is considered as time coordinate. If function type is {`tab`} then time dependence is considered only. See section 7 for more details about `gfunct` record.

The load is merged in the case of multiple assignment for the same element and the new values of load are added to the previous ones. If there is conflict in the direction of the load component (global x local coordinate systems) the error is reported otherwise a message about the successful merging is written to the log file. The command is intended for use in forced dynamics, time dependent and growing mechanical problems and it should be noted that the command cannot be used in one preprocessor file with `el_load, edge_load, surface_load and volume_load` commands where the subloadcase concept is used. See 9.3 for more details about different load case concepts.

The following command placed in the section `eledgpr` assigns constant uniform edge load 4 kN/m in the z global direction on each element edge which is involved in edge with property id 2. It is assumed that a 2D plane stress time dependent problem is being solved and load is included to the first load case.

```
edge_load propid 2 lc_id 1 ncomp 2
coord_sys 1
load_comp
funct_type stat const_val 0.0
funct_type stat const_val 4.0e3
```

The following command placed in the section `eledgpr` assigns edge load with linear course along the edge with property id 3 which is parallel with the global x axis. The edge begins at node with x=0.0 m where the load intensity will be 2 kN/m while the end of the edge is at node with x=4.0 m and there is load intensity 14 kN/m. The load intensity increases linearly on time starting from zero. It is assumed that a 2D plane stress time dependent problem is being solved where time stepping starts from 0 s, the load acts in the z global direction and load is included to the first load case.

```
edge_load propid 3 lc_id 1 ncomp 2
coord_sys 1
load_comp
funct_type stat const_val 0.0
funct_type pars (3.0e3*x+2.0e3)*t
```

9.7.10 `surf_tdload` command

The command assigns surface time dependent load to element surfaces with the given surface property id. It must be placed only in the `elsurfpr` section otherwise it is ignored. The load components are prescribed with the help of general time dependent functions and for each particular component, the different function has to be specified. The load components may be either constant or the intensity of the load components may vary with respect to spatial coordinates and time according to the `gfunct` specified. The command has the following syntax:

surf_load propid prop **lc_id** nlc **ncomp** nc
coord_sys lcs **load_comp** $\{gf_i\}\times$nc

where the parameters have the following meaning:

`prop` - the property id of the surface where the load will be applied to (%ld).

`nlc` - load case id which the load will be involved into (%ld).

`nc` - the number of surface load components (%ld).

`ft` - type of definition of load component values. There are available the following options {`stat | pars`} or corresponding integer values {`0 | 1`} where `stat` means constant load components and `pars` means components defined with help of expression string.

`lcs` - specifier of a coordinate system on element surfaces in which the surface load components will be given. There are available the following options {`1 | 2`} where 1 means surface load components given in the global coordinate system and 2 means surface load components given in the local coordinate system of the given element surface.

$\mathtt{gf}_i$ - the `gfunct` record of i-th volume load component in [N/m^2]. The function type may be one of the following options {`stat | pars | tab | pars_set` }. If function type is {`pars | pars_set` } then the expression is used for calculation of load intensity at nodes of the given region/volume according to their spatial coordinates which must be denoted by x, y or z in the expression string and t is considered as time coordinate. If function type is {`tab`} then time dependence is considered only. See section 7 for more details about `gfunct` record.

The load is merged in the case of multiple assignment for the same element and the new values of load are added to the previous ones. If there is conflict in the direction of the load component (global x local coordinate systems) the error is reported otherwise a message about the successful merging is written to the log file. The command is intended for use in forced dynamics, time dependent and growing mechanical problems and it should be noted that the command cannot be used in one preprocessor file with `el_load, edge_load, surface_load and volume_load` commands where the subloadcase concept is used. See 9.3 for more details about different load case concepts.

The following command placed in the section `elsurfpr` assigns constant uniform surface load 1.5 kN/m^2 in the x global direction on each element surface which is involved in surface with property id 2. It is assumed that a 3D time dependent problem is being solved and load is included to the first load case.

```
surf_load propid 2 lc_id 1 ncomp 3
coord_sys 1
load_comp
funct_type stat const_val 1.5e3
funct_type stat const_val 0.0
funct_type stat const_val 0.0
```

The following command placed in the section `elsurfpr` assigns surface load with linear course across the surface with property id 4 which is parallel with the global xy plane. The load intensity decreases linearly until zero values are attained at time 3600 s. It is assumed that a 3D time dependent problem is being solved, the load acts in the z global direction and load is included to the first first load case.

```
surf_load propid 4 lc_id 1 ncomp 3
coord_sys 1
load_comp
funct_type stat const_val 0.0
funct_type stat const_val 0.0
funct_type pars (6.0e3*x+3.5e3*y+2.0e3)*(1.0-t/3.6e3)
```

9.7.11 `volume_tdload` command

The command assigns time dependent volume load to elements involved in the region/volume with the given volume property id. It must be placed only in the `elvolpr` section otherwise it is ignored. The load components may be either constant or the intensity of the load components may vary with respect to spatial coordinates and time according to the expression specified. The load components are prescribed with the help of general time dependent functions and for each particular component, the different function has to be specified. The command has the following syntax:

volume_load propid prop **lc_id** nlc **ncomp** nc
load_comp {gf$_i$}×nc

where the parameters have the following meaning:

`prop` - the property id of the region/volume where the load will be applied to (%ld).

`nlc` - load case id which the load will be involved into (%ld).

`nc` - the number of volume load components (%ld).

`gf`$_i$ - the `gfunct` record of i-th volume load component in [N/m^3]. The function type may be one of the following options {`stat | pars | tab | pars_set` }. If function type is {`pars | pars_set` } then the expression is used for calculation of load intensity at nodes of the given region/volume according to their spatial coordinates which must be denoted by x, y or z in the expression string and t is considered as time coordinate. If function type is {`tab`} then time dependence is considered only. See section 7 for more details about `gfunct` record.

The load is merged in the case of multiple assignment for the same element and the new values of load are added to the previous ones. A message about the successful merging is written to the log file. The command is intended for use in forced dynamics, time dependent and growing mechanical problems and it should be noted that the command cannot be used in one preprocessor file with `el_load, edge_load, surface_load and volume_load` commands where the subloadcase concept is used. See 9.3 for more details about different load case concepts.

The following command placed in the section `elvolpr` assigns constant volume load 23 kN/m^3 in the z global direction on each element which is involved in region with property id 2. It is assumed that a 3D time dependent problem is being solved and load is included to the first load case.

```
volume_load propid 2 lc_id 1 ncomp 3
load_comp
funct_type stat const_val 0.0
funct_type stat const_val 0.0
funct_type stat const_val 2.3e4
```

The following command placed in the section `elvolpr` assigns volume load, with proportional change of intensity with respect to the x axis, to all elements involved in region with property id 4. The load intensity also varies in time according to $\sin(0.5t)$ function. It is assumed that a 3D time dependent problem is being solved, the load acts in the z global direction and load is included to the first load case.

```
volume_load propid 4 lc_id 1 ncomp 3
load_comp
funct_type stat const_val 0.0
funct_type stat const_val 0.0
funct_type pars (1.5e4+0.3e3*x)*sin(0.5*t)
```

9.7.12 `el_eigstr` command

The command assigns eigenstrains/eigenstresses to elements involved in the entity with the given property id. The eigenstrain/eigenstress components are defined with help of time dependent functions. The command has the following syntax:

el_eigstr propid prop **str_type** strt **ncomp** nc **eigstr_comp** {es$_i$}×nc

where the parameters have the following meaning:

`prop` - the property id of the given entity (%ld).

`strt` - specifier of type of applied eigen quantity, i.e. strain or stress, according to Tab 9.2 (keyword or %ld)

`nc` - the number of eigenstrain/eigenstress components (%ld).

`es`$_i$ - i-th eigenstrain/eigenstress component defined by record of a time function that has to return the component value with respect to actual time and space coordinates of the given integration point. See chapter 7 for more details about the format of time function record.

Quantity type keyword	Quantity type id	Description
`strain`	0	eigenstrains will be assumed
`stress`	1	eigenstresses will be assumed

Table 9.2: Table of eigen qunatity types

Should be noted that only one type of eigen quantity can be defined in the problem either eigenstrains or eigenstresses. Zero eigenstrains/eigenstresses are assigned to elements that are not involved by any of the `el_eigstr` commands. In the case of multiple assignment of different eigenstrain/eigenstress component to the same element, the error is reported otherwise a message about the multiple assignment of the same value is written to the log file.

The following command placed in the section `elvolpr` assigns constant eigenstrain vector $\boldsymbol{\varepsilon}_0$=$\{0.0;\ 2.5\cdot 10^{-5};\ -3.1\cdot 10^{-6};\ 0.0\}^T$ to elements involved in region with property id 0. It is assumed that a 2D time dependent plane strain problem is being solved.

```
el_eigstr propid 0  str_type strain ncomp 4 eigstr_comp
funct_type stat  const_val  0.0    # eps_x
funct_type stat  const_val  2.5e-5 # eps_y
funct_type stat  const_val -3.1e-6 # gamma_xy
funct_type stat  const_val  0.0    # eps_z
```

The following command placed in the section `elvolpr` assigns variable eigenstress vector $\boldsymbol{\sigma}_0$ according to K_0 procedure to the retangular block of soil with height 40 m. In this case, the dead weight of soil is assumed to be γ=20 kN/m^3 which generates vertical stress component $\sigma_y = -\gamma h = \gamma(y-40)$. Remaining normal stress components are assumed to be 1/3 of the vertical one. This variable eigenstress vector is assigned to elements involved in region with property id 1. It is assumed that a 2D time dependent plane strain problem is being solved.

```
el_eigstr propid 1  str_type stress ncomp 4 eigstr_comp
funct_type pars  func_formula 20000.0/3.0*(y-40.0) # sig_x
funct_type pars  func_formula 20000.0*(y-40.0)     # sig_y
funct_type pars  func_formula 0.0                  # tau_xy
funct_type pars  func_formula 20000.0/3.0*(y-40.0) # sig_z
```

9.7.13 `el_tfunc` command

This command prescribes element time function in case of growing mechanical problems. These time functions controls the element addition and withdrawing. If the element is added to the problem then its status is 'switched on' and it is taken into account at the given time while the opposite element state 'switched off' withdraw the element from the problem. Thus the time function controls the element birth and death. The time function must return 0 if the elements are required to be withdrawn (switched off) and it must return 1 if the elements are required to be added to the problem (switched on). All these time functions must be defined in the section `gfunct` and referenced by their identifiers. The command syntax follows:

`el_tfunc propid` `prop` **`tfunc_id`** `id`

where the parameters have the following meaning:

- `prop` - the property id of the given entity (%ld).
- `id` - identifier of time function from the section `gfunct` (%ld). The values range from 1 to `ngf` where `ngf` is defined in the section `gfunct`.

Should be noted that nodal DOFs are defined to be free if they are connected to elements whose status is 'switched on'. This default behaviour of nodes can be overriden by nodal command `nod_tfunct`, see section 9.6.4. In the case of multiple assignment of different time functions to the same element, the error is reported otherwise a message about the multiple assignment of the same time function is written to the log file.

The following command placed in the `elvolpr` section assigns the second time function from the section `gfunct` which switches on elements involved in region with property id

5 for period starting at 24000 s and lasting until the end of analysis. It is supposed that growing mechanical problem would be solved in this case.

```
begsec_elvolpr
.
.
el_tfunc propid 5  tfunc_id 2
.
.
endsec_elvolpr
.
.
begsec_gfunct
time_functions
num_gfunct 5
gf_id 1 funct_type itab
nitab_items 2
0.0    0
5.6e4  1
gf_id 2 funct_type itab
nitab_items 2
0.0    0
2.4e4  1
.
.
endsec_gfunct
```

9.8 Section outdrv

The section beginning is marked with the keyword begsec_outdrv and the section is closed by the keyword endsec_outdrv. It contains the description of the detailed setup of the result output. There are three types of the output file produced by MEFEL. Thirst type is represented by plain text file, the second one is represented by files in formats supported by graphic postprocessors (GiD, VTK, OpenDX, etc.) and the last one is text file with tabular output of selected quantities that is intended for the creation of diagrams (XMGrace, GNUPLot, MS Excel, etc.). All these parameters of the result output are controlled by classes outdriverm and outdiagm in MEFEL. The class contains function read which is being called for the processing of this section. Contrary to MEFEL, the keywords usage is switched on in this case. The format of this section is quite complex and it is described in details in [10].

9.9 Section gfunct

The section beginning is marked with the keyword begsec_gfunct and the section is closed by the keyword endsec_gfunct. The section gfunct is compulsory only for growing mechanical problems and it contains a list of definitions of time functions that control nodal DOFs and element addition and withdrawing. In case of elements, the time function must return 0 if the elements are required to be withdrawn (switched off) and it must return 1 if the elements are required to be added to the problem (switched on). Should be noted that nodal DOFs are defined to be free if they are connected to elements whose status is 'switched on'. This default behaviour of nodes can be overridden by the nodal command `nod_tfunct` (see section 9.6.4) in which DOFs at particular nodes are controlled by time functions which return 0 if the given DOFs should not involved in the problem solved, i.e. they are constrained or 1 if the given DOFs are free. The positive integer value greater than 1 should be returned if the given DOFs with the same time function value are coupled. The negative integer value should be returned if there are nonzero prescribed displacements at given time. In such the case, the function value represents the the negative value of index k of prescribed values $\mathtt{val}_k$ defined in section `loadcase`.

The format of the section is given as follows:

time_functions
num_gfunct ngf
{**gf_id** $\mathtt{id}_i$ $\mathtt{gfrec}_i$}×ngf

where the parameters have the following meaning:

`ngf` - the total number of time functions defined (%ld).

$\mathtt{id}_i$ - identifier of i-th time function (%ld).

`gfrec` - time function record.

Generally, the time function record `gfrec` should be in format of general function record but in this case only `itab` type of general function is accepted which is represented by a table containing times and corresponding integer values returned by time function. The format of `gfrec` follows:

funct_type itab
nitab_items nit
{$\mathtt{t}_i$ $\mathtt{val}_i$}×nit

where the parameters have the following meaning:

`nit` - the number of intervals on which the time function will be defined (%ld).

$\mathtt{t}_i$ - initial time of i-th interval (%le).

$\mathtt{val}_i$ - the value returned by function for time ranging in the i-th interval (%ld).

Should be noted that i-th time interval is defined as half-closed $[\mathtt{t}_i,\mathtt{t}_{i+1})$ and on that interval the function returns $\mathtt{val}_i$. If time argument is less than t_1 then $\mathtt{val}_1$ is being returned. If time argument is greater than $\mathtt{t}_{nit}$ then $\mathtt{val}_{nit}$ is being returned.

The following example defines two time functions for elements and two for nodes. The first function switches on elements at interval ranging from 5600 s to the end of computation and the second function switches on elements at interval ranging from 24000 s to 51000 s. The third function defines fixed nodal DOF for whole time of computation and the fourth function defines nodal DOF coupling group 2 at interval ranging from 10000 s to 30000 s and in the remaining time, the nodal DOF is fixed.

```
begsec_gfunct
time_functions
num_gfunct 4
gf_id 1 funct_type itab
nitab_items 2
0.0    0
5.6e3  1
gf_id 2 funct_type itab
nitab_items 3
0.0    0
2.4e4  1
5.1e4  0
gf_id 3 funct_type itab
nitab_items 1
0.0    0
gf_id 4 funct_type itab
nitab_items 3
0.0    0
1.0e4  2
3.0e4  0
endsec_gfunct
```

Part II

MECHPREP - Examples

This part contains several examples of mechanical problems which can be defined with help of MECHPREP. In chapters 10 and 11, mechanical problem of cantilever beam is solved in 2D and 3D where various load types are defined. In chapters 12 and 13, two nonlinear statics problems in 2D are solved with help of the Newton-Raphson method and arclength method. Examples described in this chapter can be found on [12] where the user can download one zip file with corresponding MECHPREP files. The listings of files are also involved in the text of individual sections but they are divided into several parts and inset with comments and description of these parts.

Chapter 10

Linear statics problem in 2D

This section describes how to prepare MEFEL input file with help of MECHPREP for the linear statics problem of cantilever beam modelled in 2D. The beam is subjected to four load cases:

1. Dead weight load f_1=24 kN/m^3 which is represented by volume load on elements calculated approximately from the given material density ρ=2400 kg/m^3.
2. Top edge is loaded by continuous load with linear distribution with zero value on the free end of beam and maximum value f_2=30 kN/m at fixed end of the beam.
3. Vertical force F_3=15 kN applied at the free end of the beam.
4. Beam is subjected to the uniform temperature change ΔT_4=20 °C.

The cantilever beam has length 5 m and rectangular cross section 0.3×0.5 m. Material of the beam is assumed to be elastic isotropic one where Young's modulus E=25 GPa, Poisson's ratio ν=0.25 and the thermal expansion coefficient $\alpha=12\times10^{-6}$ K^{-1}. The settings of the example is depicted in Fig. 10.1.

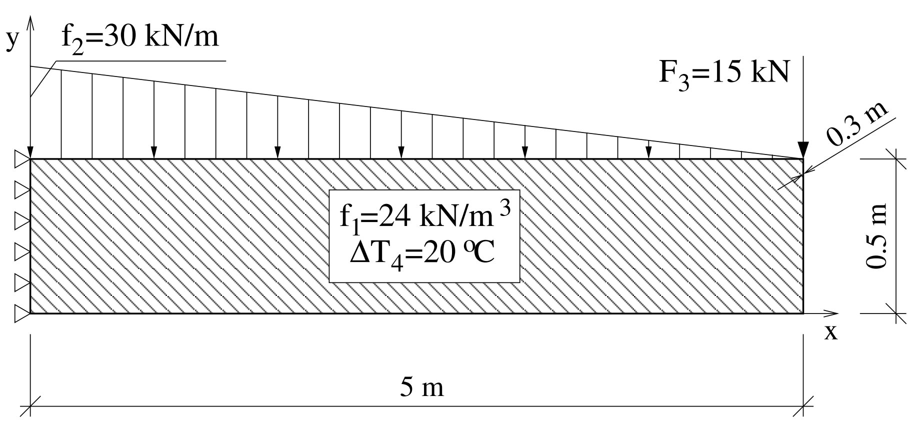


Figure 10.1: Settings of cantilever beam example in 2D

10.1 Topology file

In the first step, a mesh file must be created and corresponding property identifiers must be defined in order to MECHPREP can work. The mesh file can be created either manually or it can be generated simply with help of rectangular mesh generator gensifquad (see section 4.2.6). Run the following command:

```
gensifquad cantilever2d.top 5.0 0.5 50 15 1
```

and if everything run well the mesh file `cantilever2d.top` will be created including all property identifiers. The mesh file can be displayed with help of MeshEditor tool and property identifiers of edges and vertices can be visualized by various colors similarly as in Fig 10.2.

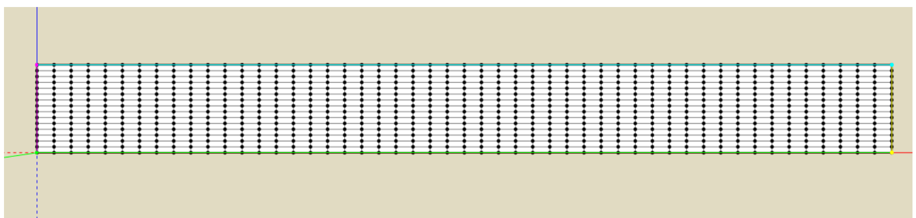


Figure 10.2: Cantilever beam - generated mesh with visualized property identifiers

10.2 Preprocessor file - section `files`

Having the mesh file with property identifiers created, a preprocessor file should be prepared. Preprocessor file is composed from the several sections that may be organized arbitrarily in the file but they will be introduced in the order of the real processing in MECHPREP. Each preprocessor file must contain section `files` which in this case has the following contents:

```
1 begsec_files
2 cantilever2d.top
3 mesh_format    sifel
4 edge_numbering 1
5 endsec_files
```

where the topology input file is being specified to be `cantilever2d.top` in line 2 then the format of the topology file is given in line 3 and finally, the it is given that edge and surface property identifiers are given also on elements - line 4. The command in line 4 is in accordance with setup of mesh generator `gensifquad` (the last argument on the command line) which cause the write of edge property numbers on elements to the topology file. See section 4.2 for more details about the SIFEL mesh format.

It is supposed that materials and cross sections will be given in the preprocessor file directly and thus no file names of material and cross section files are given. More details about this preprocessor section can be found in 9.1.

10.3 Preprocessor file - section probdesc

This section is used by MEFEL to identify the type of problem solved, the type of equation system solver and some other calculation setup. Some details about the particular cases of probdesc setup can be found in [10]. In this case, the following setup is chosen:

```
8  begsec_probdesc
9  Cantilever beam 5x0.5 m loaded by the various load
10 mespr 1
11 problemtype linear_statics
12
13 straincomp 1
14 strainpos  1
15 strainaver 0
16 stresscomp 1
17 stresspos  1
18 stressaver 0
19 othercomp  0
20 reactcomp  1
21
22 adaptivity     0
23 stochasticcalc 0
24 homogenization 0
25 noderenumber   0
26
27 stiffmatstor   skyline_matrix
28 typelinsol     ldl
29 endsec_probdesc
```

In this section, the title of the problem solved is given in line 9, detailed message printing is switched on (line 10) and linear statics problem type is specified to be solved in line 11.

In the sequential two blocks of commands, the strain calculation is on (line 13) and strains will be calculated at integration points (line 14) and therefore no averaging of strains is necessary (line 15). The same setup for stress computation is defined in lines 16–18. The problem is linear statics and in such case, the constitutive model is assumed to be elasticity where no internal variables are defined and thus the other values will not be calculated (line 19). Finally, the calculation of reactions is required in line 20. All additional advanced techniques such as mesh adaptivity (line 22), stochastic calculations (line 23), homogenization techniques (line 24) and node renumbering (line 25) are not taken into account in the computation.

The last block of commands defines the type of solver of system of linear algebraic equations. The line 27 defines that the skyline storage of system matrix will be used and system will be solved with help of LDL decomposition method (line 28) which can be applied in this case because the system matrix is symmetric and positive definite.

10.4 Preprocessor file - section `loadcase`

This section defines the number of load cases and some details about the content of particular load cases. According to the problem setting, this section looks as follows:

```
32 begsec_loadcase
33 num_loadcases      4
34 #temperature load type for the first load case
35 lc_id  1
36 temp_load_type    0
37 #temperature load type for the second load case
38 lc_id  2
39 temp_load_type    0
40 #temperature load type for the third load case
41 lc_id  3
42 temp_load_type    0
43 #temperature load type for the fourth load case
44 lc_id  4
45 temp_load_type    1
46 endsec_loadcase
```

where four individual load cases are established in line 33, and for each load case, the type of temperature load load is defined with help of command pairs `tempr_type_lc_id` and `temp_load_type`. The first three load cases are composed just from force load, i.e. no temperature load is defined by in lines 36, 39 and 42 while the last one contains this load and thus `temp_load_type` is set to 1 in line 45.

10.5 Preprocessor file - section `mater`

This section contains the list of material models and their parameters. In this example, the distribution of material properties is assumed to be homogeneous and linear elastic and therefore only one linear isotropic elastic material have to be defined. The section has the following content:

```
49 begsec_mater
50 num_mat_types 2
51 mattype elisomat   num_inst 1
52 1 25.0e9 0.25
53 mattype therisodilat num_inst 1
54 1 1.2e-5
55 endsec_mater
```

In the section, two material types (elastic isotropic and thermal dilatancy) are being defined (line 50) and sequential lines contains the specification of these material types. The line 51 defines that the material type is elastic isotropic with one instance of material parameter set. Line 52 defines the first instance of material parameter set of elastic isotropic material which requires two parameters - Young's modulus (25 GPa) and Poisson's ratio (0.25). The second material model is represented by isotropic thermal dilatancy with one instance of parameter set (line 53). The parameter set is defined on the last line where the thermal expansion coefficient α is defined to be $12{\cdot}10^{-6}$ K^{-1}.

10.6 Preprocessor file - section `crsec`

This section contains the list of cross sections and their parameters. In this example, the cross section is given by the thickness 0.3 m which is uniform across the beam. The section has the following content:

```
58 begsec_crsec
59 num_crsec_types    1
60 crstype  csplanestr   num_inst 1
61 1 0.3
62 endsec_crsec
```

In the section, only one cross section type (for plane elements) is being defined (line 59) and sequential lines contains the specification of this one cross section type. The line 60 defines that the cross section type is for plane stress/strain elements (`csplanestr`) with one instance of cross section parameter set. The last line 61 defines the first instance of cross section parameter set for plane elements which requires just one parameters - thickness (0.3 m).

10.7 Preprocessor file - number of nodal DOFs

The section `nodvolpr` defines common properties for group of nodes involved in the volume with specific property id. The most common use of this section is for the specification of number of DOFs at nodes. In this example, two DOFs are defined in all nodes of the mesh. The section content is listed below:

```
65 begsec_nodvolpr
66 # number of degrees of freedom for all nodes
67 ndofn 2 propid 1
```

```
69 endsec_nodvolpr
```

where the command `ndofn` in line 67 defines two DOFs at all nodes with region/volume property id 1, i.e. on the whole domain solved.

10.8 Preprocessor file - Dirichlet's boundary conditions

Dirichlet's boundary condition prescribes values of primary unknowns defined in the problem solved and they can be defined with help of `bocon` command. In this example, this type of boundary conditions is represented by fixed nodes on the left edge of the cantilever beam. This edge is marked by the property id 2 and therefore the `bocon` command should be placed in the `nodedgpr` section whose content is listed below

```
begsec_nodedgpr
# fixation of nodes on the left beam edge
bocon propid 2 num_bc 2 dir 1 cond 0.0  dir 2 cond 0.0
endsec_nodedgpr
```

where displacements are prescribed to be zero values for all DOFs (defined in the previous section) of nodes on the edge with property id 2.

10.9 Preprocessor file - nodal forces

The nodal forces can be applied at selected nodes with help of command `nod_load` which can be placed into arbitrary section relating with nodes. This example contains just one force 15 kN applied in the top corner node at free end of the beam. According to the setting, the force should be involved in the third load case. The mentioned node in the corner of the beam has got assigned vertex property 1 and therefore the command `nod_load` should be placed in the section `nodvertpr` whose content is listed below

```
78 begsec_nodvertpr
79 # nodal load by force 15 kN
80 nod_load propid 1 lc_id 3 load_comp 0.0 -15.0e3
81 endsec_nodvertpr
```

where line 80 represents the suitable record of the preprocessor command. Should be noted that both components of the applied force must be given in this command, i.e. horizontal component is zero while the vertical one is the 15 kN.

10.10 Preprocessor file - temperature load

The beam is also loaded by uniform change of temperature defined in the load case 4. The temperature change is defined with help of `nod_temper` and it must given at all nodes of the beam mesh which leads to placement of this command to the section `nodvolpr` because all nodes has got assigned the same volume property id 1 in the generator.

```
begsec_nodvolpr
```

```
nod_temper  propid 1 lc_id 4 temperature 20.0
endsec_nodvolpr
```

10.11 Preprocessor file - element type, material model and cross section

The FE type, material model and cross section are essential properties of elements which must be given in all 2D problems. In this example, all elements have the same FE type, material model and thickness and therefore the most simple way how to assign them to all elements is the use of corresponding commands in the element section `elvolpr` keeping in mind that the volume property id 1 is the same for all elements. The element type can be assigned by the command `el_type` while material model and cross section by the commands `el_mat` and `el_crsec` respectively.

```
84 begsec_elvolpr
85 el_type    propid 1  planeelementlq strastrestate planestress
86 el_mat     propid 1  num_mat 2 type elisomat      type_id 1
87                          type therisodilat  type_id 1
88 el_crsec   propid 1  type csplanestr type_id 1
```

```
91 endsec_elvolpr
```

where line 85 assigns plane quadrilateral element with linear shape functions and defines that plane-stress state will be assumed on these elements. The same material model of thermo-elasticity is assumed on all elements and assigned by the command in line 86. The model is composed from two independent parts (kewyword `num_mat`) one for elasticity (`elisomat` - line 86) and one for thermal dilatancy (`therisodilat` - line 87). Both models refers to the first instance of material parameter set with help of keywords `type_id`.

Definition of thickness in line 88 by the command `el_crsec` has the syntax similar to `el_mat` command where the type of cross section must be given with help of keyword `type` and then the the first instance of cross section parameter set is referenced with help of keyword `type_id`.

10.12 Preprocessor file - element load

In the first load case, the beam is loaded by dead weight load which must be applied to all elements in the mesh. It can be achieved by the command `volume_load` placed in the section `elvolpr` because all elements has got assigned the same volume property id 1. The syntax of the command is listed below

```
84 begsec_elvolpr
```

```
89 volume_load  propid 1  lc_id 1  ncomp 2
90              func_type stat coord_sys 1 load_comp 0.0 -24.0e3
91 endsec_elvolpr
```

where the line 89 defines volume load on all elements with volume property id 1, the load is applied in the load case 1 and two components of load will be given latter in line 90.

The command continues in line 90 where the load is defined to be constant (keyword `func_type`), applied in the global coordinate system (keyword `coord_sys`) and finally, two load components are given - the dead weight load is applied in the vertical direction.

Another type of load is represented by linear continuous load applied on the top edge of the beam in the second load case. It can be defined with help of command `edge_load` placed in the section `eledgpr` because the load should be applied on elements adjacent to the edge with property id 1 assigned by the generator. The content of the section is listed below

```
94 begsec_eledgpr
95 edge_load propid 1  lc_id   2  ncomp 2  func_type pars
96           coord_sys 1 load_comp 0.0 -30.0e3+6.0e3*x
97 endsec_eledgpr
```

where line 95 defines load on element edges that are adjacent to edge with property id 1 in load case 2. With respect to the element type and number of DOFs at nodes, two component of load must be given (keyword `ncomp`). The load has a linear course along the edge and therefore it must be defined with help of a parsed expression string where the appropriate function can be defined easily (keyword `func_type`). The command continues in line 96 which defines load components to be in global coordinate system (keyword `coord_sys`) and then particular load components are given after keyword `load_comp`. The horizontal component is zero while the vertical component is given by linear function $f_2(x) = -30\cdot 10^3 + 6\cdot 10^3 x$. Both components are assumed to be parsed string expressions but only the second is dependent on the spatial coordinate. Should be noted that parsed string expressions must not contain any whitespace character otherwise they would be broken into independent parts in places of whitespace characters and in better case, an error would be signalized or in the worse case, different expression would be evaluated tacitly.

10.13 Setup of the result output

The last section that has to be specified is represented by section `outdrv` where the output of results from MEFEL should be configured. More details about this section can be found in [10]. The section is composed from three parts dealing with different forms of result output. The first part controls output to the file in the text form has the following content:

```
begsec_outdrv
# Description of output to the text file
# ----------------------------------------
textout 1
# text output file name
cant2d.out
# text output at nodes
sel_nodstep   sel_all
sel_nodlc     sel_all
```

```
109 displ_nodes   sel_all displ_comp sel_all
110 strain_nodes  sel_no
111 stress_nodes  sel_no
112 other_nodes   sel_no
113 reactions     1
114 # text output at elements
115 sel_elemstep  sel_all
116 sel_elemlc    sel_all
117 strain_elems  sel_all elemstrain_comp sel_all
       elemstra_transfid 0
118 stress_elems  sel_all elemstress_comp sel_all
       elemstre_transfid 0
119 other_elems   sel_no
120 # text output at user defined points
121 sel_pointstep sel_no
```

```
146 endsec_outdrv
```

In this example, the text output of results is required (line 103) to the file `cant2d.out` (line 105). After that the time/load steps and load case numbers must be given at which the nodal results will be printed out. There are no time/load steps in linear statics problems (everything is calculated at once) and therefore if the user needs to print some nodal quantities then simply defines for keyword `nod_step` the value `sel_all` which results in selection of all time/load steps (line 107). Results from all load cases are required to print out in line 108 using the same value `sel_all` for the keyword `sel_nodlc`. Having the time/load steps and load cases specified, the print configuration of particular nodal quantities follows where for each quantity, the selection of nodes, where the given quantity will be printed out, is followed by the selection of the given quantity components. Thus line 109 specifies that for all nodes, all displacement components will be printed out while the line 110 selects no nodes (keyword value `sel_no`) for nodal strains, i.e. no nodal strains will be printed. The same option is specified for nodal stresses (line 111), and nodal `other` values (line 112) and thus they do not be printed too. Configuration in line 113 enables the reaction output.

Configuration of nodal values output is followed by the similar configuration of element values output performed in all integration points on the selected elements. It starts with selection of time/load steps (line 115) and load cases (line 116). Using the same keyword values as for nodes results to the selection of all time/load steps and all load cases. Line 117 specifies that for all elements (keyword `strain_elems`), the output of all strain components (keyword `elemstrain_comp`) will be performed with no transformation of components (keyword `elemstra_transfid`). Line 118 specifies that for all elements (keyword `stress_elems`), the output of all stress components (keyword `elemstress_comp`) will be performed with no transformation of components (keyword `elemstre_transfid`). Line 119 defines that no `other` values will be printed out. The last item of text output configuration is given in line 121 where output values at user defined points on elements can be specified but it has not been not fully implemented so no time steps are selected in this case.

Should be noted that the problem is linear statics and therefore `other` values must not be required to be printed out because the material model is linear elastic and it defines zero number of internal variables which indicates that `other` arrays on integration points are not allocated and required output of these values would lead to segmentation fault errors.

The second part controls output in the various formats used in graphic postprocessor tools. In this example, the GiD format will be required which allows for the most advanced configuration of the output. The part configuring this output is listed below:

```
100 begsec_outdrv
```

```
123 # Description of output to the graphics file in GiD format
124 #----------------------------------------------------------
125 outgr_format  grfmt_gid
126 # graphics output file name without extension
127 cant2d
128 # setup for nodal values
129 sel_nodstep  sel_all
130 sel_nodlc    sel_all
131 displ_nodes  sel_all   displ_comp   sel_all
132 strain_nodes sel_no
133 stress_nodes sel_no
134 other_nodes  sel_no
135 force_nodes  sel_all   force_comp   sel_all
136 # setup for element values
137 sel_elemstep sel_all
138 sel_elemlc   sel_all
139 strain_elems sel_all   elemstrain_comp sel_mtx
      elemstra_transfid 0
140 stress_elems sel_all   elemstress_comp sel_mtx
      elemstre_transfid 0
141 other_elems  sel_no
```

```
146 endsec_outdrv
```

Line 125 defines the format used for the result output with help of keyword `outgr_format` whose value is set to `gid`. This results into one GiD file with all result quantities (`cant2d.res`) that will be specified later in this part and another file with the mesh description (`cant2d.msh`). The common GiD file name is given in line 126 to which the corresponding suffix will be added automatically. Lines 128–141 contains the configuration of the output which uses the same keywords as in the previous part with only several differences described in the following text. The output of reactions is generally involved in the configuration of nodal forces output (line 135) where for all nodes (keyword `force_nodes`), all force components (keyword force_comp) will be printed out which results in output of nodal load components as well as reactions. There is also used different selection of strain and stress components on elements where `sel_mtx` optional value is used (lines 139, 140). This selection type provides the output of strains and stresses in the tensorial form (all their components) which allows for better postprocessing in the GiD (calculation of principal values and vectors).

The last part controls output of selected quantities in particular time/load steps which can be used for creation of diagrams which cannot be used in the case of linear statics problems and therefore the end of `outdrv` section has the following content:

```
100 begsec_outdrv
```

```
143 # Text output of diagrams
144 # -----------------------
145 numdiag 0
146 endsec_outdrv
```

where the line 145 defines that the number of diagram files created is zero.

10.14 Preprocessor file

This section contains listing of the whole preprocessor file.

```
begsec_files
cantilever2d.top
mesh_format    sifel
edge_numbering 1
endsec_files


begsec_probdesc
Cantilever beam 5x0.5 m loaded by the various load
mespr 1
problemtype linear_statics

straincomp 1
strainpos  1
strainaver 0
stresscomp 1
stresspos  1
stressaver 0
othercomp  0
reactcomp  1

adaptivity     0
stochasticcalc 0
homogenization 0
noderenumber   0
```

```
stiffmatstor   skyline_matrix
typelinsol     ldl
endsec_probdesc


begsec_loadcase
num_loadcases      4
#temperature load type for the first load case
lc_id  1
temp_load_type    0
#temperature load type for the second load case
lc_id  2
temp_load_type    0
#temperature load type for the third load case
lc_id  3
temp_load_type    0
#temperature load type for the fourth load case
lc_id  4
temp_load_type    1
endsec_loadcase


begsec_mater
num_mat_types 2
mattype elisomat   num_inst 1
1 25.0e9 0.25
mattype therisodilat num_inst 1
1 1.2e-5
endsec_mater


begsec_crsec
num_crsec_types    1
crstype  csplanestr   num_inst 1
1 0.3
endsec_crsec


begsec_nodvolpr
# number of degrees of freedom for all nodes
ndofn 2 propid 1
nod_temper  propid 1 lc_id 4 temperature 20.0
endsec_nodvolpr
```

```
begsec_nodedgpr
# fixation of nodes on the left beam edge
bocon propid 2 num_bc 2 dir 1 cond 0.0  dir 2 cond 0.0
endsec_nodedgpr


begsec_nodvertpr
# nodal load by force 15 kN
nod_load propid 1 lc_id 3 load_comp 0.0 -15.0e3
endsec_nodvertpr


begsec_elvolpr
el_type     propid 1  planeelementlq strastrestate planestress
el_mat      propid 1  num_mat 2 type elisomat      type_id 1
                                type therisodilat type_id 1
el_crsec    propid 1  type csplanestr type_id 1
volume_load  propid 1  lc_id 1  ncomp 2
             func_type stat coord_sys 1 load_comp 0.0 -24.0e3
endsec_elvolpr


begsec_eledgpr
edge_load propid 1  lc_id   2  ncomp 2  func_type pars
          coord_sys 1 load_comp 0.0 -30.0e3+6.0e3*x
endsec_eledgpr


begsec_outdrv
# Description of output to the text file
# ----------------------------------------
textout 1
# text output file name
cant2d.out
# text output at nodes
sel_nodstep   sel_all
sel_nodlc     sel_all
displ_nodes   sel_all displ_comp sel_all
strain_nodes  sel_no
stress_nodes  sel_no
other_nodes   sel_no
reactions     1
# text output at elements
sel_elemstep  sel_all
```

```
sel_elemlc    sel_all
strain_elems  sel_all elemstrain_comp sel_all
   elemstra_transfid 0
stress_elems  sel_all elemstress_comp sel_all
   elemstre_transfid 0
other_elems   sel_no
# text output at user defined points
sel_pointstep sel_no

# Description of output to the graphics file in GiD format
#----------------------------------------------------------
outgr_format  grfmt_gid
# graphics output file name without extension
cant2d
# setup for nodal values
sel_nodstep  sel_all
sel_nodlc    sel_all
displ_nodes  sel_all   displ_comp   sel_all
strain_nodes sel_no
stress_nodes sel_no
other_nodes  sel_no
force_nodes  sel_all   force_comp   sel_all
# setup for element values
sel_elemstep sel_all
sel_elemlc   sel_all
strain_elems sel_all   elemstrain_comp sel_mtx
   elemstra_transfid 0
stress_elems sel_all   elemstress_comp sel_mtx
   elemstre_transfid 0
other_elems  sel_no

# Text output of diagrams
# -----------------------
numdiag 0
endsec_outdrv
```

Chapter 11

Linear statics problem in 3D

This section describes how to prepare MEFEL input file with help of MECHPREP for the linear statics problem of cantilever beam modelled in 3D. The beam is subjected to four load cases:

1. Dead weight load f_1=24 kN/m^3 which is represented by volume load on elements calculated approximately from the given material density ρ=2400 kg/m^3.
2. Top surface is loaded by continuous load with linear distribution along the beam axis and uniform distribution in the y direction where zero value is on the free end of beam and maximum value f_2=30 kN/m at fixed end of the beam.
3. Vertical displacement w_3=8 mm is prescribed at the free end of the beam.
4. Beam is subjected to the uniform temperature change ΔT_4=20 °C.

The cantilever beam has length 5 m and rectangular cross section 0.3×0.5 m. Material of the beam is assumed to be elastic isotropic one where Young's modulus E=25 GPa, Poisson's ratio ν=0.25 and the thermal expansion coefficient $\alpha=12\times10^{-6}$ K^{-1}. The settings of the example is depicted in Fig. 11.1.

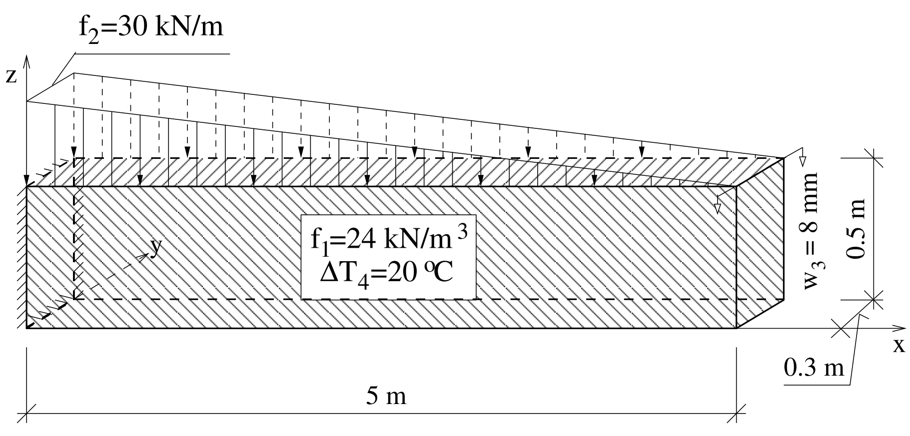


Figure 11.1: Settings of cantilever beam example in 2D

11.1 Topology file

In the first step, a mesh file must be created and corresponding property identifiers must be defined in order to MECHPREP can work. The mesh file can be created either manually or it can be generated simply with help of prism mesh generator `gensifhex` which can be found in folder SIFEL/PREP/SEQMESHGEN. Run the following command:

```
gensifhex cantilever3d.top  5.0 0.3 0.5  50 10 10  1
```

and if everything run well the mesh file `cantilever3d.top` will be created including all property identifiers. The property identifiers are generated according to Fig. 11.2 where V*i*, E*i*, S*i* and R*i* denotes property identifier *i* of vertex, edge, surface and region, respectively. The mesh file can be displayed with help of MeshEditor tool and property

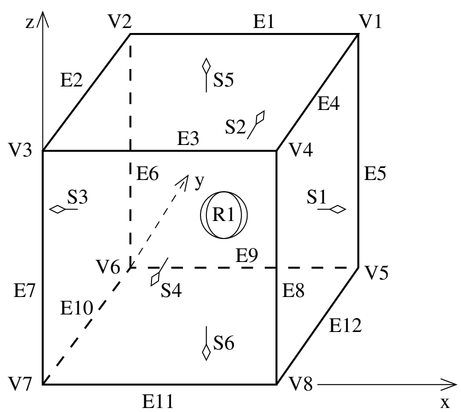


Figure 11.2: Property identifiers generated by `gensifhex` on a prism domain

identifiers of edges and vertices can be visualized by various colors similarly as in Fig 11.3.

11.2 Preprocessor file - section `files`

Having the mesh file with property identifiers created, a preprocessor file should be prepared. Preprocessor file is composed from the several sections that may be organized arbitrarily in the file but they will be introduced in the order of the real processing in MECHPREP. Each preprocessor file must contain section `files` which in this case has the following contents:

```
1 begsec_files
2 cantilever3d.top
3 mesh_format    sifel
4 edge_numbering 1
5 endsec_files
```

where the topology input file is being specified to be `cantilever3d.top` in line 2 then the format of the topology file is given in line 3 and finally, it is given that edge and surface

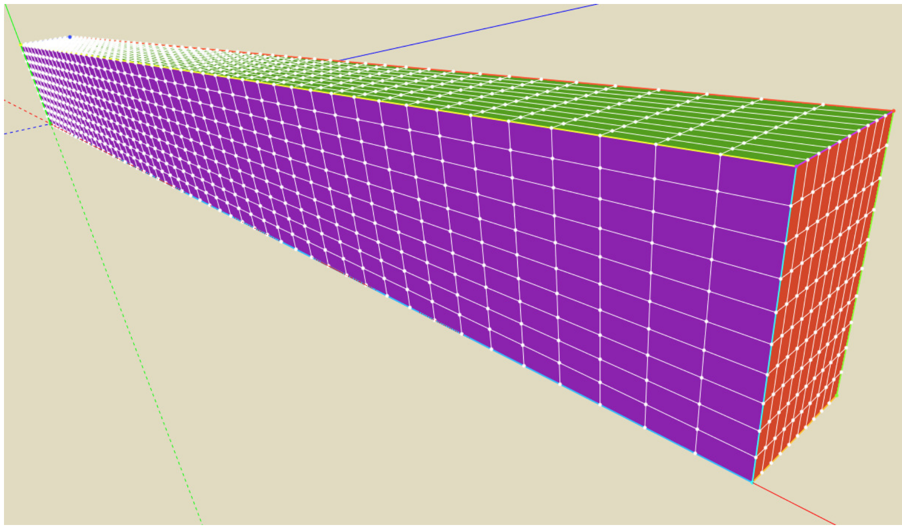


Figure 11.3: Cantilever beam - generated mesh with visualized property identifiers

property identifiers are given also on elements - line 4. The command in line 4 is in accordance with setup of mesh generator **gensifhex** (the last argument on the command line) which cause the write of edge property numbers on elements to the topology file. See section 4.2 for more details about the SIFEL mesh format.

It is supposed that materials and cross sections will be given in the preprocessor file directly and thus no file names of material and cross section files are given. More details about this preprocessor section can be found in 9.1.

11.3 Preprocessor file - section `probdesc`

This section is used by MEFEL to identify the type of problem solved, the type of equation system solver and some other calculation setup. Some details about the particular cases of `probdesc` setup can be found in [10]. In this case, the following setup is chosen:

```
begsec_probdesc
Cantilever beam 5x0.3x0.5 m loaded by the various load
mespr 1
problemtype linear_statics

straincomp 1
strainpos  1
strainaver 0
stresscomp 1
stresspos  1
stressaver 0
othercomp  0
reactcomp  1
```

```
21
22 adaptivity      0
23 stochasticcalc  0
24 homogenization  0
25 noderenumber    0
26
27 stiffmatstor    skyline_matrix
28 typelinsol      ldl
29 endsec_probdesc
```

In this section, the title of the problem solved is given in line 9, detailed message printing is switched on (line 10) and linear statics problem type is specified to be solved in line 11.

In the sequential two blocks of commands, the strain calculation is on (line 13) and strains will be calculated at integration points (line 14) and therefore no averaging of strains is necessary (line 15). The same setup for stress computation is defined in lines 16–18. The problem is linear statics and in such case, the constitutive model is assumed to be elasticity where no internal variables are defined and thus the other values will not be calculated (line 19). Finally, the calculation of reactions is required in line 20. All additional advanced techniques such as mesh adaptivity (line 22), stochastic calculations (line 23), homogenization techniques (line 24) and node renumbering (line 25) are not taken into account in the computation.

The last block of commands defines the type of solver of system of linear algebraic equations. The line 27 defines that the skyline storage of system matrix will be used and system will be solved with help of LDL decomposition method (line 28) which can be applied in this case because the system matrix is symmetric and positive definitive.

11.4 Preprocessor file - section `loadcase`

This section defines the number of load cases and some details about the content of particular load cases. According to the problem setting, this section looks as follows:

```
begsec_loadcase
num_loadcases      4
#temperature load type for the first load case
lc_id  1
temp_load_type    0
#temperature load type for the second load case
lc_id  2
temp_load_type    0
#temperature load type for the third load case
lc_id  3
temp_load_type    0
#temperature load type for the fourth load case
lc_id  4
temp_load_type    1
```

```
46 endsec_loadcase
```

where four individual load cases are established in line 33, and for each load case, the type of temperature load load is defined with help of command pairs `tempr_type_lc_id` and `temp_load_type`. The first three load cases are composed just from force load or prescribed displacements, i.e. no temperature load is defined by in lines 36, 39 and 42 while the last one contains this load and thus `temp_load_type` is set to 1 in line 45.

11.5 Preprocessor file - section `mater`

This section contains the list of material models and their parameters. In this example, the distribution of material properties is assumed to be homogeneous and linear elastic and therefore only one linear isotropic elastic material have to be defined. The section has the following content:

```
49 begsec_mater
50 num_mat_types 2
51 mattype elisomat   num_inst 1
52 1 25.0e9 0.25
53 mattype therisodilat num_inst 1
54 1 1.2e-5
55 endsec_mater
```

In the section, two material types (elastic isotropic and thermal dilatancy) are being defined (line 50) and sequential lines contains the specification of these material types. The line 51 defines that the material type is elastic isotropic with one instance of material parameter set. Line 52 defines the first instance of material parameter set of elastic isotropic material which requires two parameters - Young's modulus (25 GPa) and Poisson's ratio (0.25). The second material model is represented by isotropic thermal dilatancy with one instance of parameter set (line 53). The parameter set is defined on the last line where the thermal expansion coefficient α is defined to be $12{\cdot}10^{-6}$ K^{-1}.

11.6 Preprocessor file - section `crsec`

This section contains the list of cross sections and their parameters. In this example, no cross sections are defined because the problem is assumed in 3D and all dimensions of the domain solved are given in the mesh. In such a case, the section has the following content:

```
58 begsec_crsec
59 num_crsec_types    0
60 endsec_crsec
```

In the section, no cross section type (for plane elements) is being defined (line 59) and therefore no list of parameters is further specified.

11.7 Preprocessor file - number of nodal DOFs

The section `nodvolpr` defines common properties for group of nodes involved in the volume with specific property id. The most common use of this section is for the specification of number of DOFs at nodes. In this example, two DOFs are defined in all nodes of the mesh. The section content is listed below:

```
63 begsec_nodvolpr
64 # number of degrees of freedom for all nodes
65 ndofn 3 propid 1
```

```
67 endsec_nodvolpr
```

where the command `ndofn` in line 65 defines two DOFs at all nodes with region/volume property id 1, i.e. on the whole domain solved.

11.8 Preprocessor file - Dirichlet's boundary conditions

Dirichlet's boundary condition prescribes values of primary unknowns defined in the problem solved and they can be defined with help of `bocon` command. In this example, this type of boundary conditions is represented by fixed nodes on the left surface of the cantilever beam. This surface is marked by the property id 3 and therefore the `bocon` command should be placed in the `nodsurfpr` section whose content is listed below

```
70 begsec_nodsurfpr
71 # fixation of nodes on the left surface of beam
72 bocon propid 3 num_bc 3 dir 1 cond 0.0
73                         dir 2 cond 0.0
74                         dir 3 cond 0.0
75 endsec_nodsurfpr
```

where displacements are prescribed to be zero values for all DOFs (defined in the previous section) of nodes on the surface with property id 3.

11.9 Preprocessor file - prescribed displacement

The prescribed displacements at selected nodes can be applied with help of command `bocon` which can be placed into arbitrary section relating with nodes. This example contains just vertical displacement 8 mm applied on the top left edge nodes at free end of the beam. According to the setting, this load type should be involved in the third load case. The mentioned nodes on edge has got assigned edge property 4 and therefore the command `bocon` should be placed in the section `nodedgpr` whose content is listed below

```
78 begsec_nodedgpr
79 # prescribed displacements at nodes on the top left edge
```

```
80 bocon propid 4 num_bc 1 dir 3 cond -8.0e-3 lc_id 3
81 endsec_nodedgpr
```

where line 80 represents the suitable record of the preprocessor command.

11.10 Preprocessor file - temperature load

The beam is also loaded by uniform change of temperature defined in the load case 4. The temperature change is defined with help of `nod_temper` and it must given at all nodes of the beam mesh which leads to placement of this command to the section `nodvolpr` because all nodes has got assigned the same volume property id 1 in the generator.

```
65 begsec_nodvolpr
```

```
66 nod_temper  propid 1 lc_id 4 temperature 20.0
67 endsec_nodvolpr
```

11.11 Preprocessor file - element type and material model

The FE type and material model are essential properties of elements which must be given in all 3D problems. In this example, all elements have the same FE type and material model and therefore the most simple way how to assign them to all elements is the use of corresponding commands in the element section `elvolpr` keeping in mind that the volume property id 1 is the same for all elements. The element type can be assigned by the command `el_type` while material model by the command `el_mat`.

```
84 begsec_elvolpr
85 el_type    propid 1  linearhex
86 el_mat     propid 1  num_mat 2 type elisomat     type_id 1
87                                type therisodilat type_id 1
```

```
91 endsec_elvolpr
```

where line 85 assigns brick element with linear shape functions. The same material model of thermo-elasticity is assumed on all elements and it is assigned by the command in lines 86 and 87. The model is composed from two independent parts (keyword `num_mat`) one for elasticity (`elisomat` - line 86) and one for thermal dilatancy (`therisodilat` - line 87). Both models refers to the first instance of material parameter set with help of keywords `type_id`.

11.12 Preprocessor file - element load

In the first load case, the beam is loaded by dead weight load which must be applied to all elements in the mesh. It can be achieved by the command `volume_load` placed in the section `elvolpr` because all elements has got assigned the same volume property id 1. The syntax of the command is listed below

```
84 begsec_elvolpr
```

```
88 volume_load  propid 1  lc_id 1  ncomp 3
89              func_type stat coord_sys 1
90              load_comp 0.0 0.0 -24.0e3
91 endsec_elvolpr
```

where the line 88 defines volume load on all elements with volume property id 1, the load is applied in the load case 1 and three components of load will be given latter in line 90. The command continues in line 89 where the load is defined to be constant (keyword `func_type`), applied in the global coordinate system (keyword `coord_sys`) and finally, three load components are given in line 90 - the dead weight load is applied in the vertical direction.

Another type of load is represented by linear continuous load applied on the top surface of the beam in the second load case. It can be defined with help of command `surf_load` placed in the section `elsurfpr` because the load should be applied on elements adjacent to the surface with property id 5 assigned by the generator. The content of the section is listed below

```
94 begsec_elsurfpr
95 surf_load propid 5  lc_id   2  ncomp 3  func_type pars
96           coord_sys 1 load_comp 0.0 0.0 -30.0e3+6.0e3*x
97 endsec_elsurfpr
```

where line 95 defines load on element surfaces that are adjacent to surface with property id 5 in load case 2. With respect to the element type and number of DOFs at nodes, three components of load must be given (keyword `ncomp`). The load has a linear course along the x axis and therefore it must be defined with help of a parsed expression string where the appropriate function can be defined easily (keyword `func_type`). The command continues in line 96 which defines load components to be in global coordinate system (keyword `coord_sys`) and then particular load components are given after keyword `load_comp`. The horizontal components are zero while the vertical component is given by linear function $f_2(x) = -30 \cdot 10^3 + 6 \cdot 10^3 x$. Both components are assumed to be parsed string expressions but only the third is dependent on the spatial coordinate. Should be noted that parsed string expressions must not contain any whitespace character otherwise they would be broken into independent parts in places of whitespace characters and in better case, an error would be signalized or in the worse case, different expression would be evaluated tacitly.

11.13 Setup of the result output

The last section that has to be specified is represented by section `outdrv` where the output of results from MEFEL should be configured. More details about this section can be found in [10]. The section is composed from three parts dealing with different forms of result output. The first part controls output to the file in the text form has the following content:

```
100 begsec_outdrv
101 # Description of output to the text file
102 # ---------------------------------------
103 textout 1
104 # text output file name
105 cant3d.out
106 # text output at nodes
107 sel_nodstep   sel_all
108 sel_nodlc     sel_all
109 displ_nodes   sel_all displ_comp sel_all
110 strain_nodes  sel_no
111 stress_nodes  sel_no
112 other_nodes   sel_no
113 reactions     1
114 # text output at elements
115 sel_elemstep  sel_all
116 sel_elemlc    sel_all
117 strain_elems  sel_all elemstrain_comp sel_all
       elemstra_transfid 0
118 stress_elems  sel_all elemstress_comp sel_all
       elemstre_transfid 0
119 other_elems   sel_no
120 # text output at user defined points
121 sel_pointstep sel_no
```

```
146 endsec_outdrv
```

In this example, the text output of results is required (line 103) to the file `cant3d.out` (line 105). After that the time/load steps and load case numbers must be given at which the nodal results will be printed out. There are no time/load steps in linear statics problems (everything is calculated at once) and therefore if the user needs to print some nodal quantities then simply defines for keyword `nod_step` the value `sel_all` which results in selection of all time/load steps (line 107). Results from all load cases are required to print out in line 108 using the same value `sel_all` for the keyword `sel_nodlc`. Having the time/load steps and load cases specified, the print configuration of particular nodal quantities follows where for each quantity, the selection of nodes, where the given quantity will be printed out, is followed by the selection of the given quantity components. Thus line 109 specifies that for all nodes, all displacement components will be printed out while the line 110 selects no nodes (keyword value `sel_no`) for nodal strains, i.e. no nodal strains will be printed. The same option is specified for nodal stresses (line 111), and nodal other values (line 112) and thus they do not be printed too. Configuration in line 113 enables the reaction output.

Configuration of nodal values output is followed by the similar configuration of element values output performed in all integration points on the selected elements. It starts with selection of time/load steps (line 115) and load cases (line 116). Using the same keyword values as for nodes results to the selection of all time/load steps and all load cases. Line 117 specifies that for all elements (keyword `strain_elems`), the output of all strain components (keyword `elemstrain_comp`) will be performed with no transformation of components (keyword `elemstra_transfid`). Line 118 specifies that for all elements (keyword `stress_elems`), the output of all stress components (keyword `elemstress_comp`) will be performed with no transformation of components (keyword `elemstre_transfid`). Line 119 defines that no `other` values will be printed out. The last item of text output configuration is given in line 121 where output values at user defined points on elements can be specified but it has not been not fully implemented so no time steps are selected in this case.

Should be noted that the problem is linear statics and therefore `other` values must not be required to be printed out because the material model is linear elastic and it defines zero number of internal variables which indicates that `other` arrays on integration points are not allocated and required output of these values would lead to segmentation fault errors.

The second part controls output in the various formats used in graphic postprocessor tools. In this example, the GiD format will be required which allows for the most advanced configuration of the output. The part configuring this output is listed below:

```
begsec_outdrv
```

```
# Description of output to the graphics file in GiD format
#-----------------------------------------------------------
outgr_format  grfmt_gid
# graphics output file name without extension
cant3d
# setup for nodal values
sel_nodstep  sel_all
sel_nodlc    sel_all
displ_nodes  sel_all   displ_comp   sel_all
strain_nodes sel_no
stress_nodes sel_no
other_nodes  sel_no
force_nodes  sel_all   force_comp   sel_all
# setup for element values
sel_elemstep sel_all
sel_elemlc   sel_all
strain_elems sel_all   elemstrain_comp sel_all
   elemstra_transfid 0
stress_elems sel_all   elemstress_comp sel_all
   elemstre_transfid 0
other_elems  sel_no
```

```
146 endsec_outdrv
```

Line 125 defines the format used for the result output with help of keyword `outgr_format` whose value is set to `gid`. This results into one GiD file with all result quantities (`cant3d.res`) that will be specified later in this part and another file with the mesh description (`cant3d.msh`). The common GiD file name is given in line 126 to which the corresponding suffix will be added automatically. Lines 128–141 contains the configuration of the output which uses the same keywords as in the previous part with only several differences described in the following text. The output of reactions is generally involved in the configuration of nodal forces output (line 135) where for all nodes (keyword `force_nodes`), all force components (keyword force_comp) will be printed out which results in output of nodal load components as well as reactions.

There is also used different selection of strain and stress components on elements from the one used in chapter 10 where `sel_mtx` optional value is used (lines 139, 140 in cant2d.pr). In this example, the selection of strain and stress components is provided by `sel_all` selection type which manages the output of strain and stress components as independent scalar values.

The last part controls output of selected quantities in particular time/load steps which can be used for creation of diagrams which cannot be used in the case of linear statics problems and therefore the end of `outdrv` section has the following content:

```
100 begsec_outdrv
```

```
143 # Text output of diagrams
144 # ------------------------
145 numdiag 0
146 endsec_outdrv
```

where the line 145 defines that the number of diagram files created is zero.

11.14 Preprocessor file

This section contains listing of the whole preprocessor file.

```
begsec_files
cantilever3d.top
mesh_format    sifel
edge_numbering 1
endsec_files



begsec_probdesc
Cantilever beam 5x0.3x0.5 m loaded by the various load
mespr 1
problemtype linear_statics
```

```
straincomp 1
strainpos  1
strainaver 0
stresscomp 1
stresspos  1
stressaver 0
othercomp  0
reactcomp  1

adaptivity     0
stochasticcalc 0
homogenization 0
noderenumber   0

stiffmatstor   skyline_matrix
typelinsol     ldl
endsec_probdesc


begsec_loadcase
num_loadcases      4
#temperature load type for the first load case
lc_id  1
temp_load_type    0
#temperature load type for the second load case
lc_id  2
temp_load_type    0
#temperature load type for the third load case
lc_id  3
temp_load_type    0
#temperature load type for the fourth load case
lc_id  4
temp_load_type    1
endsec_loadcase


begsec_mater
num_mat_types 2
mattype elisomat   num_inst 1
1 25.0e9 0.25
mattype therisodilat num_inst 1
1 1.2e-5
endsec_mater
```

```
begsec_crsec
num_crsec_types    0
endsec_crsec


begsec_nodvolpr
# number of degrees of freedom for all nodes
ndofn 3 propid 1
nod_temper  propid 1 lc_id 4 temperature 20.0
endsec_nodvolpr


begsec_nodsurfpr
# fixation of nodes on the left surface of beam
bocon propid 3 num_bc 3 dir 1 cond 0.0
                        dir 2 cond 0.0
                        dir 3 cond 0.0
endsec_nodsurfpr


begsec_nodedgpr
# prescribed displacements at nodes on the top left edge
bocon propid 4 num_bc 1 dir 3 cond -8.0e-3 lc_id 3
endsec_nodedgpr


begsec_elvolpr
el_type     propid 1  linearhex
el_mat      propid 1  num_mat 2 type elisomat     type_id 1
                                type therisodilat type_id 1
volume_load  propid 1  lc_id 1  ncomp 3
             func_type stat coord_sys 1
             load_comp 0.0 0.0 -24.0e3
endsec_elvolpr


begsec_elsurfpr
surf_load propid 5  lc_id   2  ncomp 3  func_type pars
          coord_sys 1 load_comp 0.0 0.0 -30.0e3+6.0e3*x
endsec_elsurfpr


begsec_outdrv
# Description of output to the text file
```

```
# -------------------------------------
textout 1
# text output file name
cant3d.out
# text output at nodes
sel_nodstep   sel_all
sel_nodlc     sel_all
displ_nodes   sel_all displ_comp sel_all
strain_nodes  sel_no
stress_nodes  sel_no
other_nodes   sel_no
reactions     1
# text output at elements
sel_elemstep  sel_all
sel_elemlc    sel_all
strain_elems  sel_all elemstrain_comp sel_all
   elemstra_transfid 0
stress_elems  sel_all elemstress_comp sel_all
   elemstre_transfid 0
other_elems   sel_no
# text output at user defined points
sel_pointstep sel_no

# Description of output to the graphics file in GiD format
#--------------------------------------------------------
outgr_format  grfmt_gid
# graphics output file name without extension
cant3d
# setup for nodal values
sel_nodstep  sel_all
sel_nodlc    sel_all
displ_nodes  sel_all   displ_comp   sel_all
strain_nodes sel_no
stress_nodes sel_no
other_nodes  sel_no
force_nodes  sel_all   force_comp   sel_all
# setup for element values
sel_elemstep sel_all
sel_elemlc   sel_all
strain_elems sel_all   elemstrain_comp sel_all
   elemstra_transfid 0
stress_elems sel_all   elemstress_comp sel_all
   elemstre_transfid 0
other_elems  sel_no
```

```
# Text output of diagrams
# -----------------------
numdiag 0
endsec_outdrv
```

Chapter 12

Nonlinear statics problem - perfect plasticity

This section describes how to prepare MEFEL input file with help of MECHPREP for the nonlinear statics problem of simply supported beam. The nonlinearity is induced by the material model based on perfect plasticity with von Misses yield criterion. The beam is subjected to two load cases:

1. Dead weight load f_c=27 kN/m^3 which is represented by volume load on elements calculated approximately from the given material density ρ=2700 kg/m^3. This load is assumed to be a constant.
2. Additionally, the beam is subjected to vertical force F_p acting in the middle of top surface of the beam. The force is applied on the beam with help of rigid plate of 40 mm width and the value of this force increases proportionally until the limit bearing capacity of the beam is attained.

The beam has length 600 mm and rectangular cross section 150 mm×10 mm. Material of the beam is assumed to be perfectly plastic and isotropic one where yield stress f_s=20 MPa, Young's modulus E=20 GPa and Poisson's ratio ν=0.35. The settings of the example is depicted in Fig. 12.1.

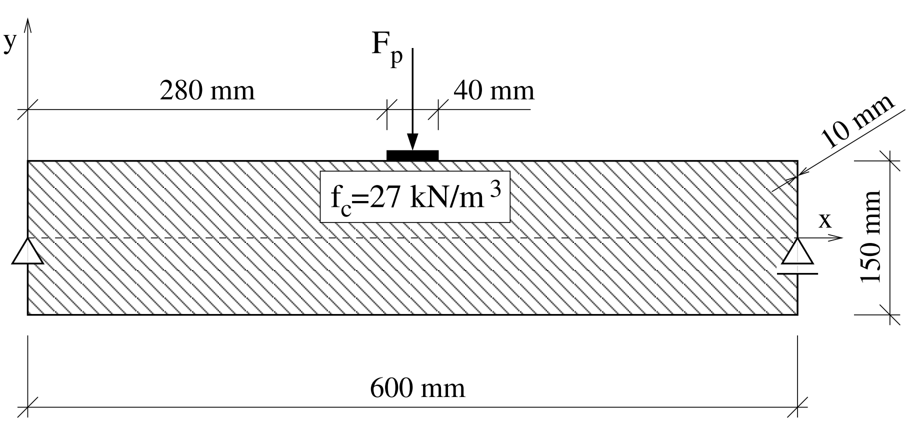


Figure 12.1: Settings of beam with von Misses plasticity in 2D

12.1 Topology file

In the first step, a mesh file must be created and corresponding property identifiers must be defined in order to MECHPREP can work. The mesh file can be created either manually or it can be generated with help of T3D mesh generator. The input file (`j2beam.t3d.in`) for T3D is listed below

```
# Mesh for simply supported beam, run command:
# /home/dr/Bin/T3d -p 264 -i j2beam.t3d.in -o j2beam.t3d -d 0.050 -X -$

# base vertices
vertex  1 xyz  0.000  0.000  0.0  property 5
vertex  2 xyz  0.000  0.075  0.0  property 2
vertex  3 xyz  0.200  0.075  0.0
vertex  4 xyz  0.280  0.075  0.0
vertex  5 xyz  0.300  0.075  0.0  property 6
vertex  6 xyz  0.320  0.075  0.0
vertex  7 xyz  0.400  0.075  0.0
vertex  8 xyz  0.600  0.075  0.0  property 1
vertex  9 xyz  0.600  0.000  0.0  property 7
vertex 10 xyz  0.600 -0.075  0.0  property 4
vertex 11 xyz  0.400 -0.075  0.0
vertex 12 xyz  0.300 -0.075  0.0  property 8
vertex 13 xyz  0.200 -0.075  0.0
vertex 14 xyz  0.000 -0.075  0.0  property 3

# edges
curve  1 vertex  1  2 property 2
curve  2 vertex  2  3 property 1
curve  3 vertex  3  4 factor 0.1 property 1
curve  4 vertex  4  5 factor 0.1 property 5
curve  5 vertex  5 12 factor 0.1
curve  6 vertex 12 13 factor 0.1 property 3
curve  7 vertex 13 14 property 3
curve  8 vertex 14  1 property 2
curve  9 vertex  5  6 factor 0.1 property 5
curve 10 vertex  6  7 factor 0.1 property 1
curve 11 vertex  7  8 property 1
curve 12 vertex  8  9 property 4
curve 13 vertex  9 10 property 4
curve 14 vertex 10 11 property 3
curve 15 vertex 11 12 factor 0.1 property 3

# beam
patch 1 normal 0 0 1 boundary curve -1 -2 -3 -4 -5 -6 -7 -8
```

```
   size def property 1
patch 2 normal 0 0 1 boundary curve -9 -10 -11 -12 -13 -14
   -15 5 size def property 1
```

The mesh is generated by the running of the following command:

```
T3d -p 264 -i j2beam.t3d.in -o j2beam.t3d -d 50.0 -X -$
```

and if everything run well the mesh file `j2beam.t3d` will be created including all property identifiers. The property identifiers are generated according to Fig. 4.1 but there are assigned additional vertex property identifiers to nodes in the middle of beam edges (property id 5-8). There is also small area on top edge in the midpoint neighbourhood which has got assigned edge property identifier 5 instead of 1. Resulting file `j2beam.t3d` can be converted to the `sifel` natural format by the following command:

```
t3dtosiffor j2beam.t3d j2beam.top 0
```

where the convertor `t3dtosiffor` can be found in SIFEL/PREP/MESHTOOL folder. The file `j2beam.top` can be displayed with help of MeshEditor tool where it is possible to visualize the property identifiers by various colors as in Fig. 12.2.

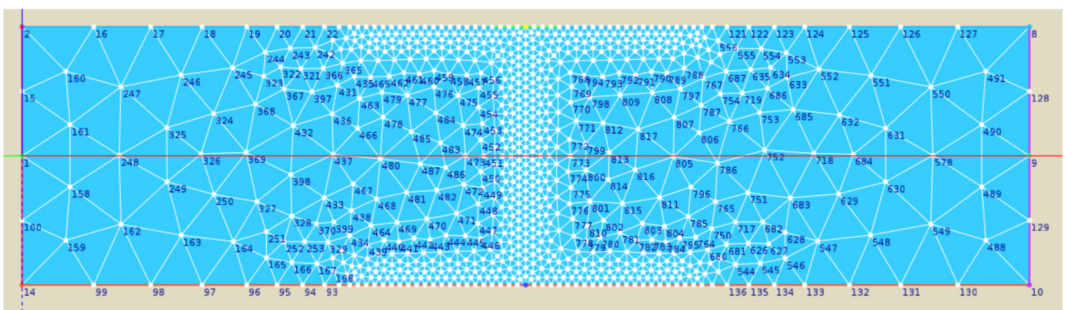


Figure 12.2: Simply supported beam - generated mesh with visualized property identifiers

12.2 Preprocessor file - section `files`

Having the mesh file with property identifiers created, a preprocessor file should be prepared. Preprocessor file is composed from the several sections that may be organized arbitrarily in the file but they will be introduced in the order of the real processing in MECHPREP. Each preprocessor file must contain section `files` which in this case has the following contents:

```
begsec_files
j2beam.t3d
mesh_format t3d
edge_numbering 1
endsec_files
```

where the topology input file is being specified to be `j2beam.t3d` in line 2 then the format of the topology file is given in line 3 and finally, it is given that edge and surface property identifiers are given also on elements - line 4. The command in line 4 is in accordance with setup of mesh generator `T3d` (argument `-p 264` on the command line) which cause the write of edge property numbers on elements to the topology file. See section 4.1 for more details about the T3D mesh format.

It is supposed that materials and cross sections will be given in the preprocessor file directly and thus no file names of material and cross section files are given. More details about this preprocessor section can be found in 9.1.

12.3 Preprocessor file - section `probdesc`

This section is used by MEFEL to identify the type of problem solved, the type of nonlinear and linear equation system solver and some other calculation setup. Some details about the particular cases of `probdesc` setup can be found in [10]. In this case, the following setup is chosen:

```
begsec_probdesc
Simply supported beam with von Mises plasticity
mespr 1        # detail output
problemtype mat_nonlinear_statics # non-linear statics

straincomp 1  # strains are computed
strainpos  1  # strains are computed in integration points
strainaver 0  # strains are not averaged
stresscomp 1  # stresses are computed
stresspos  1  # stresses are computed in integration points
stressaver 0  # stresses are not averaged
othercomp 1   # internal variables are not computed
otherpos  1   # internal variables are computed in integration points
otheraver 0   # internal variables are not averaged
reactcomp 1   # reactions are computed

adaptivity      0  # adaptivity is not used
stochasticcalc  0  # deterministic computation
homogenization  0  # homogenization is not applied
noderenumber   no_renumbering  # nodes are not renumbered

type_of_nonlin_solver  newton # the Newton-Raphson
stiffmat_type  initial_stiff # the initial stiffness matrix approach
nr_num_steps   30              # the number of increments
nr_num_iter    30              # the number of iterations within increment
```

```
33 nr_error        1.0e-02         # the required relative norm of residual
34 nr_init_incr    1.0             # the initial increment
35 nr_minincr      1.0e-08         # the minimum increment
36 nr_maxincr      1.0e+03         # the maximum increment
37 hdbackup        nohdb           # no HD backup is required
38 stiffmatstor    skyline_matrix  # the stiffness matrix is stored in skyline
39 typelinsol      ldl             # system of linear algebraic equations is solved by LDL
40 endsec_probdesc
```

In this section, the title of the problem solved is given in line 9, detailed message printing is switched on (line 10) and material nonlinear statics problem type is specified to be solved in line 11.

In the sequential two blocks of commands, the strain calculation is on (line 13) and strains will be calculated at integration points (line 14) and therefore no averaging of strains is necessary (line 15). The same setup for stress computation is defined in lines 16–18 and for internal (other) variables in lines 19–21. Finally, the calculation of reactions is required in line 22. All additional advanced techniques such as mesh adaptivity (line 24), stochastic calculations (line 25), homogenization techniques (line 26) and node renumbering (line 27) are not taken into account in the computation.

The last block of commands defines the type of solver of system of nonlinear algebraic equations. The line 29 defines that the Newton-Raphson iterative procedure is used, where the system matrix is assmebled/updated only at the beginning of the computation (line 30). The total number of performed load steps is set to 30 (line 31) and maximum number of internal steps in the iteration of residual vector is also 30 (line 32). The relative norm of residual is defined in line 33 to be 10^{-2} which means that the residual vector norm is one hundredth of the norm of the actual load vector. Setup of magnitude of load steps follows - the initial load coefficient increment is 1.0 (line 34), the minimum load coefficient increment is set to 10^{-8} and maximum load coefficient increment is given to be 10^3. Line 37 defines that backup of particular load steps on harddisk is not performed. The last two lines define that skyline storage of system matrix is used (line 38) and system will be solved with help of LDL decomposition method (line 39) which can be applied in this case because the system matrix is symmetric and positive definite for the associated plasticity.

12.4 Preprocessor file - section `loadcase`

This section defines the number of load cases and some details about the content of particular load cases. According to the problem setting, this section looks as follows:

```
begsec_loadcase
num_loadcases      2
#temperature load type for the proportional load case
lc_id  1
```

```
47 temp_load_type    0
48 #temperature load type for the constant load case
49 lc_id  2
50 temp_load_type    0
51 endsec_loadcase
```

where two individual load cases are established in line 44, and for each load case, the type of temperature load load is defined with help of command pairs `tempr_type_lc_id` and `temp_load_type`. The all load cases are composed just from force load, i.e. no temperature load is defined by in lines 47 and 50. Should be noted that only the first load case is proportional, i.e. scaled by the gradually increasing load coefficient while the second load case remains constant until the end of computation. At the beginning of computation, the value of load coefficient is zero and the initial increment of load coefficient is given in line 34. During the Newton-Raphson procedure, the values of load coefficient increment are increased/decreased automatically according to the progress of iteration keeping the range $[10^{-8};10^{3}]$. The load stepping is stopped if the maximum number of load steps is attained or the solver cannot reduce the load coefficient increment which has been already set to minimum value defined in line 35.

12.5 Preprocessor file - section `mater`

This section contains the list of material models and their parameters. In this example, the distribution of material properties is assumed to be homogeneous where the J2 flow yield criterion for the plastic behaviour while the elastic behviour is described by linear elastic isotropic material. Therefore two material types must be defined in this section whose content is listed below

```
54 begsec_mater
55 num_mat_types 2
56 # elastic isotropic material
57 mattype elisomat num_inst 1
58 1 70.0e9 0.35
59 # von Mises yield condition
60 mattype jflow num_inst 1
61 1 70.0e6 0.0 1 50 1.0e-6
62 endsec_mater
```

In the section, two material types (elastic isotropic and von Misses plasticity) are being defined (line 55) and sequential lines contains the specification of these material types. The line 57 defines that the material type is elastic isotropic with one instance of material parameter set. Line 58 defines the first instance of material parameter set of elastic isotropic material which requires two parameters - Young's modulus (70 GPa) and Poisson's ratio (0.35). The second material model is represented by plasticity with von Misses yield criterion with one instance of parameter set (line 60). The parameter set is defined on the last line where the yield stress is set to τ_0=70 MPa, hardening modulus to 0, i.e. perfect plasticity with no hardening is to be solved. Additionally, the setup of stress return algorithm must be given at the end of material parameter record. In this case, the cutting plane stress return algorithm is selected, the maximum number of iterations is set to 50 and the error of yield function residual is set to be 10^{-6}.

12.6 Preprocessor file - section `crsec`

This section contains the list of cross sections and their parameters. In this example, the cross section is given by the thickness 0.01 m which is uniform across the beam. The section has the following content:

```
65 begsec_crsec
66 num_crsec_types 1
67 crstype csplanestr num_inst 1
68 1 0.01
69 endsec_crsec
```

In the section, only one cross section type (for plane elements) is being defined (line 66) and sequential lines contains the specification of this one cross section type. The line 67 defines that the cross section type is for plane stress/strain elements (`csplanestr`) with one instance of cross section parameter set. The last line 68 defines the first instance of cross section parameter set for plane elements which requires just one parameters - thickness (0.01 m).

12.7 Preprocessor file - number of nodal DOFs

The section `nodsurfpr` defines common properties for group of nodes involved in the surface with specific property id. In 2D problems, the most common use of this section is for the specification of number of DOFs at nodes. In this example, two DOFs are defined in all nodes of the mesh. The section content is listed below:

```
88 begsec_nodsurfpr
89 ndofn 2 propid 1  # number of degrees of freedom at nodes
90 endsec_nodsurfpr
```

where the command `ndofn` in line 89 defines two DOFs at all nodes with surface property id 1, i.e. on the whole domain solved.

12.8 Preprocessor file - Dirichlet's boundary conditions

Dirichlet's boundary condition prescribes values of primary unknowns defined in the problem solved and they can be prescribed with help of `bocon` command. In this example, this type of boundary conditions are represented by fixed node on the left end point of beam axis and vertically fixed node on right end point of the beam axis. These nodes are marked by the vertex property identifiers 5 and 7 respectively and therefore the `bocon` command should be placed in the `nodvertpr` section whose content is listed below

```
73 begsec_nodvertpr
74 bocon  propid 5  num_bc 2  dir 1  cond 0.0  dir 2 cond 0.0 # left support
75 bocon  propid 7  num_bc 1  dir 2  cond 0.0  # right support
```

```
77 endsec_nodvertpr
```

where displacements are prescribed to be zero values for all DOFs (defined in the previous section) of node with vertex property id 5 (line 74) while for node with property id 7, only the vertical (2nd) DOF is prescribed to be zero (line 75).

12.9 Preprocessor file - simulation of rigid plate

Another type of nodal boundary condition represents the simulation of rigid plate for the applied load transfer which can be simulated with help of coupled vertical DOFs at narrow area around the middle of beam. These nodes has got assigned the edge property id 5 in the generator and therefore the `dof_coupl` command providing DOF coupling of selected nodes should be placed in `nodedgpr` section

```
81 begsec_nodedgpr
82 # rigid plate <=> coupled DOFs
83 dof_coupl propid 5 ndir 1 dir 2
84 endsec_nodedgpr
```

where line 83 represents the suitable record of the preprocessor command coupling vertical DOFs of all nodes at edge with property id 5 to the single DOF.

12.10 Preprocessor file - proportional load

The proportional load is represented by vertical force applied in the middle of beam on the top edge. The node has got assigned vertex property id 6 by the mesh generator and thus the `nod_load` command should be placed in the `nodvertpr` section.

```
73 begsec_nodvertpr
```

```
76 nod_load  propid 6  lc_id 1  load_comp 0.0 -1.0e3  # proportional load
77 endsec_nodvertpr
```

In the command in line 76, load case id must be set 1 in order to be load scaled by the gradually increasing load coefficient. In this case, the basic magnitude of vertical load component is set to -10^3 and thus the resulting load coefficient obtained at the end of computation represents the limit vertical force in kN.

12.11 Preprocessor file - element type, material model, cross section

The FE type and material model are essential properties of elements which must be given in all kinds of problems. In this example, all elements have the same FE type and material model and therefore the most simple way how to assign them to all elements is the use of corresponding commands in the element section `elsurfpr` keeping in mind that the surface property id 1 is the same for all elements. The element type can be assigned by the command `el_type` while material model by the command `el_mat`.

```
93  begsec_elsurfpr
94  el_type   propid 1 planeelementlt strastrestate planestress
95  el_mat    propid 1 num_mat 2  type jflow    type_id 1
96                                type elisomat type_id 1
97  el_crsec propid 1  type csplanestr  type_id 1
```

```
101 endsec_elsurfpr
```

where line 94 assigns triangle element with linear shape functions. The same material elastoplastic model is assumed to be on all elements and it is assigned by the command in lines 95 and 96. The model is composed from two independent parts (keyword `num_mat`) one for plasticity (`jflow` - line 95) and one for isotropic elasticity (`elisomat` - line 96). Both models refers to the first instance of material parameter set with help of keywords `type_id`. Cross section type and parameters are defined in line 97 with help of command `el_crsec` which refers to the first instance of cross section parameters defined in above in the file in section `crsec` (line 68).

12.12 Preprocessor file - constant load

In the second load case, the beam is loaded by dead weight load which must be applied to all elements in the mesh. It can be achieved by the command `volume_load` placed in the section `elsurfpr` because all elements has got assigned the same surface property id 1. The syntax of the command is listed below

```
93  begsec_elsurfpr
```

```
98  volume_load propid 1 lc_id 2  ncomp 2
99              func_type stat coord_sys 1
100             load_comp 0.0 -12.0e3
101 endsec_elsurfpr
```

where the line 98 defines volume load on all elements with surface property id 1, the load is applied in the load case 2 which is kept constant for the whole computation procedure. The command continues in line 99 where the load is defined to be with constant distribution (keyword `func_type`), applied in the global coordinate system (keyword `coord_sys`) and finally, two load components are given in line 100 - the dead weight load is applied in the vertical direction.

12.13 Setup of the result output

The last section that has to be specified is represented by section `outdrv` where the output of results from MEFEL should be configured. More details about this section can be found in [10]. The section is composed from three parts dealing with different forms of result output. The first part controls output to the file in the text form has the following content:

```
104 begsec_outdrv
105 #--------------------------------
106 #  Definition of MEFEL output  |
107 #--------------------------------
108
109 # description of output to the text file
110 textout 0
```

```
152 endsec_outdrv
```

In this example, no text output of results is required (line 110).

The second part controls output in the various formats used in graphic postprocessor tools. In this example, the GiD format is required which allows for the most advanced configuration of the output. The part configuring this output is listed below:

```
104 begsec_outdrv
```

```
112 # description of output to the text file
113 # 3 = GiD format - one huge file
114 outgr_format grfmt_gid
115
116 # graphics output file name without extension
117 j2beam
118
119 sel_nodstep  sel_all
120 sel_nodlc    sel_all
121 displ_nodes  sel_all  displ_comp sel_all
122 strain_nodes sel_no
123 stress_nodes sel_no
124 other_nodes  sel_no
125 force_nodes  sel_all  force_comp sel_all
126
127 sel_elemstep sel_all
128 sel_elemlc   sel_all
129 strain_elems sel_all  elemstrain_comp sel_mtx
       elemstra_transfid 0
130 stress_elems sel_all  elemstress_comp sel_mtx
       elemstre_transfid 0
```

```
131 other_elems  sel_all  elemother_comp  sel_list numlist_items
      1 5
```

```
152 endsec_outdrv
```

Line 114 defines the format used for the result output with help of keyword `outgr_format` whose value is set to `gid`. This results into one GiD file with all result quantities (`j2beam.res`) that will be specified later in this part and another file with the mesh description (`j2beam.msh`). The common GiD file name is given in line 117 to which the corresponding suffix will be added automatically.

After that the time/load steps and load case numbers must be given at which the nodal results will be printed out. In this case, all load steps are selected (line 119) and all load cases are selected (line 120). Should be noted that load cases cannot be selected independently because of nonlinearity of the problem and results are calculated for sum of proportional (1st) load case and constant (2nd) load case and thus load case selection should be just `sel_all` defined for keyword `sel_nodlc`. Having the time/load steps and load cases specified, the print configuration of particular nodal quantities follows providing the selection of nodes for each quantity, where the given quantity will be printed out, followed by the selection of the given quantity components. Thus line 121 specifies that for all nodes, all displacement components will be printed out while the line 122 selects no nodes (keyword value `sel_no`) for nodal strains, i.e. no nodal strains will be printed. The same option is specified for nodal stresses (line 123), and nodal `other` values (line 124) and therefore they do not be printed too. Line 125 defines that at all nodes all components of internal force vector will be printed.

Configuration of nodal values output is followed by the similar configuration of element values output performed in all integration points on the selected elements. It starts with selection of time/load steps (line 127) and load cases (line 128). Using the same keyword values as for nodes results to the selection of all time/load steps and all load cases. Line 129 specifies that for all elements (keyword `strain_elems`), the output of all strain components (keyword `elemstrain_comp`) will be performed with no transformation of components (keyword `elemstra_transfid`). Line 130 specifies that for all elements (keyword `stress_elems`), the output of all stress components (keyword `elemstress_comp`) will be performed with no transformation of components (keyword `elemstre_transfid`). Selection of all strain and stress components on elements is specified by `sel_mtx` optional value that provides the output of strains and stresses in the tensorial form (all their components) which allows for better postprocessing in the GiD (calculation of principal values and vectors). Line 131 defines that for all elements, one `other` value will be printed out. It is consistency parameter (plastic multiplier) γ defined in plasticity model which is stored in the `other` array as the 5-th component. In this case, the list selection type `sel_list` is used in `other` array component selection and this list contains only one item (`numlist_items 1`). The list of item identifiers stands at the end of record and in this case, it contains just number 5, i.e. reference to the 5-th component of other array.

Should be noted that the order of internal variables of the `other` array can be arbitrary and it is defined inside the given material model. In this example, the von Misses plasticity model is used which is contained in the `j2flow.cpp` source file. Most of plasticity models in MEFEL adopt the following order of internal variables for the `other` array:

1. plastic strain components $\boldsymbol{\varepsilon}_p$ stored in Voigth notation,
2. consistency parameter (plastic multiplier) γ, $\dot{\boldsymbol{\varepsilon}}_p = \dot{\gamma}\dfrac{\partial g}{\partial \boldsymbol{\sigma}}$,
3. hardening parameters.

The beam is solved as a plane stress problem where the strain vector contains only four components and therefore the consistency parameter is stored on the 5-th position of `other` array.

The last part controls output of selected quantities in particular time/load steps which can be used for creation of diagrams. It will be suitable to print the value of vertical displacement in the middle of beam, vertical reaction in the left support and the value of load coefficient in all load steps. The content of last part of `outdrv` section is listed below:

```
104 begsec_outdrv
```

```
133 # text output of graphs
134 # number of created files with diagrams
135 numdiag 1
136 j2beam.dat
137 numunknowns  3       # number of printed unknowns
138 sel_diagstep sel_all # type of load step selection = all
    load steps
139
140 # point type = node, node id = 5
141 point atnode node 5
142 # printed nodal quantity = displacement, 2nd component
143 quant_type pr_displ compid 2
144 # point type = node, node id = 1
145 point atnode node 1
146 # printed nodal quantity = reaction, 2nd component
147 quant_type pr_react compid 2
148 # point type = node, node id = 5
149 point atnode node 5
150 # printed nodal quantity = load coefficient
151 quant_type pr_appload
152 endsec_outdrv
```

where the line 135 defines that the number of diagram files created is 1. The name of the output diagram file follows on the next line 136. For each diagram file, the number of printed quantities must be given (line 137) and this number equals to the number of columns in the table created in the diagram file. The values of selected quantities are printed to the file in each load step selected. All load steps are selected in this case in line 138.

After the above initial setup, the definition of particular quantities follows. Each record contains definition of point at which the given quantity should be printed out followed by the type of quantity. The first quantity record starts at line 141 where the position at node 5 is given and the first quantity type is given in line 143 which defines the type of quantity to be displacement (`pr_displ`) where the vertical displacement component is selected (`compid 2`). The second quantity record starts at line 145 where the position at node 1 is given and the quantity type is given in line 147 which defines the type to be reaction (`pr_react`) in vertical direction (`compid 2`). The last required quantity is the value of load coefficient where the same value is printed in arbitrary node. In this case node 5 is selected (line 149) and the type of quantity `pr_appload` is defined in line 151.

12.14 Preprocessor file

This section contains listing of the whole preprocessor file.

```
begsec_files
j2beam.t3d
mesh_format t3d
edge_numbering 1
endsec_files


begsec_probdesc
Simply supported beam with von Mises plasticity
mespr 1        # detail output
problemtype mat_nonlinear_statics # non-linear statics

straincomp 1  # strains are computed
strainpos  1  # strains are computed in integration points
strainaver 0  # strains are not averaged
stresscomp 1  # stresses are computed
stresspos  1  # stresses are computed in integration points
stressaver 0  # stresses are not averaged
othercomp 1   # internal variables are not computed
otherpos  1   # internal variables are computed in
   integration points
otheraver 0   # internal variables are not averaged
reactcomp 1   # reactions are computed

adaptivity     0  # adaptivity is not used
stochasticcalc 0  # deterministic computation
homogenization 0  # homogenization is not applied
noderenumber   no_renumbering  # nodes are not renumbered

type_of_nonlin_solver  newton # the Newton-Raphson
```

```
stiffmat_type  initial_stiff # the initial stiffness matrix
   approach
nr_num_steps   30            # the number of increments
nr_num_iter    30            # the number of iterations
   within increment
nr_error       1.0e-02       # the required relative norm
   of residual
nr_init_incr   1.0           # the initial increment
nr_minincr     1.0e-08       # the minimum increment
nr_maxincr     1.0e+03       # the maximum increment
hdbackup       nohdb         # no HD backup is required
stiffmatstor   skyline_matrix # the stiffness matrix is
   stored in skyline
typelinsol     ldl           # system of linear algebraic
   equations is solved by LDL
endsec_probdesc


begsec_loadcase
num_loadcases     2
#temperature load type for the proportional load case
lc_id  1
temp_load_type   0
#temperature load type for the constant load case
lc_id  2
temp_load_type   0
endsec_loadcase


begsec_mater
num_mat_types 2
# elastic isotropic material
mattype elisomat num_inst 1
1 70.0e9 0.35
# von Mises yield condition
mattype jflow num_inst 1
1 70.0e6 0.0 1 50 1.0e-6
endsec_mater


begsec_crsec
num_crsec_types 1
crstype csplanestr num_inst 1
1 0.01
endsec_crsec
```

```
# properties of nodes defined by vertices
begsec_nodvertpr
bocon  propid 5  num_bc 2  dir 1  cond 0.0  dir 2 cond 0.0 #
   left support
bocon  propid 7  num_bc 1  dir 2  cond 0.0  # right support
nod_load  propid 6  lc_id 1  load_comp 0.0 -1.0e3  #
   proportional load
endsec_nodvertpr


# properties of nodes on edges
begsec_nodedgpr
# rigid plate <=> coupled DOFs
dof_coupl propid 5 ndir 1 dir 2
endsec_nodedgpr


# properties of nodes defined at regions
begsec_nodsurfpr
ndofn 2 propid 1  # number of degrees of freedom at nodes
endsec_nodsurfpr


begsec_elsurfpr
el_type  propid 1 planeelementlt strastrestate planestress
el_mat   propid 1 num_mat 2  type jflow    type_id 1
                             type elisomat type_id 1
el_crsec propid 1  type csplanestr  type_id 1
volume_load propid 1 lc_id 2  ncomp 2
            func_type stat coord_sys 1
            load_comp 0.0 -12.0e3
endsec_elsurfpr


begsec_outdrv
#-------------------------------
#  Definition of MEFEL output  |
#-------------------------------

# description of output to the text file
textout 0

# description of output to the text file
```

```
# 3 = GiD format - one huge file
outgr_format grfmt_gid

# graphics output file name without extension
j2beam

sel_nodstep  sel_all
sel_nodlc    sel_all
displ_nodes  sel_all  displ_comp sel_all
strain_nodes sel_no
stress_nodes sel_no
other_nodes  sel_no
force_nodes  sel_all  force_comp sel_all

sel_elemstep sel_all
sel_elemlc   sel_all
strain_elems sel_all  elemstrain_comp sel_mtx
   elemstra_transfid 0
stress_elems sel_all  elemstress_comp sel_mtx
   elemstre_transfid 0
other_elems  sel_all  elemother_comp  sel_list numlist_items
   1 5

# text output of graphs
# number of created files with diagrams
numdiag 1
j2beam.dat
numunknowns  3       # number of printed unknowns
sel_diagstep sel_all # type of load step selection = all
   load steps

# point type = node, node id = 5
point atnode node 5
# printed nodal quantity = displacement, 2nd component
quant_type pr_displ compid 2
# point type = node, node id = 1
point atnode node 1
# printed nodal quantity = reaction, 2nd component
quant_type pr_react compid 2
# point type = node, node id = 5
point atnode node 5
# printed nodal quantity = load coefficient
quant_type pr_appload
endsec_outdrv
```

Chapter 13

Nonlinear statics problem - scalar damage model

This section describes how to prepare MEFEL input file with help of MECHPREP for the nonlinear statics problem of simply supported beam with notch. The nonlinearity is induced by the material model based on perfect scalar isotropic damage model. The beam is subjected to two load cases:

1. Dead weight load f_c=12 kN/m^3 which is represented by volume load on elements calculated approximately from the given material density ρ=1200 kg/m^3. This load is assumed to be a constant.
2. Additionally, the beam is subjected to vertical force F_p acting in the middle of top surface of the beam. The force is applied on the beam with help of rigid plate of 40 mm width and the value of this force increases proportionally until the the crack propagation across the beam height is attained.

The beam has length 160 mm and rectangular cross section 40 mm×40 mm. Material of the beam is assumed to be mortar with Young's modulus E=10 GPa, Poisson's ratio ν=0.25, tensile strength f_t=0.4 MPa and the fracture energy G_f=9 J/m^2. The settings of the example is depicted in Fig. 13.1. With respect to the setting of the task, the

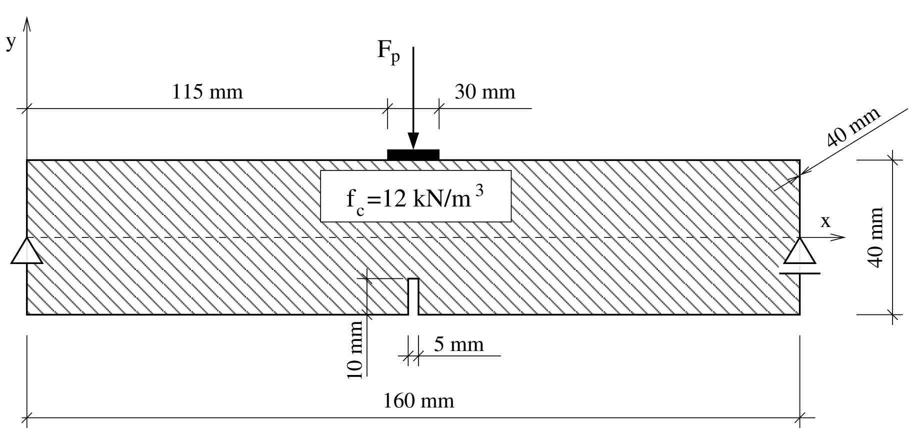


Figure 13.1: Settings of notched beam with scalar isotropic damage model in 2D

arclength method has to be exploited in order to capture the limit notch opening when the crack propagated almost across the whole beam height. Considering that the full crack propagation should be captured in the numerical simulation, the load path of such problem configuration contain in any case a drop after the peak load was attained and it would contain even snap-back in case of very narrow notch. In such the case, the displacement control must be used at least but for better results the loading will be controlled by the increasing distance of the corner nodes at mouth of the notch. This type of control is involves the arclength method implemented in MEFEL where the system of equations is complemented by the additional condition for the length of arc of the loading path in the form

$$\Delta l = \sqrt{\Delta \boldsymbol{r}^T \Delta \boldsymbol{r} + \Delta\lambda^2 \psi \boldsymbol{f}_p^T \boldsymbol{f}_p}, \tag{13.1}$$

where $\Delta\boldsymbol{r}$ is the vector of displacement increments, $\Delta\lambda$ is the increment of load coefficient, ψ is the scaling factor and $\boldsymbol{f}_p$ is the vector of proportional load.

13.1 Topology file

In the first step, a mesh file must be created and corresponding property identifiers must be defined in order to MECHPREP can work. The mesh file can be created either manually or it can be generated with help of T3D mesh generator. The input file (`damage-beam.t3d.in`) for T3D is listed below

```
# Mesh for simply supported beam, run command:
# /home/dr/Bin/T3d -p 264 -i damage-beam.t3d.in -o damage-beam.t3d -d 0.015 -X -$

# base vertices
vertex  1 xyz  0.0000   0.020  0.0  property 2
vertex  2 xyz  0.0500   0.020  0.0
vertex  3 xyz  0.0650   0.020  0.0
vertex  4 xyz  0.0800   0.020  0.0  property 5
vertex  5 xyz  0.0950   0.020  0.0
vertex  6 xyz  0.1100   0.020  0.0
vertex  7 xyz  0.1600   0.020  0.0  property 1
vertex  8 xyz  0.1600   0.000  0.0  property 8
vertex  9 xyz  0.1600  -0.020  0.0  property 4
vertex 10 xyz  0.1000  -0.020  0.0
vertex 11 xyz  0.0825  -0.020  0.0
vertex 12 xyz  0.0825  -0.010  0.0
vertex 13 xyz  0.0800  -0.010  0.0  property 7
vertex 14 xyz  0.0775  -0.010  0.0
vertex 15 xyz  0.0775  -0.020  0.0
vertex 16 xyz  0.0600  -0.020  0.0
vertex 17 xyz  0.0000  -0.020  0.0  property 3
vertex 18 xyz  0.0000   0.000  0.0  property 6

```

```

# edges
curve  1 vertex  1  2 property 1
curve  2 vertex  2  3 factor 0.1 property 1
curve  3 vertex  3  4 factor 0.1 property 5
curve  4 vertex  4 13 factor 0.1
curve  5 vertex 13 14 factor 0.1 property 3
curve  6 vertex 14 15 factor 0.1 property 3
curve  7 vertex 15 16 factor 0.1 property 3
curve  8 vertex 16 17 property 3
curve  9 vertex 17 18 property 2
curve 10 vertex 18  1 property 2

curve 11 vertex  4  5 factor 0.1 property 5
curve 12 vertex  5  6 factor 0.1 property 1
curve 13 vertex  6  7 property 1
curve 14 vertex  7  8 property 4
curve 15 vertex  8  9 property 4
curve 16 vertex  9 10 property 3
curve 17 vertex 10 11 factor 0.1 property 3
curve 18 vertex 11 12 factor 0.1 property 3
curve 19 vertex 12 13 factor 0.1 property 3

# beam
patch 1 normal 0.0 0.0 1.0 \
boundary curve -1 -2 -3 -4 -5 -6 -7 -8 -9 -10 \
size def property 1

patch 2 normal 0.0 0.0 1.0 \
boundary curve -11 -12 -13 -14 -15 -16 -17 -18 -19 4 \
size def property 1
```

The mesh is generated by the running of the following command:

```
T3d -p 264 -i damage-beam.t3d.in -o damage-beam.t3d -d 15.0 -X -$
```

and if everything run well the mesh file `damage-beam.t3d` will be created including all property identifiers. The property identifiers are generated according to Fig. 4.1 but there are assigned additional vertex property identifiers to nodes in the middle of beam edges (property id 5-8). There is also small area on top edge in the midpoint neighbourhood which has got assigned edge property identifier 5 instead of 1. Resulting file `damage-beam.t3d` can be converted to the `sifel` natural format by the following command:

```
t3dtosiffor damage-beam.t3d damage-beam.top 0
```

where the convertor `t3dtosiffor` can be found in SIFEL/PREP/MESHTOOL folder. The file `damage-beam.top` can be displayed with help of MeshEditor tool where it is possible to visualize the property identifiers by various colors as in Fig. 13.2.

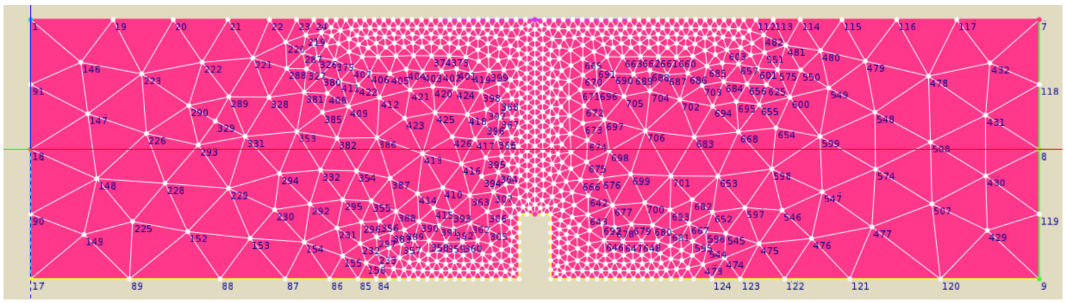


Figure 13.2: Simply supported notched beam - generated mesh with visualized property identifiers

13.2 Preprocessor file - section `files`

Having the mesh file with property identifiers created, a preprocessor file should be prepared. Preprocessor file is composed from the several sections that may be organized arbitrarily in the file but they will be introduced in the order of the real processing in MECHPREP. Each preprocessor file must contain section `files` which in this case has the following contents:

```
4 begsec_files
5 damage-beam.top
6 mesh_format sifel
7 edge_numbering 1
8 read_mat_strings no
9 read_mat_kwd yes
10 endsec_files
```

where the topology input file is being specified to be `damage-beam.top` in line 2 then the format of the topology file is given in line 3 and finally, it is given that edge and surface property identifiers are given also on elements - line 4. The command in line 4 is in accordance with setup of mesh generator `T3d` (argument `-p 264` on the command line) which cause the write of edge property numbers on elements to the topology file. See section 4.1 for more details about the T3D mesh format.

It is supposed that materials and cross sections will be given in the preprocessor file directly and thus no file names of material and cross section files are given. More details about this preprocessor section can be found in 9.1. Additionally, the material parameters will not be processed as strings but by particular material model `read` procedures (line 8) and keywords will be used in the parameter definition (line 9).

13.3 Preprocessor file - section probdesc

This section is used by MEFEL to identify the type of problem solved, the type of nonlinear and linear equation system solver and some other calculation setup. Some details about the particular cases of probdesc setup can be found in [10]. In this case, the following setup is chosen:

```
begsec_probdesc
Three point bending test of mortar MPA35 beam 40x40x160 mm
mespr 1
problemtype mat_nonlinear_statics

straincomp 1
strainpos  1
strainaver 0
stresscomp 1
stresspos  1
stressaver 0
othercomp  1
otherpos   1
otheraver  0
reactcomp  1

adaptivity      0
stochasticcalc  0
homogenization  0
noderenumber    0

type_of_nonlin_solver  arclg
stiffmat_type          ijth_tangent_stiff
1
10
lambda_determination minangle
al_num_steps         300
al_num_iter          50
al_error             1.0e-3
al_init_length       1.0e-6
al_min_length        1.0e-12
al_max_length        1.0e-5
al_psi               0.0
check_div            off
check_total_arcl     on
max_total_arc_length 0.0003
check_req_lambda     off
al_displ_contr_type  nodesdistincr
probdimal            2
```

```
51 11  15
52 hdbackup              nohdb
53 stiffmatstor          spdirect_stor_scr
54 typelinsol            spdirldl
55 endsec_probdesc
```

In this section, the title of the problem solved is given in line 11, detailed message printing is switched on (line 13) and material nonlinear statics problem type is specified to be solved in line 14.

In the sequential two blocks of commands, the strain calculation is on (line 17) and strains will be calculated at integration points (line 18) and therefore no averaging of strains is necessary (line 19). The same setup for stress computation is defined in lines 20–22 and for internal (other) variables in lines 23–25. Finally, the calculation of reactions is required in line 26. All additional advanced techniques such as mesh adaptivity (line 28), stochastic calculations (line 29), homogenization techniques (line 30) and node renumbering (line 31) are not taken into account in the computation.

The last block of commands defines the type of solver of system of nonlinear algebraic equations. The line 33 defines that the arclength method is used, where the system matrix is updated at the beginning of each load step and for each 10-th step of iteration (line 34-36). The load coefficient λ is solved from the quadratic equation in the arclength method where two roots are obtained generally. The line 37 specifies that the λ is chosen so that it yields the minimum angle between the actual and previous vectors of displacement increments. The total number of performed load steps is set to 300 (line 38) and maximum number of internal steps in the iteration of residual vector is 50 (line 39). The relative norm of residual is defined in line 40 to be 10^{-3} which means that the residual vector norm is one thousandth of the norm of the actual load vector. Setup of magnitude of load steps follows - the initial step length is 10^{-6} (line 41), the minimum step length is set to 10^{-8} (line 42) and maximum step length is given to be 10^{-12} (line 43). Length of arc can be calculated either form displacement increments or nodal distance increments or the norm of load vector increments is also taken into account. Selection of type of arclength method is controlled with help of coefficient ψ which is used as magnitude factor of the norm of load vector increments. In this case, the load vector increments are not taken into account and the load process is driven just by distance increments of selected nodes, i.e. ψ=0 (line 44). Checking of divergence in course of iteration process is switched off (line 45). Except of the total number of loading steps, there are two additional conditions that can be employed in the stopping criteria of the arclength method. The first criterion (line 46) defines the maximum total length of arc (line 47) which can be attained in the computation. The second criterion (line 48) defines the limit value of the load coefficient λ which cannot be exceeded. In this case, only total length of arc criterion is switched on (lines 46–47) while the maximum λ criterion is not used (line 48). The last setup option is connected with type of arclength method and selection of nodes whose distance increment will be used for the control of the method. The line 49 defines that the nodal distance increments will be used for the arclength control, line 50 defines the dimension of the problem, i.e. number of coordinate components used in the distance calculation and selection of two nodes follows in line 51 - nodes 11 and 15 are being specified in this case.

Line 52 defines that backup of particular load steps on harddisk is not performed. The last two lines define that spdirect_stor_scr storage of system matrix is used (line 53) and system will be solved with help of sparse direct LDL solver (line 54). It is possible to apply this setup in this case because the system matrix is symmetric and positive definite for the secant matrix defined in the scalar damage model.

13.4 Preprocessor file - section `loadcase`

This section defines the number of load cases and some details about the content of particular load cases. According to the problem setting, this section looks as follows:

```
57 begsec_loadcase
58 num_loadcases  2
59 #temperature load type for the first load case
60 lc_id   1
61 temp_load_type     0
62 #temperature load type for the second load case
63 lc_id   2
64 temp_load_type     0
65 endsec_loadcase
```

where two individual load cases are established in line 58, and for each load case, the type of temperature load load is defined with help of command pairs `tempr_type_lc_id` and `temp_load_type`. The all load cases are composed just from force load, i.e. no temperature load is defined by in lines 61 and 64. Should be noted that only the first load case is proportional, i.e. scaled by the gradually increasing load coefficient λ while the second load case remains constant until the end of computation. At the beginning of computation, the value of load coefficient is zero and the initial increment of load coefficient is calculated with help of Eq. 13.1. During the arclength procedure, the values of load coefficient increment are calculated automatically according to the progress of iteration keeping the range of the arc length $[10^{-12};10^{-5}]$. The load stepping is stopped if the maximum number of load steps is attained or the solver cannot reduce the step length which has been already set to minimum value defined in line 42 or the total length of arc is being attained (line 47).

13.5 Preprocessor file - section `mater`

This section contains the list of material models and their parameters. In this example, the distribution of material properties is assumed to be homogeneous where the initial loading is defined by the linear elastic isotropic material while the crack propagation is defined according to scalar damage model. It leads to the definition of two material types in this section whose content follows section has the following content:

```
67 begsec_mater
```

```
68 num_mat_types 2
69 # elastic isotropic material
70 mattype elisomat   num_inst 1
71 1 e 10.0e9 nu 0.25
72 # Scalar damage model
73 # Given: Gf = 9 [N/m]
74 # Estimated ft = 0.40 MPa => wf=Gf/ft=2.25e-05
75 mattype scaldamage   num_inst 1
76 1 ft 4.0e5 uf 2.25e-05 norm_type normazar
77   cor_dis_energy corr_on gsra ni 50 err 1.0e-2
78 endsec_mater
```

In the section, two material types (elastic isotropic and scalar damage) are being defined (line 68) and sequential lines contains the specification of these material types. The line 70 defines that the material type is elastic isotropic with one instance of material parameter set. Line 71 defines the first instance of material parameter set of elastic isotropic material which requires two parameters - Young's modulus (10 GPa) and Poisson's ratio (0.25). The second material model is represented by a scalar isotropic damage model with one instance of parameter set (line 75). The parameter set is defined on the last line where the tensile strength f_t=400 kPa is defined, the softening parameter w_f is calculated from the fracture energy G_f=9 N/m as $w_f = G_f/f_t = 2.25 \times 10^{-5}$ m, the Mazars' equivalent strain norm is assumed in the model definition. Additionally, the correction of dissipated energy with respect to size of mesh elements is switched on and in such case, the setup of iteration algorithm must be given at the end of material parameter record (line 77). The type of algorithm is `gsra` where the maximum number of iterations is set to 50 and the error of damage parameter residual is set to be 10^{-2}.

13.6 Preprocessor file - section `crsec`

This section contains the list of cross sections and their parameters. In this example, the cross section is given by the thickness 0.04 m which is uniform across the beam. The section has the following content:

```
80 begsec_crsec
81 num_crsec_types    1
82 crstype csplanestr num_inst 1
83 1 0.04
84 endsec_crsec
```

In the section, only one cross section type (for plane elements) is being defined (line 69) and sequential lines contains the specification of this one cross section type. The line 70 defines that the cross section type is for plane stress/strain elements (`csplanestr`) with one instance of cross section parameter set. The last line 71 defines the first instance of cross section parameter set for plane elements which requires just one parameters - thickness (0.04 m).

13.7 Preprocessor file - number of nodal DOFs

The section `nodsurfpr` defines common properties for group of nodes involved in the surface with specific property id. In 2D problems, the most common use of this section is for the specification of number of DOFs at nodes. In this example, two DOFs are defined in all nodes of the mesh. The section content is listed below:

```
86 begsec_nodsurfpr
87 ndofn 2 propid 1
88 endsec_nodsurfpr
```

where the command `ndofn` in line 87 defines two DOFs at all nodes with surface property id 1, i.e. on the whole domain solved.

13.8 Preprocessor file - Dirichlet's boundary conditions

Dirichlet's boundary condition prescribes values of primary unknowns defined in the problem solved and they can be prescribed with help of `bocon` command. In this example, this type of boundary conditions are represented by fixed node on the left end point of beam axis and vertically fixed node on right end point of the beam axis. These nodes are marked by the vertex property identifiers 6 and 8 respectively and therefore the `bocon` command should be placed in the `nodvertpr` section whose content is listed below

```
90 begsec_nodvertpr
91 # fixed nodes in both directions
92 bocon propid 6 num_bc 2 dir 1 cond 0.0 dir 2 cond 0.0
93 # nodes fixed in vertical direction
94 bocon propid 8 num_bc 1 dir 2 cond 0.0
```

```
97 endsec_nodvertpr
```

where displacements are prescribed to be zero values for all DOFs (defined in the previous section) of node with vertex property id 6 (line 92) while for node with property id 8, only the vertical (2nd) DOF is prescribed to be zero (line 94).

13.9 Preprocessor file - simulation of rigid plate

Another type of nodal boundary condition represents the simulation of rigid plate for the applied load transfer which can be simulated with help of coupled vertical DOFs at narrow area around the middle of beam. These nodes has got assigned the edge property id 5 in the generator and therefore the `dof_coupl` command providing DOF coupling of selected nodes should be placed in `nodedgpr` section

```
99 begsec_nodedgpr
100 # rigid plate <=> coupled DOFs on edge 5
```

```
101 dof_coupl propid 5 ndir 1 dir 2
102 endsec_nodedgpr
```

where line 101 represents the suitable record of the preprocessor command coupling vertical DOFs of all nodes at edge with property id 5 to the single DOF.

13.10 Preprocessor file - proportional load

The proportional load is represented by vertical force applied in the middle of beam on the top edge. The node has got assigned vertex property id 5 by the mesh generator and thus the `nod_load` command should be placed in the `nodvertpr` section.

```
90 begsec_nodvertpr
```

```
95 # vertical load
96 nod_load propid 5 lc_id 1 load_comp 0.0 -1.0
97 endsec_nodvertpr
```

In the command in line 96, load case id must be set 1 in order to be load scaled gradually by the variable load coefficient. In this case, the basic magnitude of vertical load component is set to -1.0 and thus the resulting load coefficient obtained at the end of computation represents the limit vertical force in N.

13.11 Preprocessor file - element type, material model, cross section

The FE type and material model are essential properties of elements which must be given in all kinds of problems. In this example, all elements have the same FE type and material model and therefore the most simple way how to assign them to all elements is the use of corresponding commands in the element section `elsurfpr` keeping in mind that the surface property id 1 is the same for all elements. The element type can be assigned by the command `el_type` while material model by the command `el_mat`.

```
107 # material model assigning
108 el_mat  propid 1 num_mat 2 type scaldamage type_id 1
109                            type elisomat type_id 1
110 # thickness of the specimen 40 mm
111 el_crsec  propid 1  type csplanestr  type_id 1
```

```
116 endsec_elsurfpr
```

where line 106 assigns triangle element with linear shape functions. The same material type of scalar damage model is assumed to be on all elements and it is assigned by the command in lines 108 and 109. The model is composed from two independent parts (keyword `num_mat`) one for scalar damage (`scaldamage` - line 108) and one for isotropic elasticity (`elisomat` - line 109). Both models refers to the first instance of material parameter set with help of keywords `type_id`. Cross section type and parameters are defined in line 111 with help of command `el_crsec` which refers to the first instance of cross section parameters defined above in the file in section `crsec` (line 83).

13.12 Preprocessor file - constant load

In the second load case, the beam is loaded by dead weight load which must be applied to all elements in the mesh. It can be achieved by the command `volume_load` placed in the section `elsurfpr` because all elements has got assigned the same surface property id 1. The syntax of the command is listed below

```
104 begsec_elsurfpr
```

```
112 # volume load
113 volume_load  propid 1  lc_id 2  ncomp 2
114              func_type stat coord_sys 1
115              load_comp 0.0 -12.0e3
116 endsec_elsurfpr
```

where the line 113 defines volume load on all elements with surface property id 1, the load is applied in the load case 2 which is kept constant for the whole computation procedure. The command continues in line 114 where the load is defined to be with constant distribution (keyword `func_type`), applied in the global coordinate system (keyword `coord_sys`) and finally, two load components are given in line 115 - the dead weight load 12 kN/m^3 is applied in the vertical direction.

13.13 Setup of the result output

The last section that has to be specified is represented by section `outdrv` where the output of results from MEFEL should be configured. More details about this section can be found in [10]. The section is composed from three parts dealing with different forms of result output. The first part controls output to the file in the text form has the following content:

```
begsec_outdrv
#-------------------------------
#  Definition of MEFEL output  |
#-------------------------------

# description of output to the text file
textout 0
```

```
endsec_outdrv
```

In this example, no text output of results is required (line 124).

The second part controls output in the various formats used in graphic postprocessor tools. In this example, the GiD format is required which allows for the most advanced configuration of the output. The part configuring this output is listed below:

```
118 begsec_outdrv
```

```
126 # description of output to the text file
127 # GiD format - one huge file
128 outgr_format grfmt_gid
129
130 # graphics output file name without extension
131 damage-beam
132
133 sel_nodstep  sel_all
134 sel_nodlc    sel_all
135
136 displ_nodes  sel_all   displ_comp   sel_all
137
138 strain_nodes sel_no
139 stress_nodes sel_no
140 other_nodes  sel_no
141
142 force_nodes sel_all  force_comp sel_all
143
144 sel_elemstep sel_all
145 sel_elemlc   sel_all
146
147 strain_elems sel_all   elemstrain_comp sel_mtx
       elemstra_transfid 0
148 stress_elems sel_all   elemstress_comp sel_mtx
       elemstre_transfid 0
149 other_elems       sel_prop numprop 1 prop 1 ent gregion
150 elemother_comp    sel_list numlist_items 1 2
```

```
166 endsec_outdrv
```

Line 128 defines the format used for the result output with help of keyword `outgr_format` whose value is set to `gid`. This results into one GiD file with all result quantities (`damage-beam.res`) that will be specified later in this part and another file with the mesh description (`damage-beam.msh`). The common GiD file name is given in line 131 to which the corresponding suffix will be added automatically.

After that the time/load steps and load case numbers must be given at which the nodal results will be printed out. In this case, all load steps are selected (line 133) and all load cases are selected (line 134). Should be noted that load cases cannot be selected independently because of nonlinearity of the problem and results are calculated for sum of proportional (1st) load case and constant (2nd) load case and thus load case selection should be just `sel_all` defined for keyword `sel_nodlc`. Having the time/load steps and load cases specified, the print configuration of particular nodal quantities follows providing the selection of nodes for each quantity, where the given quantity will be printed out, followed by the selection of the given quantity components. Thus line 136 specifies that for all nodes, all displacement components will be printed out while the line 138 selects no nodes (keyword value `sel_no`) for nodal strains, i.e. no nodal strains will be printed. The same option is specified for nodal stresses (line 139), and nodal other values (line 140) and therefore they do not be printed too. Line 141 defines that at all nodes all components of internal force vector will be printed.

Configuration of nodal values output is followed by the similar configuration of element values output performed in all integration points on the selected elements. It starts with selection of time/load steps (line 144) and load cases (line 145). Using the same keyword values as for nodes results to the selection of all time/load steps and all load cases. Line 147 specifies that for all elements (keyword `strain_elems`), the output of all strain components (keyword `elemstrain_comp`) will be performed with no transformation of components (keyword `elemstra_transfid`). Line 148 specifies that for all elements (keyword `stress_elems`), the output of all stress components (keyword `elemstress_comp`) will be performed with no transformation of components (keyword `elemstre_transfid`). Selection of all strain and stress components on elements is specified by `sel_mtx` optional value that provides the output of strains and stresses in the tensorial form (all their components) which allows for better postprocessing in the GiD (calculation of principal values and vectors). A different way of element selection with help of their property id that is used for `other` value output. The line 149 defines that the output of `other` values will be carried out for all elements with region property 1. The output will be configured to print only one other value which is represented by the damage parameter ω. The parameter ω is stored in the damage model in the other array as the 2nd component. In this case, the list selection type `sel_list` is used in `other` array component selection and this list contains only one item (`numlist_items 1`) - see line 150. The list of item identifiers stands at the end of record and in this case, it contains just number 2, i.e. reference to the 2nd component of `other` array. Should be noted that the order of internal variables of the `other` array can be arbitrary and it is defined inside the given material model. In this example, the scalar damage model is used which is contained in the `scaldam.cpp` source file where the order of quantities in the `other` array can be found.

The last part controls output of selected quantities in particular time/load steps which can be used for creation of diagrams. It will be suitable to print the value of vertical displacement in the middle of beam and the value of load coefficient in all load steps. The content of last part of `outdrv` section is listed below:

```
118 begsec_outdrv
```

```
152 # text output of graphs
153 # number of created files with diagrams
154 numdiag 1
155 damage-beam.dat
156 numunknowns 2 # number of printed unknowns
```

```
157 sel_diagstep sel_all # type of time step selection = all time steps
158
159 # setup of the first column
160 point atnode  node 4 # node number 4 (applied force)
161 quant_type pr_displ  compid 2 # vertical component of displacement
162
163 # setup of the second column
164 point atnode   node 4 # node number 4
165 quant_type pr_appload # unknown type = time
166 endsec_outdrv
```

where the line 154 defines that the number of diagram files created is 1. The name of the output diagram file follows on the next line 155. For each diagram file, the number of printed quantities must be given (line 156) and this number equals to the number of columns in the table created in the diagram file. The values of selected quantities are printed to the file in each load step selected. All load steps are selected in this case in line 157.

After the above initial setup, the definition of particular quantities follows. Each record contains definition of point at which the given quantity should be printed out followed by the type of quantity. The first quantity record starts at line 160 where the position at node 4 is given and the first quantity type is given in line 161 which defines the type of quantity to be displacement (`pr_displ`) where the vertical displacement component is selected (`compid 2`). The second quantity record starts at line 164 where the position at node 4 is given again and the quantity type is given in line 165 which defines the type to be load coefficient `pr_appload`.

13.14 Preprocessor file

This section contains listing of the whole preprocessor file.

```
#
# Run with: mechprep damage-beam.pr damage-beam.in
#
begsec_files
damage-beam.top
mesh_format sifel
edge_numbering 1
read_mat_strings no
read_mat_kwd yes
endsec_files

begsec_probdesc
Three point bending test of mortar MPA35 beam 40x40x160 mm
```

```
mespr 1
problemtype mat_nonlinear_statics

straincomp 1
strainpos  1
strainaver 0
stresscomp 1
stresspos  1
stressaver 0
othercomp  1
otherpos   1
otheraver  0
reactcomp  1

adaptivity      0
stochasticcalc  0
homogenization  0
noderenumber    0

type_of_nonlin_solver  arclg
stiffmat_type          ijth_tangent_stiff
1
10
lambda_determination minangle
al_num_steps         300
al_num_iter          50
al_error             1.0e-3
al_init_length       1.0e-6
al_min_length        1.0e-12
al_max_length        1.0e-5
al_psi               0.0
check_div            off
check_total_arcl     on
max_total_arc_length 0.0003
check_req_lambda     off
al_displ_contr_type  nodesdistincr
probdimal            2
11 15
hdbackup             nohdb
stiffmatstor         spdirect_stor_scr
typelinsol           spdirldl
endsec_probdesc

begsec_loadcase
num_loadcases  2
```

```
#temperature load type for the first load case
lc_id   1
temp_load_type     0
#temperature load type for the second load case
lc_id   2
temp_load_type     0
endsec_loadcase

begsec_mater
num_mat_types 2
# elastic isotropic material
mattype elisomat   num_inst 1
1 e 10.0e9 nu 0.25
# Scalar damage model
# Given: Gf = 9 [N/m]
# Estimated ft = 0.40 MPa => wf=Gf/ft=2.25e-05
mattype scaldamage   num_inst 1
1 ft 4.0e5 uf 2.25e-05 norm_type normazar
  cor_dis_energy corr_on gsra ni 50 err 1.0e-2
endsec_mater

begsec_crsec
num_crsec_types    1
crstype csplanestr num_inst 1
1 0.04
endsec_crsec

begsec_nodsurfpr
ndofn 2 propid 1
endsec_nodsurfpr

begsec_nodvertpr
# fixed nodes in both directions
bocon propid 6 num_bc 2 dir 1 cond 0.0 dir 2 cond 0.0
# nodes fixed in vertical direction
bocon propid 8 num_bc 1 dir 2 cond 0.0
# vertical load
nod_load propid 5 lc_id 1 load_comp 0.0 -1.0
endsec_nodvertpr

begsec_nodedgpr
# rigid plate <=> coupled DOFs on edge 5
dof_coupl propid 5 ndir 1 dir 2
endsec_nodedgpr
```

```
begsec_elsurfpr
# element type specification
el_type  propid 1  planeelementlt  strastrestate planestress
# material model assigning
el_mat  propid 1 num_mat 2 type scaldamage type_id 1
                           type elisomat type_id 1
# thickness of the specimen 40 mm
el_crsec  propid 1  type csplanestr  type_id 1
# volume load
volume_load  propid 1  lc_id 2  ncomp 2
             func_type stat coord_sys 1
             load_comp 0.0 -12.0e3
endsec_elsurfpr

begsec_outdrv
#--------------------------------
#  Definition of MEFEL output  |
#--------------------------------

# description of output to the text file
textout 0

# description of output to the text file
# GiD format - one huge file
outgr_format grfmt_gid

# graphics output file name without extension
damage-beam

sel_nodstep  sel_all
sel_nodlc    sel_all

displ_nodes  sel_all   displ_comp   sel_all

strain_nodes sel_no
stress_nodes sel_no
other_nodes  sel_no

force_nodes sel_all  force_comp sel_all

sel_elemstep sel_all
sel_elemlc   sel_all

strain_elems sel_all   elemstrain_comp sel_mtx
   elemstra_transfid 0
```

```
stress_elems sel_all   elemstress_comp sel_mtx
   elemstre_transfid 0
other_elems       sel_prop numprop 1 prop 1 ent gregion
elemother_comp    sel_list numlist_items 1 2

# text output of graphs
# number of created files with diagrams
numdiag 1
damage-beam.dat
numunknowns 2 # number of printed unknowns
sel_diagstep sel_all # type of time step selection = all
   time steps

# setup of the first column
point atnode  node 4 # node number 4 (applied force)
quant_type pr_displ  compid 2 # vertical component of
   displacement

# setup of the second column
point atnode   node 4 # node number 4
quant_type pr_appload # unknown type = time
endsec_outdrv
```

Chapter 14

Time dependent problem - visco-plastic model

This section describes how to prepare MEFEL input file with the help of MECHPREP for the time dependent problem with nonlinear visco-plastic material. The problem involves a bar composed from two parts with the different cross section area subjected by load which is variable in course of time. The nonlinearity is induced by the material model of visco-plasticity which is handled by Perzyna's approach [13] where J2 flow criterion is being adopted. One part of the bar has length of 500 mm and cross section area A_1=201.1 mm^2, the other part has length of 400 mm and cross section area A_2=113.1 mm^2. The settings of the example is depicted in Fig. 14.1.

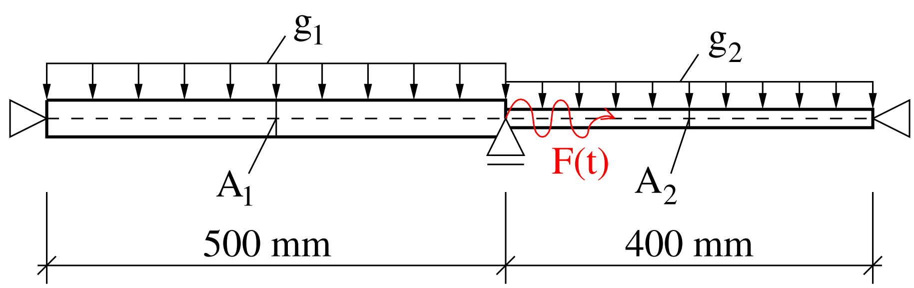


Figure 14.1: Settings of a bar with the visco-plastic model

The loading consists of two load cases that are assumed to influence the structure simultaneously:

1. defines static dead weight load given by values g_1 and g_2
2. defines load $F(t)$ whose components are prescribed by suitable time functions; in this case, the horizontal component is defined by time function whose diagram is given in Fig. 14.2 and the vertical component is 0.

There are two possibilities how to define load cases. In the first approach, the load is defined with the help of subloadcases that are involved in one main load case. The definition of subloadcases is the same as in the linear or nonlinear statics problems but

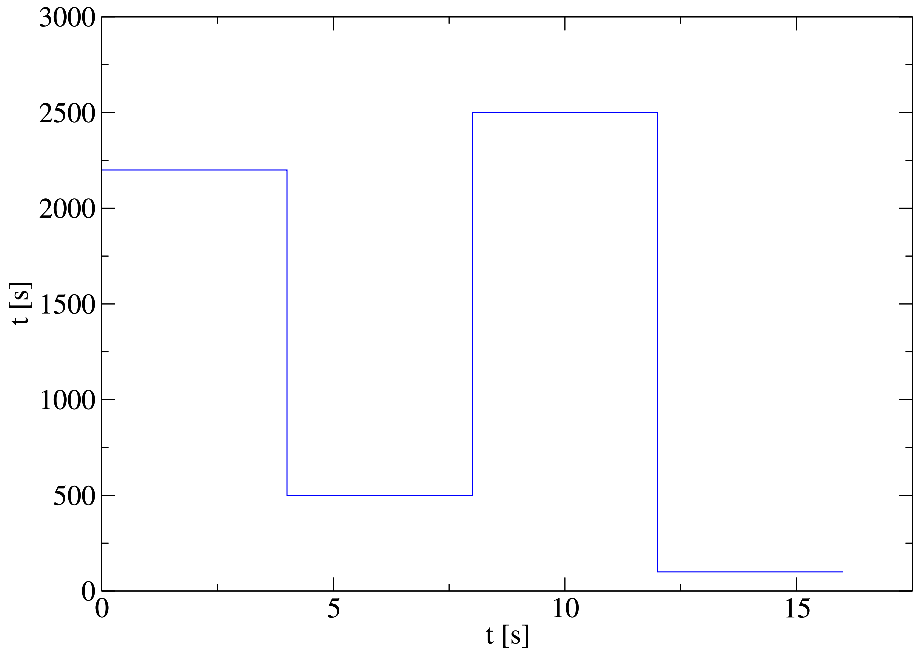


Figure 14.2: Diagram of time dependent force F(t).

the time function for load coefficient of every subloadcase must be given. Each load component of the given subloadcase is multiplied by the actual value of load coefficient in course of the time stepping. In the second approach, the definition of load in the given load case is more general - every load component must be specified as an independent time function. In both approaches, the same results can be attained but they result in different size of load section of the MEFEL input file depending on the problem solved. The type of approach is selected by the appropriate value of `num_sublc`. For the subloadcase approach, a nonzero value must be given after the keyword `num_sublc` while the zero must be specified for load components defined by independent time functions.

The material of the bar is assumed with following parameters: Young modulus E=2 GPa, Poisson's ratio ν=0.35, yield stress of part 1 f_{s1}=10.06 MPa, yield stress of part 2 f_{s2}=2.55 MPa and coefficient of viscosity η=2.5×10^{-9} Pa·s.

14.1 Topology file

In the first step, a mesh file must be created and corresponding property identifiers must be defined in order to MECHPREP can work. The mesh file can be created manually in editor with respect to the simple topology. More details about the `sifel` topology format can be found in section 4.2. Should be noted that property numbering scheme exploits just vertex property identifiers at nodes, for the assignment of supports and nodal force, and volume properties on elements for the assignment of material parameters. It should be noted that node 2 forms interface between elements and thus consistently, it should involve volume property identifiers from both elements. The topology file `bar-viscopl.top` (including comments of the SIFEL format) is listed below

```
#
# section of nodes
#

3 # number of nodes

# node_id, coord_x, coord_y, coord_z, num_prop,
# {prop_ent  prop_ent_num} x numprop
1  0.0 0.0 0.0  2  1 1  4 1
2  0.5 0.0 0.0  3  1 2  4 1  4 2
3  0.9 0.0 0.0  2  1 1  4 2

#
# section of elements
#

2 #number of elements

# elem_id, elem_type=bar, node_1, node_2, volume_prop_id
1  1  1 2  1
2  1  2 3  2
```

The file `bar-viscopl.top` can be displayed with the help of MeshEditor tool where it is possible to visualize the property identifiers by various colors as in Fig. 14.3.

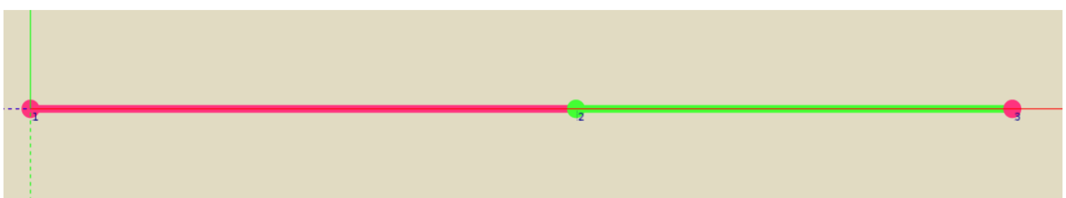


Figure 14.3: Simple bar structure - mesh with visualized property identifiers

14.2 Preprocessor file - section `files`

Having the mesh file with property identifiers created, a preprocessor file should be prepared. Preprocessor file is composed from the several sections that may be organized arbitrarily in the file but they will be introduced in the order of the real processing in MECHPREP. Each preprocessor file must contain section `files` which in this case has the following contents:

```
begsec_files
bar-viscopl.top
```

```
3 mesh_format       sifel
4 edge_numbering    0
5 read_mat_strings no
6 read_mat_kwd      yes
7 read_crs_strings no
8 read_crs_kwd      yes
9 endsec_files
```

where the topology input file is being specified to be `bar-viscopl.top` in line 2 then the format of the topology file is given in line 3 and finally, it is given that edge and surface property identifiers are not given on elements - line 4.

It is supposed that materials and cross sections will be given in the preprocessor file directly and thus no file names of material and cross section files are given. More details about this preprocessor section can be found in 9.1. Additionally, the material parameters will not be processed as strings but by particular material model read procedures (line 8) and keywords will be used in the parameter definition (line 9). Additionally, the same setup for the cross section parameter processing is given in lines 10 and 11. Cross section parameters will not be processed as strings but by particular cross section read procedures (line 10) and keywords will be used in the parameter definition (line 11).

14.3 Preprocessor file - section `probdesc`

This section is used by MEFEL to identify the type of problem solved, the type of nonlinear and linear equation system solver and some other calculation setup. Some details about the particular cases of `probdesc` setup can be found in [10]. In this case, the following setup is chosen:

```
begsec_probdesc
Simple bar structure with viscoplasticity model
mespr 1        # detail output
problemtype    mech_timedependent_prob

straincomp 0  # no explicit strain computation
stresscomp 0  # no explicit stress computation
othercomp  0  # no explicit internal variable computation
reactcomp  0  # no explicit reaction computation

adaptivity      0  # adaptivity is not used
stochasticcalc 0  # deterministic computation
homogenization 0  # homogenization is not applied
noderenumber    no_renumbering  # nodes are not renumbered

time_contr_type fixed # time steps with fixed length
start_time       0.0
end_time         15.5
```

```
33 num_imp_times   0 # the number of important times
34 funct_type      stat
35 const_val       0.01 # initial time step length
36
37 timetypeprin    seconds
38 hdbackup        nohdb
39
40 nr_num_iter     10
41 nr_error        1.0e-6
42 check_div       off
43
44 stiffmatstor   skyline_matrix # skyline storage of system matrix
45 stiffmat_type  initial_stiff
46 typelinsol     ldl            # solution by LDL decomposition
47 endsec_probdesc
```

In this section, the title of the problem solved is given in line 16, detailed message printing is switched on (line 17) and material time dependent problem type with negligible inertial forces is specified to be solved in line 18.

In the sequential two blocks of commands, the explicit strain calculation is off (line 20), the same setup for explicit stress computation is defined in line 21 and for internal (other) variables in lines 22. Finally, the explicit calculation of reactions is not required in line 23. Should be noted the above setup may ignored with respect to problem type which requires calculation of strains, stresses and internal variables at integration points. Additionally, output of reactions will be required in the outdriver section and therefor they will be calculated before the call of output procedure. All additional advanced techniques such as mesh adaptivity (line 25), stochastic calculations (line 25), homogenization techniques (line 37) and node renumbering (line 28) are not taken into account in the computation.

Remaining blocks of commands defines the time stepping and the type of solver of system of nonlinear algebraic equations. Line 30 defines that the time stepping will be proceeded with the fixed time step length, with start time 0.0 (line 31) and finish time 15.5 (line 32). There are no important times in which the calculation should be performed (line 33). The time step length is given by gfunct type record (see 7) which can define step length with respect to attained time. In this case, constant type of time function is given in line 34 and the fixed time step length of 0.01 s is defined in line 35. Basically, all time values should be specified in seconds in the input file but resulting time units at output files can be defined by keyword `timetypeprin`. In this case, there is not required conversion of time values on the output and therefor the keyword value is seconds in line 37. Backup of particular load steps on harddisk is not performed (line 38).

The block with the setup of Newton-Raphson iteration procedure for nonlinear calculation follows. Line 40 defines the maximum number of iterations performed in the equilibrium search. Required residual norm follows in line 41 and no iteration divergency will be checked (line 42). The last block define that skyline_matrix storage of system matrix is used (line 44), the initial stiffness matrix approach will be employed (line 45) and the equation system will be solved with the help of LDL solver (line 46). It is possible to apply this setup in this case because the system matrix is symmetric and positive definite for the initial matrix defined in the visco-plastic model.

14.4 Preprocessor file - section `loadcase`

This section defines the number of load cases and some details about the content of particular load cases. According to the problem setting, this section looks as follows:

```
begsec_loadcase
num_loadcases  1

# the subloadcase approach for the load definition is selected, i.e.
# the number of subloadcases is nonzero
lc_id 1    num_sublc 2 # the main load case 1 involves 2 subloadcases

# load coefficient of the 1. subloadcase
tfunc_lc_id   1
tfunc_slc_id  1
funct_type    tab    # type of general function - table
approx_type   linear # piecewise linear interpolation
ntab_items    8      # the number of rows in table
# {time, load_coef_value} x 8
-1.0 2.2e3
 4.0 2.2e3
 4.0 5.0e2
 8.0 5.0e2
 8.0 2.5e3
12.0 2.5e3
12.0 1.0e2
16.0 1.0e2
# load coefficient of the 2. subloadcase
tfunc_lc_id    1
tfunc_slc_id   2
funct_type     stat  # type of general function - constant
const_val      1.0   # constant value

# temperature load type for the first load case
#
# the first subloadcase
tempr_type_lc_id   1
tempr_type_slc_id  1
temp_load_type     0
```

```
84 # the second subloadcase
85 tempr_type_lc_id   1
86 tempr_type_slc_id  2
87 temp_load_type     0
88 endsec_loadcase
```

where where one load case is established in line 51, and for each load case, the number of subloadcases must be specified. For each subloadcase, the time function for load coefficient magnitude and type of temperature load must be given. The load coefficient magnitude is given by command triples `tfunc_lc_id`, `tfunc_slc_id` and `gfunct` while temprature load is given by command triples `tempr_type_lc_id`, `tempr_type_slc_id` and `temp_load_type` for each subloadcase. All load cases are composed just from force load, i.e. no temperature load is defined in lines 81–83 and 85–87. Only the first subloadcase is fully time dependent, i.e. scaled by the piece-wise constant function defined by `gfunct` record in lines 58–71, while the second subloadcase is intended for constant dead weight load which remains constant until the end of computation (lines 73–76).

14.5 Preprocessor file - section `mater`

This section contains the list of material models and their parameters. In this example, the distribution of material properties is assumed to be different for both bar elements, the material of element 1 has yield stress f_{s1}=10.06 MPa while yield stress of element 2 is f_{s2}=2.55 MPa. It is assumed to visco-plastic material model for both elements and this material type requires to specify one model of viscous behaviour and other model for plasticity. Furthermore, each plasticity model requires definition of elastic behaviour. Therefore four material types must be defined in this section whose content is listed below

```
91 begsec_mater
92 num_mat_types 4
93 # elastic isotropic material
94 mattype elisomat num_inst 1
95 1 e 2.0e9 nu 0.35
96 # simple 1D plasticity yield condition
97 mattype simplas1d num_inst 2
98 1 fs 1.06e7 k 0.0  nostressretalg  nohs
99 2 fs 2.55e6 k 0.0  nostressretalg  nohs
100 # simple viscous material
101 mattype simvisc num_inst 1
102 1 eta 2.5e-9
103 # artificial material for combination of visco-plasticity
104 mattype viscoplasticity num_inst 1
105 1 # there are no parameters of visco-plasticity material
106 endsec_mater
```

In the section, four material types (elastic isotropic, simple 1D plasticity, simple viscosity and visco-plasticity) are being defined (line 92) and sequential lines contains the specification of these material types. The line 94 defines that the material type is elastic isotropic with one instance of material parameter set. Line 95 defines the first instance of material parameter set of elastic isotropic material which requires two parameters - Young's modulus (2 GPa) and Poisson's ratio (0.35). The second material model is represented by simple 1D plasticity model with two instances of parameter sets (line 97). The parameter sets are defined in lines 98 and 99 where the yield stress is set to f_{s1}=10.6 MPa and f_{s2}=2.55 MPa respectively. In both cases, hardening modulus is 0, i.e. perfect plasticity with no hardening is to be solved. Additionally, the setup of stress return algorithm and setup of hardening modulus evolution must be given at the end of material parameter record. In this case, no stress return algorithm is required because visco-plastic is integrated explicitly and no hardening setup is required with respect to zero hardening modulus. One instance of model of simple viscosity model is defined in line 101 for which the viscosity coefficient η=2.5×10^{-9} Pa·s is being defined in line 102. The last material model type is represented by artificial material model for combination of viscous and plasticity material models whose one instance is defined in line 104. The last line 105 of the section defines material parameters for visco-plastic artificial material which require no parameters.

14.6 Preprocessor file - section `crsec`

This section contains the list of cross sections and their parameters. In this example, the cross section is given by the cross section area A_1=201.1×10^{-6} m^2 for the bar element 1 and cross section area A_2=113.1×10^{-6} m^2 of the element 2. The section has the following content:

```
109 begsec_crsec
110 num_crsec_types 1
111 crstype csbar2d num_inst 2
112 1 a 201.1e-6
113 2 a 113.1e-6
114 endsec_crsec
```

In the section, only one cross section type (for 2D bar elements) is being defined (line 110) and sequential lines contains the specification of instances of the cross section type. The line 111 defines that the cross section type is for plane bar elements (`csbar2d`) with two instances of cross section parameter set. The last two lines 112 and 113 defines cross section parameter sets for these instances which require just one parameter - cross section area.

14.7 Preprocessor file - number of nodal DOFs

The section `nodvolpr` defines common properties for group of nodes involved in the volume/region with specific property id. The most common use of this section is for the specification of number of DOFs at nodes. In this example, two DOFs are defined in all nodes of the mesh. The section content is listed below:

```
127 begsec_nodvolpr
128 ndofn 2 propid 1  # number of degrees of freedom at nodes
129 ndofn 2 propid 2  # number of degrees of freedom at nodes
130 endsec_nodvolpr
```

where the command `ndofn` in line 89 defines two DOFs at all nodes with surface property id 1, i.e. on the whole domain solved.

14.8 Preprocessor file - Dirichlet's boundary conditions

Dirichlet's boundary condition prescribes values of primary unknowns defined in the problem solved and they can be prescribed with help of `bocon` command. In this example, this type of boundary conditions are represented by fixed node on the left and right end points of bar structure and vertically fixed node in the middle. These nodes are marked by the vertex property identifiers 1 and 2 respectively and therefore the `bocon` command should be placed in the `nodvertpr` section whose content is listed below

```
118 begsec_nodvertpr
119 bocon  propid 1  num_bc 2  dir 1  cond 0.0  dir 2 cond 0.0 #
       left and right supports
120 bocon  propid 2  num_bc 1  dir 2  cond 0.0  # middle support
```

```
123 endsec_nodvertpr
```

where displacements are prescribed to be zero values for all DOFs (defined in the previous section) of node with vertex property id 1 (line 119) while for node with property id 2, only the vertical (2nd) DOF is prescribed to be zero (line 120).

14.9 Preprocessor file - nodal time dependent load

The time dependent load is represented by horizontal force F(t) applied in the middle node of bar structure. The node has got assigned vertex property id 2 and thus the `nod_load` command should be placed in the `nodvertpr` section.

```
118 begsec_nodvertpr
```

```
121 # time dependent force in the middle node
122 nod_load  propid 2  lc_id 1  slc_id 1 load_comp 1.0 0.0
123 endsec_nodvertpr
```

In the command in line 122, load case id must be set 1 and and subloadcase id must be set also to 1 in order to be load components scaled by the load coefficient defined in lines 60-71. In this case, the basic magnitude of the horizontal load component is set to 1.0 and thus the resulting load component is prescribed by load coefficient value directly.

14.10 Preprocessor file - element type, material model, cross section

The FE type and material model are essential properties of elements which must be given in all kinds of problems. In this example, all elements have the same FE type and material models differs not in types but in parameters only. Therefore the most simple way how to assign them to elements is the use of corresponding commands in the element section `elvolpr` keeping in mind that the volume/region property id 1 is assigned to element 1 and property id 2 is assigned to element 2. The element type can be assigned by the command `el_type` while material model by the command `el_mat`.

```
133 begsec_elvolpr
134 # element properties for thick bar
135 el_type  propid 1 bar2d
136 el_mat   propid 1 num_mat 4  type viscoplasticity type_id 1
137                              type simvisc         type_id 1
138                              type simplas1d       type_id 1
139                              type elisomat        type_id 1
140 el_crsec propid 1  type csbar2d  type_id 1
```

```
146 # element properties for narrow bar
147 el_type  propid 2 bar2d
148 el_mat   propid 2 num_mat 4  type viscoplasticity type_id 1
149                              type simvisc         type_id 1
150                              type simplas1d       type_id 2
151                              type elisomat        type_id 1
152 el_crsec propid 2  type csbar2d  type_id 2
```

```
157 endsec_elvolpr
```

where lines 135 and 147 assigns plane bar element with linear shape functions. The visco-plastic material model type is assumed to be on both elements. The model with higher yield stress 10.6 MPa is assigned to element 1 by the command in lines 136 and 139. The model is composed from four independent parts (keyword `num_mat`) one for visco-plastic model (`viscoplasticity` - line 136), one for model of viscosity (`simvisc` - line 137), the other for plasticity (`simplas1d` - line 138) and the last model for for elasticity (`elisomat` - line 139). All models refers to the first instance of material parameter set with help of keywords `type_id`. Cross section type and parameters for element 1 are defined in line 140 which refers to parameter set instance with larger cross section area.

The model with lower yield stress 2.55 MPa is assigned to element 2 by the command in lines 147 and 152. The model is composed from four independent parts (keyword `num_mat`) one for visco-plastic model (`viscoplasticity` - line 148), one for model of viscosity (`simvisc` - line 149), the other for plasticity (`simplas1d` - line 150) and the last model for for elasticity (`elisomat` - line 151). All models except of plasticity refers to the first instance of material parameter set with help of keywords `type_id`. The plasticity model refers to the second instance of parameter set where the lower yield stress is being defined. Cross section type and parameters for element 2 are defined in line 152 which refers to parameter set instance with a smaller cross sectional area.

14.11 Preprocessor file - constant load

In the second subloadcase, the structure is loaded by dead weight load which must be applied to all elements in the mesh. It can be achieved by the command `volume_load` placed in the section `elvolpr` because all elements has got assigned the same volume property ids 1 and 2. Both commands should refer to the second subloadcase which is being defined as constant. The syntax of the command is listed below

```
133 begsec_elvolpr
```

```
141 # dead weight load defined as volumetric load
142 volume_load propid 1  lc_id 1        slc_id 2
143             ncomp  2  func_type stat  coord_sys 1
144             load_comp 0.0 -12.0e3
```

```
153 # dead weight load defined as volumetric load
154 volume_load propid 2  lc_id 1        slc_id 2
155             ncomp  2  func_type stat  coord_sys 1
156             load_comp 0.0 -12.0e3
157 endsec_elvolpr
```

where the lines 142–144 defines volume load on all elements with volume/region property id 1, the load is applied in the load case 1 and subloadcase 2 which is kept constant for the whole computation procedure. The same command is being applied in lines 154–156 but for elements with property id 2. In both cases, two load components are given (keyword `ncomp`), the load is defined to be with constant distribution (keyword `func_type`), applied in the global coordinate system (keyword `coord_sys`) and finally, the dead weight load 12 kN/m^3 is applied in the vertical direction (keyword `load_comp`).

14.12 Setup of the result output

The last section that has to be specified is represented by section `outdrv` where the output of results from MEFEL should be configured. More details about this section can be found in [10]. The section is composed from three parts dealing with different forms of result output. The first part controls output to the file in the text form has the following content:

```
160 begsec_outdrv
161 #--------------------------------
162 #  Definition of MEFEL output   |
163 #--------------------------------
164 
```

```
165 # description of output to the text file
166 textout on
167 bar-viscopl-slc.out
168 sel_nodstep  sel_all
169 sel_nodlc    sel_all
170 displ_nodes  sel_all  displ_comp sel_all
171 strain_nodes sel_no
172 stress_nodes sel_no
173 other_nodes  sel_no
174 reactions    1
175 
176 sel_elemstep sel_all
177 sel_elemlc   sel_all
178 strain_elems sel_all  elemstrain_comp sel_all
       elemstra_transfid 0
179 stress_elems sel_all  elemstress_comp sel_all
       elemstre_transfid 0
180 other_elems  sel_all  elemother_comp  sel_all
181 
182 sel_pointstep sel_no
```

```
232 endsec_outdrv
```

The text output is switched on by the command on line 166. The results in plain format will be printed to the text file whose name is given in line 167 (`bar-viscopl-slc.out`).

After that the time/load steps and load case numbers must be given at which the nodal results will be printed out. In this case, all load steps are selected (line 168) and all load cases are selected (line 169). Should be noted that subloadcases cannot be selected independently because of nonlinearity of the problem and results are calculated for sum of all subloadcases and thus load case selection should be just `sel_all` defined for keyword `sel_nodlc`. Having the time/load steps and load cases specified, the print configuration of particular nodal quantities follows providing the selection of nodes for each quantity, where the given quantity will be printed out, followed by the selection of the given quantity components. Thus line 170 specifies that for all nodes, all displacement components will be printed out while the line 171 selects no nodes (keyword value `sel_no`) for nodal strains, i.e. no nodal strains will be printed. The same option is specified for nodal stresses (line 172), and nodal `other` values (line 173) and therefore they do not be printed too. Line 174 defines that at all nodes all reaction components will be printed.

Configuration of nodal values output is followed by the similar configuration of element values output performed in all integration points on the selected elements. It starts with selection of time/load steps (line 176) and load cases (line 177). Using the same keyword values as for nodes results to the selection of all time/load steps and all load cases. Line 178 specifies that for all elements (keyword `strain_elems`), the output of all strain components (keyword `elemstrain_comp`) will be performed with no transformation of components (keyword `elemstra_transfid`). Line 179 specifies that for all elements (keyword `stress_elems`), the output of all stress components (keyword `elemstress_comp`) will be performed with no transformation of components (keyword `elemstre_transfid`). Selection of all strain and stress components on elements is specified by `sel_all` optional value that provides the output of strain and stress components as independent scalar values. The line 180 defines that the output of `other` values will be carried out for all elements and all internal variables will be printed (keyword `elemother_comp`). Finally, there is no output of quantities at user defined points on elements (line 182).

The second part controls output in the various formats used in graphic postprocessor tools. In this example, the GiD format is required which allows for the most advanced configuration of the output. The part configuring this output is listed below:

```
160 begsec_outdrv
```

```
184 # description of output to the text file
185 # 3 = GiD format - one huge file
186 outgr_format grfmt_gid
187
188 # graphics output file name without extension
189 bar-viscopl-slc
190
191 sel_nodstep  sel_all
192 sel_nodlc    sel_all
193 displ_nodes  sel_all  displ_comp sel_all
194 strain_nodes sel_no
195 stress_nodes sel_no
196 other_nodes  sel_no
197 force_nodes  sel_all  force_comp sel_all
198
199 sel_elemstep sel_all
200 sel_elemlc   sel_all
201 strain_elems sel_all  elemstrain_comp sel_all
       elemstra_transfid 0
202 stress_elems sel_all  elemstress_comp sel_all
       elemstre_transfid 0
203 other_elems  sel_all  elemother_comp  sel_all
```

```
232 endsec_outdrv
```

Line 186 defines the format used for the result output with help of keyword `outgr_format` whose value is set to `grfmt_gid`. This results into one GiD file with all result quantities (`bar-viscopl-slc.res`) that will be specified later in this part and another file with the mesh description (`bar-viscopl-slc.msh`). The common GiD file name is given in line 189 to which the corresponding suffix will be added automatically.

After that the time/load steps and load case numbers must be given at which the nodal results will be printed out. In this case, all load steps are selected (line 191) and all load cases are selected (line 192). Should be noted that subloadcases cannot be selected independently because of nonlinearity of the problem and results are calculated for sum of all subloadcases and thus load case selection should be just `sel_all` defined for keyword `sel_nodlc`. Having the time/load steps and load cases specified, the print configuration of particular nodal quantities follows providing the selection of nodes for each quantity, where the given quantity will be printed out, followed by the selection of the given quantity components. Thus line 193 specifies that for all nodes, all displacement components will be printed out while the line 194 selects no nodes (keyword value `sel_no`) for nodal strains, i.e. no nodal strains will be printed. The same option is specified for nodal stresses (line 195), and nodal `other` values (line 196) and therefore they do not be printed too. Line 197 defines that at all nodes all components of internal force vector will be printed.

Configuration of nodal values output is followed by the similar configuration of element values output performed in all integration points on the selected elements. It starts with selection of time/load steps (line 199) and load cases (line 200). Using the same keyword values as for nodes results to the selection of all time/load steps and all load cases. Line 201 specifies that for all elements (keyword `strain_elems`), the output of all strain components (keyword `elemstrain_comp`) will be performed with no transformation of components (keyword `elemstra_transfid`). Line 202 specifies that for all elements (keyword `stress_elems`), the output of all stress components (keyword `elemstress_comp`) will be performed with no transformation of components (keyword `elemstre_transfid`). Selection of all strain and stress components on elements is specified by `sel_all` optional value that provides the output of strain and stress components as independent scalar values. The line 203 defines that the output of `other` values will be carried out for all elements and all internal variables will be printed (keyword `elemother_comp`).

The last part controls output of selected quantities in particular time/load steps which can be used for creation of diagrams. It will be suitable to print the value of horizontal displacement and horizontal force component in the middle of bar structure and the value of actual time at all time steps. Additionally, attained stresses will be printed at integration points of both elements. The content of last part of `outdrv` section is listed below:

```
160 begsec_outdrv
```

```
205 # text output of graphs
206 # number of created files with diagrams
207 numdiag 1
208 bar-viscopl-slc.dat
209 numunknowns  5       # number of printed unknowns
210 sel_diagstep sel_all # type of load step selection = all load steps
211
212 # point type = node, node id = 2
213 point atnode node 2
214 # printed nodal quantity = time
215 quant_type pr_time
216 # point type = node, node id = 2
```

```
217 point atnode node 2
218 # printed nodal quantity = displacement, 1st component
219 quant_type pr_displ compid 1
220 # point type = node, node id = 2
221 point atnode node 2
222 # printed nodal quantity = force, 1st component
223 quant_type pr_forces compid 1
224 # point type = element ip, local integration point id = 2
225 point atip elem 1 ip 1
226 # printed element quantity = stress, 1st component
227 quant_type pr_stresses compid 1
228 # point type = element ip, local integration point id = 1
229 point atip elem 2 ip 1
230 # printed nodal quantity = stress, 1st component
231 quant_type pr_stresses compid 1
232 endsec_outdrv
```

where the line 207 defines that the number of diagram files created is 1. The name of the output diagram file follows on the next line 208. For each diagram file, the number of printed quantities must be given (line 209) and this number equals to the number of columns in the table created in the diagram file. The values of selected quantities are printed to the file in each load step selected. All load steps are selected in this case in line 210.

After the above initial setup, the definition of particular quantities follows. Each record contains definition of point at which the given quantity should be printed out followed by the type of quantity. The first quantity record starts at line 213 where the position at node 2 is given and the first quantity type is given in line 215 which defines the type of quantity to be actual time.

The second quantity record starts at line 217 where the position at node 2 is given and the second quantity type is given in line 219 which defines the type of quantity to be displacement (`pr_displ`) where the horizontal displacement component is selected (`compid 1`).

The third quantity record starts at line 221 where the position at node 2 is given again and the quantity type is given in line 223 which defines the type to be force `pr_forces` and its horizontal component is selected (keyword `compid`).

The fourth quantity record starts at line 225 where the position at the first integration point (keywords `atip`, `ip`) of element 1 (keyword `elem`) is being defined. The type of quantity is specified in line 227 which defines the type to be stress `pr_stresses` and its first component (keyword `compid`). The same record with different element number is defined for the fifth quantity in lines 229–231.

14.13 Preprocessor file

This section contains listing of the whole preprocessor file.

```
#
# Run with: mechprep bar-viscopl.pr bar-viscopl.in
#
begsec_files
bar-viscopl.top
mesh_format       sifel
edge_numbering    0
read_mat_strings  no
read_mat_kwd      yes
read_crs_strings  no
read_crs_kwd      yes
endsec_files


begsec_probdesc
Simple bar structure with viscoplasticity model
mespr 1        # detail output
problemtype    mech_timedependent_prob

straincomp 0  # no explicit strain computation
stresscomp 0  # no explicit stress computation
othercomp  0  # no explicit internal variable computation
reactcomp  0  # no explicit reaction computation

adaptivity     0  # adaptivity is not used
stochasticcalc 0  # deterministic computation
homogenization 0  # homogenization is not applied
noderenumber   no_renumbering  # nodes are not renumbered

time_contr_type fixed # time steps with fixed length
start_time      0.0
end_time        15.5
num_imp_times   0 # the number of important times
funct_type      stat
const_val       0.01 # initial time step length

timetypeprin    seconds
hdbackup        nohdb

nr_num_iter     10
nr_error        1.0e-6
check_div       off

stiffmatstor   skyline_matrix # skyline storage of system
   matrix
```

```
stiffmat_type  initial_stiff
typelinsol     ldl            # solution by LDL decomposition
endsec_probdesc


begsec_loadcase
num_loadcases  1

# the subloadcase approach for the load definition is
   selected, i.e.
# the number of subloadcases is nonzero
lc_id 1    num_sublc 2 # the main load case 1 involves 2
   subloadcases

# load coefficient of the 1. subloadcase
tfunc_lc_id   1
tfunc_slc_id  1
funct_type    tab    # type of general function - table
approx_type   linear # piecewise linear interpolation
ntab_items    8      # the number of rows in table
# {time, load_coef_value} x 8
-1.0 2.2e3
 4.0 2.2e3
 4.0 5.0e2
 8.0 5.0e2
 8.0 2.5e3
12.0 2.5e3
12.0 1.0e2
16.0 1.0e2
# load coefficient of the 2. subloadcase
tfunc_lc_id    1
tfunc_slc_id   2
funct_type     stat  # type of general function - constant
const_val      1.0   # constant value

# temperature load type for the first load case
#
# the first subloadcase
tempr_type_lc_id   1
tempr_type_slc_id  1
temp_load_type     0
# the second subloadcase
tempr_type_lc_id   1
tempr_type_slc_id  2
temp_load_type     0
```

```
endsec_loadcase


begsec_mater
num_mat_types 4
# elastic isotropic material
mattype elisomat num_inst 1
1 e 2.0e9 nu 0.35
# simple 1D plasticity yield condition
mattype simplas1d num_inst 2
1 fs 1.06e7 k 0.0  nostressretalg  nohs
2 fs 2.55e6 k 0.0  nostressretalg  nohs
# simple viscous material
mattype simvisc num_inst 1
1 eta 2.5e-9
# artificial material for combination of visco-plasticity
mattype viscoplasticity num_inst 1
1 # there are no parameters of visco-plasticity material
endsec_mater


begsec_crsec
num_crsec_types 1
crstype csbar2d num_inst 2
1 a 201.1e-6
2 a 113.1e-6
endsec_crsec


# properties of nodes defined by vertices
begsec_nodvertpr
bocon  propid 1  num_bc 2  dir 1  cond 0.0  dir 2 cond 0.0 #
   left and right supports
bocon  propid 2  num_bc 1  dir 2  cond 0.0  # middle support
# time dependent force in the middle node
nod_load  propid 2  lc_id 1  slc_id 1 load_comp 1.0 0.0
endsec_nodvertpr


# properties of nodes defined at regions
begsec_nodvolpr
ndofn 2 propid 1  # number of degrees of freedom at nodes
ndofn 2 propid 2  # number of degrees of freedom at nodes
endsec_nodvolpr
```

```
begsec_elvolpr
# element properties for thick bar
el_type   propid 1 bar2d
el_mat    propid 1 num_mat 4  type viscoplasticity type_id 1
                              type simvisc         type_id 1
                              type simplas1d       type_id 1
                              type elisomat        type_id 1
el_crsec propid 1  type csbar2d  type_id 1
# dead weight load defined as volumetric load
volume_load propid 1  lc_id 1         slc_id 2
            ncomp  2  func_type stat  coord_sys 1
            load_comp 0.0 -12.0e3

# element properties for narrow bar
el_type   propid 2 bar2d
el_mat    propid 2 num_mat 4  type viscoplasticity type_id 1
                              type simvisc         type_id 1
                              type simplas1d       type_id 2
                              type elisomat        type_id 1
el_crsec propid 2  type csbar2d  type_id 2
# dead weight load defined as volumetric load
volume_load propid 2  lc_id 1         slc_id 2
            ncomp  2  func_type stat  coord_sys 1
            load_comp 0.0 -12.0e3
endsec_elvolpr


begsec_outdrv
#-------------------------------
#  Definition of MEFEL output  |
#-------------------------------

# description of output to the text file
textout on
bar-viscopl-slc.out
sel_nodstep  sel_all
sel_nodlc    sel_all
displ_nodes  sel_all  displ_comp sel_all
strain_nodes sel_no
stress_nodes sel_no
other_nodes  sel_no
reactions    1

sel_elemstep sel_all
```

```
sel_elemlc   sel_all
strain_elems sel_all  elemstrain_comp sel_all
   elemstra_transfid 0
stress_elems sel_all  elemstress_comp sel_all
   elemstre_transfid 0
other_elems  sel_all  elemother_comp  sel_all

sel_pointstep sel_no

# description of output to the text file
# 3 = GiD format - one huge file
outgr_format grfmt_gid

# graphics output file name without extension
bar-viscopl-slc

sel_nodstep  sel_all
sel_nodlc    sel_all
displ_nodes  sel_all  displ_comp sel_all
strain_nodes sel_no
stress_nodes sel_no
other_nodes  sel_no
force_nodes  sel_all  force_comp sel_all

sel_elemstep sel_all
sel_elemlc   sel_all
strain_elems sel_all  elemstrain_comp sel_all
   elemstra_transfid 0
stress_elems sel_all  elemstress_comp sel_all
   elemstre_transfid 0
other_elems  sel_all  elemother_comp  sel_all

# text output of graphs
# number of created files with diagrams
numdiag 1
bar-viscopl-slc.dat
numunknowns  5       # number of printed unknowns
sel_diagstep sel_all # type of load step selection = all
   load steps

# point type = node, node id = 2
point atnode node 2
# printed nodal quantity = time
quant_type pr_time
# point type = node, node id = 2
```

```
point atnode node 2
# printed nodal quantity = displacement, 1st component
quant_type pr_displ compid 1
# point type = node, node id = 2
point atnode node 2
# printed nodal quantity = force, 1st component
quant_type pr_forces compid 1
# point type = element ip, local integration point id = 2
point atip elem 1 ip 1
# printed element quantity = stress, 1st component
quant_type pr_stresses compid 1
# point type = element ip, local integration point id = 1
point atip elem 2 ip 1
# printed nodal quantity = stress, 1st component
quant_type pr_stresses compid 1
endsec_outdrv
```

Bibliography

[1] SVN pages of SIFEL versions *http://cml.fsv.cvut.cz/websvn/*

[2] Download page of SIFEL *https://cml.fsv.cvut.cz/getsifel/*

[3] SIFEL web pages *http://mech.fsv.cvut.cz/~sifel*

[4] GiD - The Personal Pre and Post Processor *http://www.gidhome.com/*

[5] - T3D Mesh Generator *http://ksm.fsv.cvut.cz/~dr/t3d.html*

[6] - T3D User Guide *http://mech.fsv.cvut.cz/~dr/software/T3d/guide/guide.html*

[7] - GEFEL manual

[8] - PARGEF manual

[9] - MEFEL manual

[10] - MEFEL input files on *http://mech.fsv.cvut.cz/~sifel/MA1/ONLINE/infiles.html*

[11] - MIDAS manual on *http://mech.fsv.cvut.cz/~da/MIDAS/en/*

[12] - MECHPREP example files on *http://mech.fsv.cvut.cz/~sifel/MAN/mechprep-exam.zip*

[13]

Index