

ELP USER MANUAL

Version for the HPC course
2007

Originally written by
Jelle Muylle
Peter Iványi

Modified by
Peter Iványi

Contents

1	Introduction	4
2	ELP program interaction	5
2.1	Overview	5
2.2	ELP programs	5
3	E_Library file types	7
3.1	The Mesh Definition File	8
3.2	The Material File	15
3.3	The Domain Decomposition File	17
3.4	The Mesh Parameters File	18
3.5	The Finite Element Error File	20
3.6	The Finite Element Stress File	21
3.7	The Element Geometric Definition File	23
4	Unstructured mesh generation: MGN	26
4.1	Status	26
4.2	Syntax	26
4.3	Overview	26
4.4	MGN batch file syntax	27
4.4.1	Tasks	27
4.4.2	Methods	28
4.4.3	Delaunay options	28
4.4.4	Control types	29
4.4.5	Mesh options	29
4.4.6	Filenames	30
4.4.7	Smoothing	30

4.4.8	Output format	30
4.5	CTL local control sources syntax	31
5	2D Finite element analysis: FEM	32
5.1	Status	32
5.2	Syntax	32
5.3	Overview	32
5.4	Program limitations	33
6	Error analysis: ADAPT	34
6.1	Status	34
6.2	Syntax	34
6.3	Overview	34
6.4	Program limitations	35
7	Viewing and printing: E_PLOT32	36
7.1	Status	36
7.2	Syntax	36
7.3	Overview	36
7.4	Help	37
7.5	Program limitations	37
7.6	Alternative platforms	37
8	Editing and modifying E_Library files: MDFTOOLS	38
8.1	CHKMDF	39
8.1.1	Status	39
8.1.2	Syntax	39
8.1.3	Overview	39
8.2	CLEANUP	40
8.2.1	Status	40
8.2.2	Syntax	40
8.2.3	Overview	40
8.3	EMP2MPR	41
8.3.1	Status	41
8.3.2	Syntax	41
8.3.3	Overview	41

8.4	FLOOR	42
8.4.1	Status	42
8.4.2	Syntax	42
8.4.3	Overview	42
8.5	MAKEDOM	43
8.5.1	Status	43
8.5.2	Syntax	43
8.5.3	Overview	43
8.6	MAKEMAT	44
8.6.1	Status	44
8.6.2	Syntax	44
8.6.3	Overview	44
8.7	MAKEMPR	45
8.7.1	Status	45
8.7.2	Syntax	45
8.7.3	Overview	45
8.8	MDFMERGE	46
8.8.1	Status	46
8.8.2	Syntax	46
8.8.3	Overview	46
8.9	MPR2EMP	47
8.9.1	Status	47
8.9.2	Syntax	47
8.9.3	Overview	47
8.10	RENUMBER	48
8.10.1	Status	48
8.10.2	Syntax	48
8.10.3	Overview	48
8.10.4	Program limitations	48

Chapter 1

Introduction

This document contains the user manuals for the different tools bundled in ELP. ELP stands for E_Library Package. It is a series of general purpose finite element analysis and design tools developed by the Structural Engineering Computational Technology (SECT) Research Group at Heriot-Watt University, Edinburgh, UK. All the tools in ELP support the same standard, the E_Library.

The purpose of this manual is to explain the usage and behaviour of the different components of ELP. It is not the aim of this manual to explain the code or supply developer information. For further information see Reference [1]

Chapter 2

ELP program interaction

2.1 Overview

Figure 2.1 shows the different programs which are currently in the ELP. It also shows what kind of files every program expects as input and which ones are generated. More details about the requirements of each program are explained in the appropriate chapters for each program.

The ELP actually splits up into two major components.

- programs for 2D structures and meshes, where the focus is set on efficient and adaptive high quality multi purpose mesh generation; and
- the utilities that are common and that are tools to inspect, correct, view and print meshes at any time in the interactive process.

2.2 ELP programs

The following programs are described in the chapters of this manual:

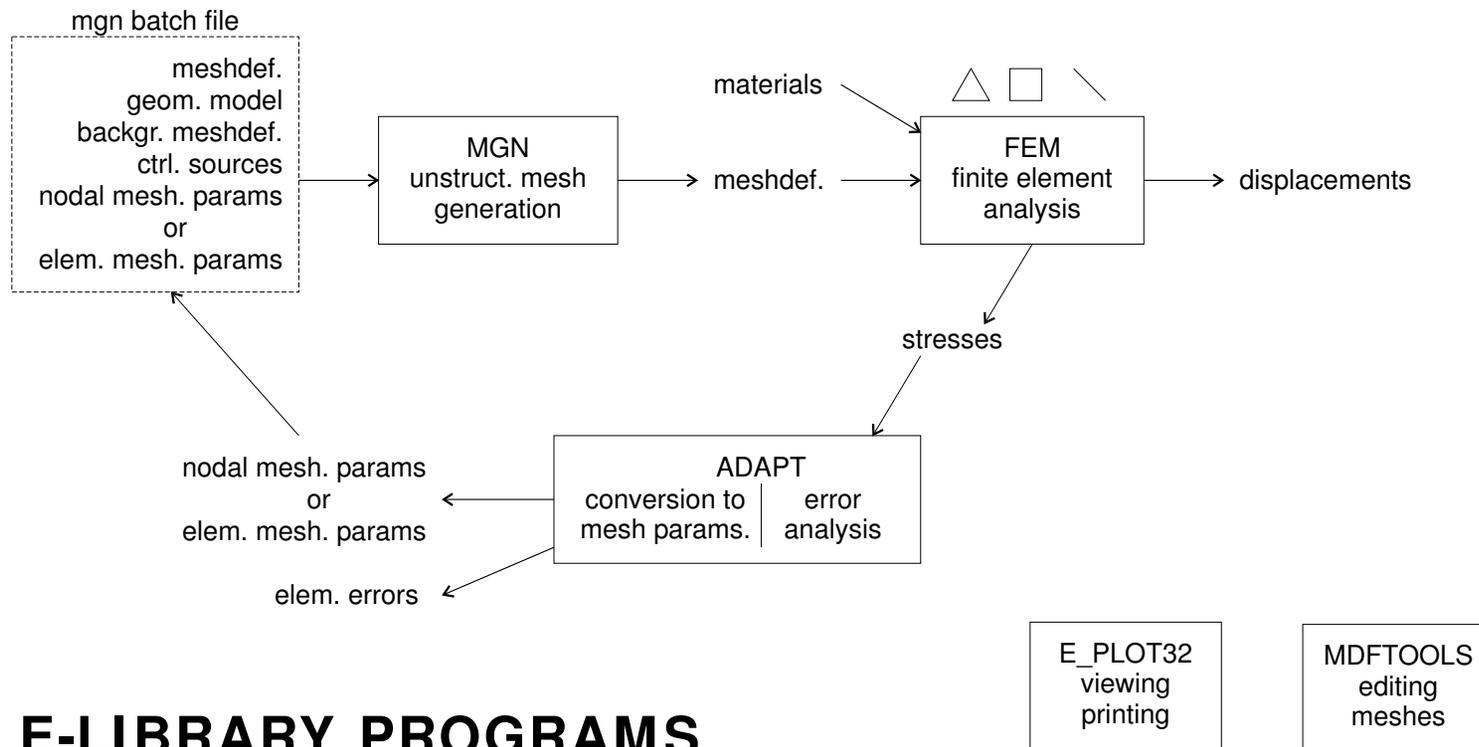
MGN: a multi purpose unstructured 2D mesh generator with a variety of ways to control mesh density at any place in the domain using different mesh generation algorithms. (Chapter 4)

FEM: a very basic, robust finite element analysis program for 2D meshes. (Chapter 5)

ADAPT: generates finite element errors and mesh parameters for an adaptive remeshing of a 2D mesh. (Chapter 6)

E_PLOT32: viewing and printing of meshes, subdomains and mesh parameters in an easy to use windows interface. (Chapter 7).

MDFTOOLS: a series of basic editing tools to check and modify mesh definition files (Chapter 8).



E-LIBRARY PROGRAMS

Figure 2.1: The ELP programs

Chapter 3

E_Library file types

The E-Lib format supports meshes of a single element type and mixed meshes. The E-Lib mesh file format takes the form of a series of keywords followed by one or more data items. For example,

```
NELEMENTS_TRIANG1 120
```

denotes that the mesh contains 120 elements of type **TRIANG1**.

IMPORTANT Most keywords are optional, but those which are declared must be in a strict, predefined order. This order automatically accounts for dependencies between the data.

The format also allows the use of file comments. If the first non-white character on a line is the **#** character, then all remaining text on that line is ignored. Two or more lines can be commented out by enclosing the lines inside a **/*** and ***/**. The opening **/*** must be the first two characters on the first comment line and ***/** the first two characters on the last comment line. All text on the line following **/*** and ***/** is ignored. The **/*...*/** comments must not be nested, although they can contain **#** comments. Blank lines are also permitted.

There are currently seven types of data file – the mesh definition (**.mdf**), geometric definition (NURBS curve) (**.gmf**), materials (**.mat**), domain decomposition (**.dom**), element or nodal mesh parameters (**.mpr**), stresses (**.ste**) and finite element errors files (**.fee**). The mesh definition file contains the main description of the mesh. The other files describe additional properties of the mesh or they represent a state of the mesh.

The following sections describe the keywords and keyword data for the above data files. Each section contains a table which lists the keywords **in the order** they must be declared and defines the associated keyword data. The data type – whether it is an integer (i), real (r) or a character string (s) – is also given, where, for example, ‘i 3*r’ denotes that the data consist of an integer followed by three reals. Units, where relevant, are enclosed in square brackets [...].

IMPORTANT Keyword data consisting of a single data item must follow the keyword on the same line. For vectors and matrices, each vector component or matrix row must start on a new line

and each line must begin with an integer index. Unless otherwise stated, this index is either the vector component number or matrix row index. All keyword data strings must be enclosed in double quotes.

3.1 The Mesh Definition File

The mesh definition file (**.mdf**) contains the main description of the mesh.

At present twelve different element types are supported – five one dimensional (**LINK1**, **LINK2**, **LINK3**, **LINK4**, **LINK5**), five triangular (**TRIANG1**, **TRIANG2**, **TRIANG3**, **TRIANG4**, **TRIANG5**), two quadrilateral (**QUAD1** and **QUAD3**), one tetrahedral (**TETRAH1**) and one three dimensional block element (**BLOCK1**). These element types are defined in Figure 3.1 which shows the order in which the element nodes must be labelled and in Table 3.1 which gives a short description as well. With the exception of the **TRIANG2** and **QUAD3** elements, individual elements have uniform material properties. **TRIANG2** and **QUAD3** elements are composed of layered composite materials referred to as composite materials of type 1.

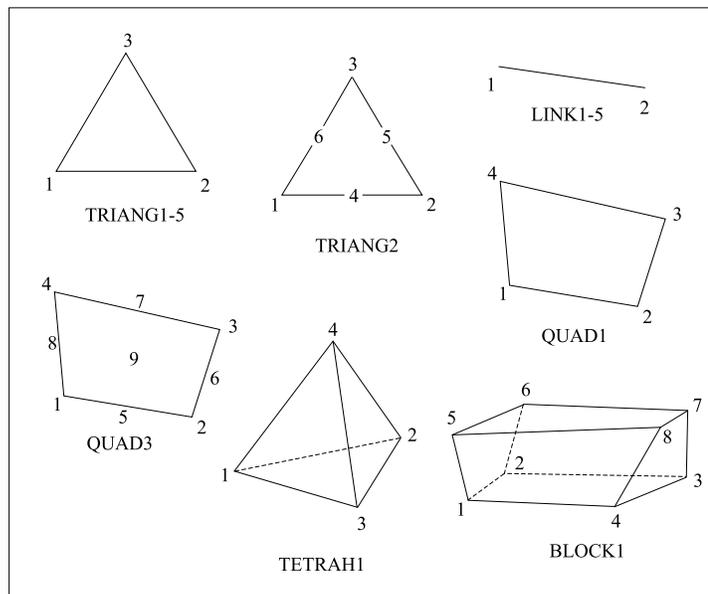


Figure 3.1: The element node orders

An example mesh (Figure 3.2) definition file is shown in the following

Element Type	Description	Number of Vertices	Number of Nodes
LINK1	Truss	2	2
LINK2	Cable	2	2
LINK3	Fixed tension 1-D link	2	2
LINK4	Fixed force density 1-D link	2	2
LINK5	Geodesic string	2	2
TRIANG1	Constant strain triangular	3	3
TRIANG2	A combined constant strain plane stress and constant moment plate bending simple facet triangular	3	6
TRIANG3	Solid triangular	3	3
TRIANG4	Membrane triangular	3	3
TRIANG5	Constant stress triangular	3	3
QUAD1	Plane stress quadrilateral	4	4
QUAD3	Mindlin plate quadrilateral	4	9
TETRAH1	Tetrahedral	4	4
BLOCK1	Block	8	8

Table 3.1: The element types

```
# An example mesh definition file

TITLE "An example mesh"
NMESHPOINTS      6
NNODES           6
NELEMENTS_TRIANG1 4

MESHPOINT_COORDINATES
 1  0.000  0.000  0.000
 2  3.000  0.000  0.000
 3  6.000  0.000  0.000
 4  0.000  3.000  0.000
 5  3.000  3.000  0.000
 6  6.000  3.000  0.000

NODES_TRIANG1
 1  1  2  4
 2  2  5  4
 3  2  3  5
 4  3  6  5
```

The first keyword **TITLE**, is compulsory and specifies the title of the mesh. The title can be used to annotate output such as the display in a plot program and, as with all string arguments, must be enclosed in double quotes.

A mesh is defined in terms of a set of *mesh-points* which are joined by straight lines to form the edges or surfaces of the elements.

IMPORTANT A distinction is made between the terms *mesh-point*, *vertex* and *finite element node*. A

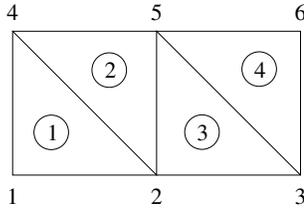


Figure 3.2: An example with TRIANG1 elements

mesh-point is a point in space which may or may not form an element vertex, whereas a finite element node (or simply *node*) is a point on an element used for function interpolation during the finite element analysis. Unlike mesh-points, nodes need not coincide with the element vertices. (See Figure 3.3.)

NMESHPOINTS is the keyword for the number of mesh-points defined in the mesh definition file and **NNODES** defines the total number of finite element nodes in the mesh. The coordinates of the mesh-points follow the keyword **MESHPOINT_COORDS**. All three of the keywords **NMESHPOINTS**, **NNODES** and **MESHPOINT_COORDS** are compulsory. The keyword **NELEMENTS_TRIANG1** act both to declare the type of elements in the mesh and to give the number of the element. If this number is zero then the keyword should be omitted. At least one of the keywords which defines the element type must be declared and for each such keyword present there must also be the corresponding keyword like **NODES_TRIANG1** etc. The data for these keywords define the node indices of each element in turn in the order given in Figure 3.1. Element vertex nodes are always labelled before non-vertex nodes and for two dimensional elements, labelling occurs in an anti-clockwise direction.

Another example is presented to demonstrate the difference between *mesh-point*, *vertex* and *finite element node*. The geometric arrangement of the elements can be seen in Figure 3.3 while the generated mesh file is the following:

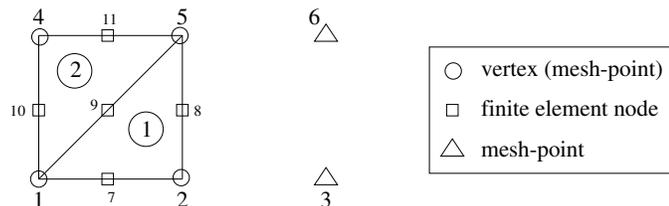


Figure 3.3: An example with TRIANG2 elements

```

# Example with TRIANG 2 elements

TITLE "An example mesh"
NMESHPOINTS      6
NNODES           10
NELEMENTS_TRIANG2  2

MESHPOINT_COORDINATES
  1  0.000  0.000  0.000
  2  3.000  0.000  0.000
  3  6.000  0.000  0.000
  4  0.000  3.000  0.000
  5  3.000  3.000  0.000
  6  6.000  3.000  0.000

NODES_TRIANG2
  1  1  2  5  7  8  9
  2  1  5  4  9  11  10

```

NOTE *All other keywords are optional* except in cases where the declaration of one keyword implies that one or more further keywords will be declared later. For example, if the number of boundary condition nodes is defined (**NBOUNDARY_CONDITION_NODES**) then so must the nodal boundary conditions (**BOUNDARY_CONDITIONS**), see example below.

```

# Example with extra keywords

TITLE "An example mesh"
NMESHPOINTS      3
NNODES           3
NELEMENTS_TRIANG1  1
NBOUNDARY_CONDITION_NODES  2

MESHPOINT_COORDINATES
  1  0.000  0.000  0.000
  2  6.000  0.000  0.000
  3  6.000  6.000  0.000

NODES_TRIANG1
  1  1  2  3

BOUNDARY_CONDITIONS
  1  "FIXED" 0.000  "FIXED" 0.000  "FREE"
  2  "FREE"  "FIXED" 0.000  "FREE"

```

An exhaustive list of possible keywords in an **.mdf** file is presented in Table 3.2a,b,c . For the use of keywords not discussed here see the relevant section in the documentation.

Keyword	Keyword Data	Data Type
TITLE	Mesh title	s
NMESHPOINTS	Number of mesh-points	i
NNODES	Number of nodes	i
NELEMENTS_LINK1	Number of LINK1 elements	i
NELEMENTS_LINK2	Number of LINK2 elements	i
NELEMENTS_LINK3	Number of LINK3 elements	i
NELEMENTS_LINK4	Number of LINK4 elements	i
NELEMENTS_LINK5	Number of LINK5 elements	i
NELEMENTS_TRIANG1	Number of TRIANG1 elements	i
NELEMENTS_TRIANG2	Number of TRIANG2 elements	i
NELEMENTS_TRIANG3	Number of TRIANG3 elements	i
NELEMENTS_TRIANG4	Number of TRIANG4 elements	i
NELEMENTS_TRIANG5	Number of TRIANG5 elements	i
NELEMENTS_QUAD1	Number of QUAD1 elements	i
NELEMENTS_QUAD3	Number of QUAD3 elements	i
NELEMENTS_TETRAH1	Number of TETRAH1 elements	i
NELEMENTS_BLOCK1	Number of BLOCK1 elements	i
NBOUNDARY_CONDITION_NODES	Number of boundary condition nodes	i
NLOADED_NODES	Number of loaded nodes	i
NMATERIALS	Total number of materials including those declared in composite materials	i
NCOMP_MATERIALS_TYPE1	Number of type 1 composite materials	i
NNURBS_CURVES	Number of NURBS curves	i
TIMESTEP	The value of the timestep	r
NTIMESTEPS	Number of time-steps	i
	This keyword and the pair of keywords NINTERNAL_TIMESTEPS and NEXTERNAL_TIMESTEPS are mutually exclusive	
NINTERNAL_TIMESTEPS	Number internal and external time-steps	i
NEXTERNAL_TIMESTEPS	These data enable time-stepping to be broken down into a series of <i>NExternalTimesteps</i> sets of <i>NinternalTimesteps</i> time-steps (See NTIMESTEPS)	
DAMPING_FACTOR	Viscous damping factor	r
BETA	Newmark's β integration constant	r
GAMMA	Newmark's γ integration constant	r

Table 3.2: (a) The mesh definition file keywords and keyword data (cont.)

Keyword	Keyword Data	Data Type
NSTRESS_POINTS_LINK1	Number of stress points in a LINK1 element	i
NSTRESS_POINTS_LINK2	Number of stress points in a LINK2 element	i
NSTRESS_POINTS_LINK3	Number of stress points in a LINK3 element	i
NSTRESS_POINTS_LINK4	Number of stress points in a LINK4 element	i
NSTRESS_POINTS_LINK5	Number of stress points in a LINK5 element	i
NSTRESS_POINTS_TRIANG1	Number of stress points in a TRIANG1 element	i
NSTRESS_POINTS_TRIANG2	Number of stress points in a TRIANG2 element	i
NSTRESS_POINTS_TRIANG3	Number of stress points in a TRIANG3 element	i
NSTRESS_POINTS_TRIANG4	Number of stress points in a TRIANG4 element	i
NSTRESS_POINTS_TRIANG5	Number of stress points in a TRIANG5 element	i
NSTRESS_POINTS_QUAD1	Number of stress points in a QUAD1 element	i
NSTRESS_POINTS_QUAD3	Number of stress points in a QUAD3 element	i
NSTRESS_POINTS_TETRAH1	Number of stress points in a TETRAH1 element	i
NSTRESS_POINTS_BLOCK1	Number of stress points in a BLOCK1 element	i
MESHPPOINT_COORDS	x-,y- and z-coordinates of the mesh-point	i 3*r
NODES_LINK1	Node indices of each LINK1 element	i 2*i
NODES_LINK2	Node indices of each LINK2 element	i 2*i
NODES_LINK3	Node indices of each LINK3 element	i 2*i
NODES_LINK4	Node indices of each LINK4 element	i 2*i
NODES_LINK5	Node indices of each LINK5 element	i 2*i
NODES_TRIANG1	Node indices of each TRIANG1 element	i 3*i
NODES_TRIANG2	Node indices of each TRIANG2 element	i 6*i
NODES_TRIANG3	Node indices of each TRIANG3 element	i 3*i
NODES_TRIANG4	Node indices of each TRIANG4 element	i 3*i
NODES_TRIANG5	Node indices of each TRIANG5 element	i 3*i
NODES_QUAD1	Node indices of each QUAD1 element	i 4*i
NODES_QUAD3	Node indices of each QUAD3 element	i 9*i
NODES_TETRAH1	Node indices of each TETRAH1 element	i 4*i
NODES_BLOCK1	Node indices of each BLOCK1 element	i 8*i
BOUNDARY_CONDITIONS	Nodal boundary conditions For each boundary condition node, the node index followed by, for each degree of freedom, the string " FREE " if no boundary conditions are imposed, " FIXED " followed by a real displacement [m], or " SPRING " followed by a non-negative spring-constant [Nm^{-1}]. The displacement argument specifies an initial displacement of a node. The spring constant enables elastic forces to be modelled.	←

Table 3.2: (b) The mesh definition file keywords and keyword data (cont.)

Keyword	Keyword Data	Data Type
LOADS	Nodal loads For each loaded node, the node index followed by the applied load [N] for each degree of freedom	i r
MATERIALS_LINK1	Material name of each LINK1 element	i s
MATERIALS_LINK2	Material name of each LINK2 element	i s
MATERIALS_LINK3	Material name of each LINK3 element	i s
MATERIALS_LINK4	Material name of each LINK4 element	i s
MATERIALS_LINK5	Material name of each LINK5 element	i s
MATERIALS_TRIANG1	Material name of each TRIANG1 element	i s
COMP_MATERIALS_TRIANG2	Composite material name of each TRIANG2 element	i s
MATERIALS_TRIANG3	Material name of each TRIANG3 element	i s
MATERIALS_TRIANG4	Material name of each TRIANG4 element	i s
MATERIALS_TRIANG5	Material name of each TRIANG5 element	i s
MATERIALS_QUAD1	Material name of each QUAD1 element	i s
COMP_MATERIALS_QUAD3	Composite material name of each QUAD3 element	i s
MATERIALS_TETRAH1	Material name of each TETRAH1 element	i s
MATERIALS_BLOCK1	Material name of each BLOCK1 element	i s
REMESH_DATA	Defines the beginning of NURBS definition	

Table 3.2: (c) The mesh definition file keywords and keyword data

3.2 The Material File

The materials file (**.mat**) contains the properties of the material and composite material declared in the mesh definition file. Any number of materials can be defined and not just those used by the current mesh. This enables a number of different meshes to use a single materials file. The materials and composite materials can be defined in any order.

Each definition of material properties begins with the keyword **MATERIAL** and ends with the keyword **END**. Type 1 composite material properties begin with the keyword **COMP_MATERIAL_TYPE1** and must also end with the keyword **END**. Tables 3.3 and 3.4 define the property keywords.

Keyword	Keyword Data	Data Type
MATERIAL	Material name	s
MAT_TYPE	Material model index	i
DENSITY	Density	r
YMOD	Isotropic Young's modulus This keyword and the x-,y- and z-direction Young's moduli keywords are mutually exclusive	r
YMOD_X	x-direction Young's modulus	r
YMOD_Y	y-direction Young's modulus	r
YMOD_Z	z-direction Young's modulus	r
THICKNESS	Thickness	r
POISSONS_RATIO	Poisson ratio	r
IPARAMn	n -th integer property, $n = 1, \dots, 6$	i
DPARAMn	n -th double precision property, $n = 1, \dots, 20$	r
END		

Table 3.3: The material property keywords and keyword data

Keyword	Keyword Data	Data Type
COMP_MATERIAL_TYPE1	Type 1 composite material name	s
NLAYERS	Number of layers	i
LAYER_MATERIALS	Material name of each layer	i s
LAYER_THICKNESSES	Thickness of each layer	i r
END		

Table 3.4: The type 1 composite material property keywords and keyword data

The structure offers a possibility to introduce new material properties which are not listed among the keywords. For integer values **IPARAM n** and for real values **DPARAM n** should be used.

IMPORTANT When new material property is defined, do not forget to document that which parameters correspond to which material property.

In the case of the composite material the thicknesses can have any value and their sum is not

checked.

An example **.mdf** with materials would be the following.

```
# Example with TRIANG 1 elements and materials

TITLE "An example mesh"
NMESHPOINTS      6
NNODES           6
NELEMENTS_TRIANG1 4
NMATERIALS       2

MESHPOINT_COORDINATES
  1  0.000  0.000  0.000
  2  3.000  0.000  0.000
  3  6.000  0.000  0.000
  4  0.000  3.000  0.000
  5  3.000  3.000  0.000
  6  6.000  3.000  0.000

NODES_TRIANG1
  1  1  2  4
  2  2  5  4
  3  2  3  5
  4  3  6  5

MATERIALS_TRIANG1
  1  "steel"
  2  "concrete"
  3  "concrete"
  4  "steel"
```

While the corresponding **.mat** file looks like this.

```
MATERIAL "steel"
DENSITY 2000.0
YMOD    210000.0
# Yielding stress
DPARAM1 20
END

MATERIAL "concrete"
DENSITY 1000.0
YMOD    50000.0
# Yielding stress
DPARAM1 10
END
```

3.3 The Domain Decomposition File

The domain decomposition file (**.dom**) specifies the subdomains into which the elements have been partitioned (Table 3.5). The keywords **NSUBDOMAINS** and **SUBDOMAINS_TRIANG1** etc. specify the number of subdomains to be created and the partition indices for the various element types. The indices must lie between 1 and the number of subdomains inclusive and the number of each element type must be consistent with the number of that type defined in the mesh definition file.

Keyword	Keyword Data	Data Type
NSUBDOMAINS	Number of subdomains	i
SUBDOMAINS_LINK1	Subdomain index of each LINK1 element	i i
SUBDOMAINS_LINK2	Subdomain index of each LINK2 element	i i
SUBDOMAINS_LINK3	Subdomain index of each LINK3 element	i i
SUBDOMAINS_LINK4	Subdomain index of each LINK4 element	i i
SUBDOMAINS_LINK5	Subdomain index of each LINK5 element	i i
SUBDOMAINS_TRIANG1	Subdomain index of each TRIANG1 element	i i
SUBDOMAINS_TRIANG2	Subdomain index of each TRIANG2 element	i i
SUBDOMAINS_TRIANG3	Subdomain index of each TRIANG3 element	i i
SUBDOMAINS_TRIANG4	Subdomain index of each TRIANG4 element	i i
SUBDOMAINS_TRIANG5	Subdomain index of each TRIANG5 element	i i
SUBDOMAINS_QUAD1	Subdomain index of each QUAD1 element	i i
SUBDOMAINS_QUAD3	Subdomain index of each QUAD3 element	i i
SUBDOMAINS_TETRAH1	Subdomain index of each TETRAH1 element	i i
SUBDOMAINS_BLOCK1	Subdomain index of each BLOCK1 element	i i

Table 3.5: The domain decomposition file keywords and keyword data

An example domain decomposition file is shown for Figure 3.2

```
# An example domain decomposition file
NSUBDOMAINS 2

SUBDOMAINS_TRIANG1
1 1
2 1
3 2
4 2
```

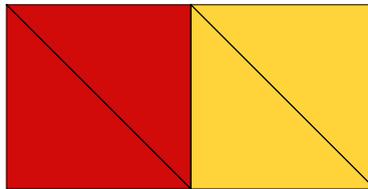


Figure 3.4: An example domain decomposition with **TRIANG1** elements

3.4 The Mesh Parameters File

The mesh parameters file (**.mpr**) defines the element or nodal mesh parameters of each element or node. The number of element mesh parameters for each element type must equal the number of elements defined in the mesh definition file. The keywords for the different element types is shown in Table 3.6.

When the mesh parameters are defined on a nodal basis the **NODAL_MESHPARAMS** keyword should be specified and after that the mesh parameter for each mesh-point should also be defined.

Keyword	Keyword Data	Data Type
NODAL_MESHPARAMS	Mesh parameters for each node	i r
MESHPARAMS_LINK1	Element mesh parameter of each LINK1 element	i r
MESHPARAMS_LINK2	Element mesh parameter of each LINK2 element	i r
MESHPARAMS_LINK3	Element mesh parameter of each LINK3 element	i r
MESHPARAMS_LINK4	Element mesh parameter of each LINK4 element	i r
MESHPARAMS_LINK5	Element mesh parameter of each LINK5 element	i r
MESHPARAMS_TRIANG1	Element mesh parameter of each TRIANG1 element	i r
MESHPARAMS_TRIANG2	Element mesh parameter of each TRIANG2 element	i r
MESHPARAMS_TRIANG3	Element mesh parameter of each TRIANG3 element	i r
MESHPARAMS_TRIANG4	Element mesh parameter of each TRIANG4 element	i r
MESHPARAMS_TRIANG5	Element mesh parameter of each TRIANG5 element	i r
MESHPARAMS_QUAD1	Element mesh parameter of each QUAD1 element	i r
MESHPARAMS_QUAD3	Element mesh parameter of each QUAD3 element	i r
MESHPARAMS_TETRAH1	Element mesh parameter of each TETRAH1 element	i r
MESHPARAMS_BLOCK1	Element mesh parameter of each BLOCK1 element	i r

Table 3.6: The element mesh parameters file keywords and keyword data

An example element mesh parameters file is shown below and in Figure 3.5, which corresponds to the mesh in Figure 3.2.

```
# An example element mesh parameters file
MESHPARAMS_TRIANG1
1 0.1
2 0.2
3 0.2
4 0.3
```

An example nodal mesh parameters file is shown below and in in Figure 3.6, which corresponds to the mesh in Figure 3.2. It should be noted that Figure 3.6 is **NOT** equivalent to Figure 3.5. However it is possible to convert nodal mesh parameters to element mesh parameters and vica versa by the tools **MPR2EMP** and **EMP2MPR**, see Sections 8.9 and 8.3.

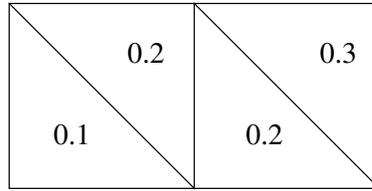


Figure 3.5: An example element mesh parameter definition

```
# An example nodal mesh parameters file

NODAL_MESHPARAMS
1 0.1
2 0.2
3 0.3
4 0.1
5 0.2
6 0.3
```

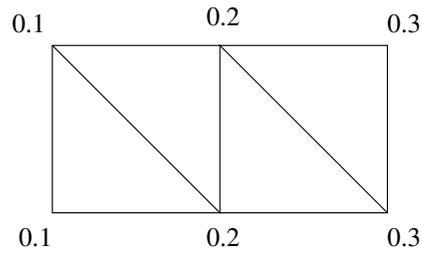


Figure 3.6: An example nodal mesh parameter definition

3.5 The Finite Element Error File

The finite element errors are stored in the finite element error file (**.fee**). One error value must be defined for each element (Table 3.7). Even in the case of **TRIANG2** and **QUAD3** elements which are layered finite elements only one value should be defined for one element.

Keyword	Keyword Data	Data Type
FEERRORS_LINK1	Finite element error of each LINK1 element	i r
FEERRORS_LINK2	Finite element error of each LINK2 element	i r
FEERRORS_LINK3	Finite element error of each LINK3 element	i r
FEERRORS_LINK4	Finite element error of each LINK4 element	i r
FEERRORS_LINK5	Finite element error of each LINK5 element	i r
FEERRORS_TRIANG1	Finite element error of each TRIANG1 element	i r
FEERRORS_TRIANG2	Finite element error of each TRIANG2 element	i r
FEERRORS_TRIANG3	Finite element error of each TRIANG3 element	i r
FEERRORS_TRIANG4	Finite element error of each TRIANG4 element	i r
FEERRORS_TRIANG5	Finite element error of each TRIANG5 element	i r
FEERRORS_QUAD1	Finite element error of each QUAD1 element	i r
FEERRORS_QUAD3	Finite element error of each QUAD3 element	i r
FEERRORS_TETRAH1	Finite element error of each TETRAH1 element	i r
FEERRORS_BLOCK1	Finite element error of each BLOCK1 element	i r

Table 3.7: The finite element error file keywords and keyword data

An example finite element error file is shown for Figure 3.2

```
# An example finite element error file

FEERRORS_TRIANG1
1 0.0021
2 0.0032
3 0.0123
4 0.0001
```

3.6 The Finite Element Stress File

The element stresses are defined in the stress file (**.ste**). The number of stress points for each element must be defined in the **.mdf** file with the corresponding keyword, like **NSTRESS_POINTS_LINK1**, etc. For each stress point a line is defined consisting of 6 components, e.g. normal stresses in the x, y and z directions, the three shear stresses. Other interpretation is possible as well, e.g. the first three components store the principal stresses while the other three components store the angles of the principal directions. Any of the components can be ignored, e.g. in a plane problem only the first three components will be used, as the two normal stresses and a shear stress, or in the case of truss elements only one component is used.

IMPORTANT In the case of **TRIANG2** and **QUAD3** elements the stresses are defined at the stress points per layer per element, therefore it is a requirement that the material file is available for these elements. For all other elements the existence of the material file is not necessary to load/write stresses.

Table 3.8 shows the possible keywords.

Keyword	Keyword Data	Data Type
STRESSES_LINK1	6 available components of stresses	i i 6*r
STRESSES_LINK2	6 available components of stresses	i i 6*r
STRESSES_LINK3	6 available components of stresses	i i 6*r
STRESSES_LINK4	6 available components of stresses	i i 6*r
STRESSES_LINK5	6 available components of stresses	i i 6*r
STRESSES_TRIANG1	6 available components of stresses	i i 6*r
STRESSES_TRIANG2	6 available components of stresses	i i i 6*r
STRESSES_TRIANG3	6 available components of stresses	i i 6*r
STRESSES_TRIANG4	6 available components of stresses	i i 6*r
STRESSES_TRIANG5	6 available components of stresses	i i 6*r
STRESSES_QUAD1	6 available components of stresses	i i 6*r
STRESSES_QUAD3	6 available components of stresses	i i i 6*r
STRESSES_TETRAH1	6 available components of stresses	i i 6*r
STRESSES_BLOCK1	6 available components of stresses	i i 6*r

Table 3.8: The stress file keywords and keyword data

An example finite element stress file is shown for Figure 3.2

```

# An example stress file

# Triang1 stresses
# element index
# Stress point index
#      Szigma x      Szigma y      Szigma z      Tau xy      Tau yz      Tau zx
#-----
STRESSES_TRIANG1
  1  1      1.000e+00      2.000e+00      3.000e+00      4.000e+00      5.000e+00      6.000e+00
  1  2      7.000e+00      8.000e+00      9.000e+00     10.000e+00     11.000e+00     12.000e+00
  2  1     13.000e+00     14.000e+00     15.000e+00     16.000e+00     17.000e+00     18.000e+00
  2  2     19.000e+00     20.000e+00     21.000e+00     22.000e+00     23.000e+00     24.000e+00
  3  1     25.000e+00     26.000e+00     27.000e+00     28.000e+00     29.000e+00     30.000e+00
  3  2     31.000e+00     32.000e+00     33.000e+00     34.000e+00     35.000e+00     36.000e+00
  4  1     37.000e+00     38.000e+00     39.000e+00     40.000e+00     41.000e+00     42.000e+00
  4  2     43.000e+00     44.000e+00     45.000e+00     46.000e+00     47.000e+00     48.000e+00

```

3.7 The Element Geometric Definition File

The element geometric definition file **.gmf** defines the boundary of a finite element problem using Non-Uniform Rational B-Splines (NURBS).

There are two most common nonlinear mathematical forms in geometric modeling for curve and surface representation, one is implicit and another is parametric polynomial forms. The implicit form has the advantage that circles, conics, and primitive quadric surfaces, such as cylinders, spheres and cones can be concisely and precisely represented. A disadvantage of the implicit form is that free-form curves and surfaces, which also important in geometric modeling can not be represented. With parametric polynomials, such as polynomial B-splines, one can represent and manipulate free-form curves and surfaces; but unfortunately, circles, conics and the quadric primitives cannot be represented precisely. Non-uniform Rational B-spline (NURBS) is a geometric modeller that offers the advantages of both forms.

Despite the versatility of NURBS in our implementation only “cubic” NURBS are implemented. Cubic NURBS are defined by two end points and two control points. (All other manipulation of NURBS, such as degree elevation, Bezier Curves conversion, knot-removal and local smoothing or modification is not present in the current implementation.)

The element geometric definition file (**.gmf**) file contains the following keywords:

- **NENDPOINTS**: Number of end points used for the geometric modelling.
- **NNURBS_CURVES**: Number of NURBS curves.
- **ENDPOINT_COORDINATES**: A list of coordinates of end points. These nodes will be referred by their node number in the curve specifications. (Good practice to use the same mesh points here as they are in the mesh. In this case the compatibility between the mesh and the geometric definition can always be ensured.)
- **NURBS_CURVE**: Defines each NURBS curve.
- **DEGREE**: The number of freedom for the NURBS curve. At the moment it must always be equal to three!
- **CONTROL_POINTS**: The four control points for Cubic NURBS curve.
- **WEIGHTS**: The value of weights for the four control points.

The geometric definition file should only be used together with the remeshing data extensions of the mesh definition file. These extensions are not covered in the main definition of the MDF syntax and can be best explained by the following example.

Figure 3.7a shows one coarse triangle defined by nodes 1,2 and 3. On the side between nodes 1 and 2 a NURBS curve **C1** is defined. The curve is cubic (degree 3) and stretches from node 1 where it has paramete 0.0 to node 2 where it has parameter 1.0. Two control points determine the shape of the curve. They have coordinates (0,-10,0) and (10,-10,0). The weights for these control points are set to 0.5. Figure 3.7b shows how the remeshed triangle may look like when three nodal meshparameters were defined with value $\delta=0.8$.

The following three files show how to create the curve definition in the geometric definition file and how to assign it in the mesh definition file.

```

# coarse.mdf : mesh definition file

TITLE "coarse mesh"
NMESHPOINTS 3
NNODES 3
NELEMENTS_TRIANG1 1
NNURBS_CURVES 1

MESHPOINT_COORDINATES
1 0.0 0.0 0.0
2 10.0 0.0 0.0
3 5.0 8.67 0.0

NODES_TRIANG1
1 1 2 3

REMESH_DATA
1 "EXTERIOR"
  NNURBS_CURVES 1
  1 NURBS_CURVE "C1"
  PARAMETER 0.0
2 "EXTERIOR"
  NNURBS_CURVES 1
  1 NURBS_CURVE "C1"
  PARAMETER 1.0
3 "EXTERIOR"

```

```

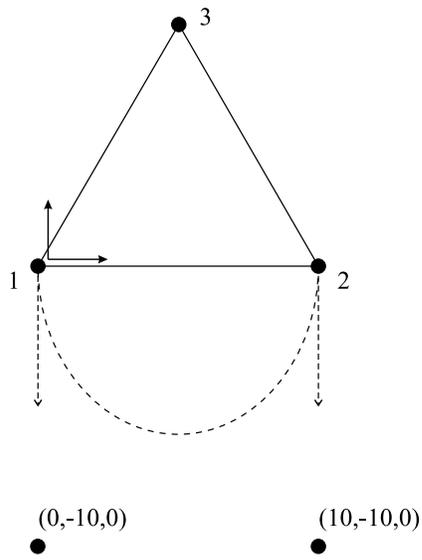
# coarse.gmf : geometric definition file

NENDPOINTS 3

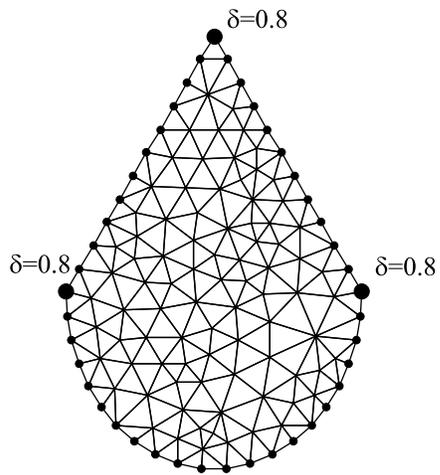
ENDPOINT_COORDINATES
1 0.0 0.0 0.0
2 10.0 0.0 0.0
3 5.0 8.67 0.0

NURBS_CURVE "C1"
DEGREE 3
CONTROL_POINTS
  1 1
  2 0.0 -10.0 0.0
  3 10.0 -10.0 0.0
  4 2
WEIGHTS
  1 1
  2 0.5
  3 0.5
  4 1
END

```



(a)



(b)

Figure 3.7: An example for geometric definition

Chapter 4

Unstructured mesh generation: MGN

4.1 Status

name of executable(s)	mgn mgn.exe
platform(s)	irix linux win32
current version	v1.0
date	July 2003
release	full release
E_lib filetypes	*.mdf *.mpr *.gmf
own filetypes	*.mgn *.ctl

4.2 Syntax

SECT Research Group, Heriot-Watt University Edinburgh

usage: mgn filename [-options]

filename : name of mesh generation batch file without extension

-g : activates the graphical interface

4.3 Overview

MGN is a multi-purpose unstructured 2D mesh generator. It generates, refines or remeshes 2D meshes according to four different algorithms:

- advancing front mesh generation;
- Delaunay triangulation;
- paving; and
- regular grid method.

The density of the generated mesh can be controlled by four different mechanisms:

- The mesh is uniform in size and a uniform mesh parameter is specified.
- The mesh parameter is determined from a corresponding value in a background mesh.

- The mesh parameter is directly specified for each element or each node in a mesh parameter file.
- The mesh size is controlled by a series of local control sources, points or series of points in which neighbourhood the density is specified.

Every triangular mesh, whether it was newly generated or just refined or remeshed can be converted into a quadrilateral mesh by fusion or fission of triangles. Every quadrilateral mesh, whether it was newly generated or just refined or remeshed can be converted into a triangular mesh by splitting of quadrilaterals into triangles.

Smoothing of the mesh in a specified number of iterative steps is also supported.

4.4 MGN batch file syntax

The tasks set for the mesh generator are specified in a batch file. The MGN batch file controls the entire behaviour of the MGN program by listing all the instructions and filenames for the input and output files. The syntax of the batch file is given in the following table. (Although this syntax is not incorporated in the E_Library it follows the same syntax rules set for all the E_Library files: i.e. the use of keywords the rules for comments and empty lines.)

TASK	"REMESH" "REFINE" "EBE_REFINE" "POSTPROC"
METHOD	"ADV_FRONT" "DELAUNAY" "PAVING" "REG_GRID" "NONE"
ALFA	d
BETA	d
DLN_OPTION	"CENTROID" "CENTR_RAD" "PROJECTION" "CHEW1"
CONTROL_TYPE	"BACKGROUND" "MPR" "SOURCES" "UNIFORM"
MESH_OPTION	"SPLIT_TO_TRIANG" "FISSION_TO_QUAD" "RANK_TO_QUAD" "HALF_TO_QUAD" "NONE"
GEOM_MODEL_FILE	"*"
COARSE_MESH_FILE	"*"
RESULT_MESH_FILE	"*"
BKGRND_MESH_FILE	"*"
BKGRND_MESH_PARAMS_FILE	"*"
COARSE_MESH_PARAMS_FILE	"*"
LOCAL_MESH_CTRL_FILE	"*"
UNIFORM_MESH_PARAM	d
SMOOTHING_TYPE	"LAPLACIAN" "LAPLACIAN_NDE" "NODAL_AVG" "PAVING" "NONE"
SMOOTHING_ITER	n
OUTPUT_FORMAT	"E_LIB" "SMSH"

4.4.1 Tasks

First of all one has to decide what the mesh generator has to do. Four tasks are available:

- Choose "REMESH" as TASK setting for the remeshing of an existing mesh, keeping only the

boundaries of the mesh and discarding any existing points on the interior of the domain. If a geometric model is supplied, a new approximation of the boundary will be made as well.

- Choose "REFINE" as TASK setting for the refinement of an existing mesh, keeping all existing nodes and refining every existing triangle or quad.
- Choose "EBE_REFINE" as TASK setting for the refining of an existing mesh on an element by element basis.
- Choose "POSTPROC" as TASK setting if only postprocessing and mesh conversion is required.

4.4.2 Methods

The mesh generator supports five methods for generating the unstructured mesh:

- Choose "ADV_FRONT" as METHOD setting for the advancing front technique which is rather slow, but creates very regular high quality meshes.
- Choose "DELAUNAY" as METHOD setting for the Delaunay triangulation algorithm (by point insertion) which is on average ten times faster, but for which regularity and quality of the mesh can be less than advancing front.
- Choose "PAVING" as METHOD setting for the direct generation of quadrilaterals using the paving technique.
- Choose "REG_GRID" as METHOD setting for the regular grid method, which is the subject of reference [2]

When the Delaunay triangulation or regular grid method is chosen, the user has to specify two extra parameters ALFA and BETA and an extra option DLN_OPTION.

- Alfa controls the conformity of the mesh density with the parameters set by the control type. For DLN_OPTION set to "CENTROID" the requirement for alfa is $1.0 < \alpha$. For DLN_OPTION set to "CENTR_RAD" the requirement for alfa is $0.5 \leq \alpha \leq 2.0$. Values close to 1.0 are a good starting point. For METHOD set to "REG_GRID" alfa takes the function of parameter `gprm` from reference [2].
- Beta ($0.1 \leq \beta \leq 15.0$) controls the regularity of the mesh. Lower values of beta generate more chaotic meshes whereas higher values would give a more regular mesh. A good default value is 2.0. For METHOD set to "REG_GRID" beta takes the function of parameter `rprm` from reference [2].

4.4.3 Delaunay options

When the Delaunay triangulation is chosen the user must define which Delaunay refinement strategy is used. This is controlled by the DLN_OPTION keyword. The following options are implemented:

- "CENTROID" controls the mesh size on the basis of side lengths of triangles. Centroidal nodes are inserted into a triangle when at least one of its side lengths is longer than allowed.
- "CENTR_RAD" controls the mesh size on the basis of the lengths of the radial connection between the centroid node and the corner node of triangles. Centroid node is inserted into a triangle when any of the radial connections are longer than allowed.
- "PROJECTION" behaves like "CENTROID" but uses projective point placement instead of centroidal nodes.
- "CHEW1" uses the point insertion method suggested by Chew [3].

4.4.4 Control types

An efficient control of the mesh density at any place inside the domain and on the borders is essential for any good mesh generator. MGN offers four control types:

- Choose "UNIFORM" as `CONTROL_TYPE` for the generation of uniform meshes. The uniform size will be taken from the value specified for the `UNIFORM_MESH_SIZE` keyword.
- The use of a background mesh which sits behind the domain is a common technique. For every new point to be generated in the mesh the meshparameter is derived from the value this point would have if it belonged to the background mesh. Therefore care has to be taken that the background mesh covers the complete area of the domain, including potential extensions of the domain by a new approximation of the boundary curves. The background mesh also has to be convex. Choose "BACKGROUND" as `CONTROL_TYPE` setting and don't forget to define the `BKGRND_MESH_FILE` and the `BKGRND_MESH_PARAMS_FILE` keywords.
- Mesh parameter controlled meshing is equivalent to background controlled meshing where the background mesh is the coarse mesh itself. However the mesh parameters specified in the `COARSE_MESH_PARAMS_FILE` have to be nodal mesh parameters to make sense. (If element mesh parameters are specified they will be converted to nodal parameters by averaging and will therefore loose effectiveness.) Choose "MPR" as `CONTROL_TYPE` and don't forget to define the `COARSE_MESH_PARAMS_FILE` keyword.
- The use of local control sources is the third type. It is an extension to background controlled meshing whereby the user can define some local control sources in a separate file that will influence the mesh density on a very small scale. Section 4.5 will explain the syntax and the effect of these sources. Choose "SOURCES" as `CONTROL_TYPE` and don't forget to define the `LOCAL_MESH_CTRL_FILE` keyword.

4.4.5 Mesh options

When the coarse mesh is not triangular but contains quadrilaterals, you have to specify a mesh option when you generate the mesh using Delaunay triangulation. This option will determine the way the Delaunay triangles are converted into quads so that the output is of the same element type as the input.

When the coarse mesh is triangular, you can specify a mesh option anyway if you want the output to be a mesh of quads. This works for both mesh generation methods.

Currently three quad conversion options are supported:

- "FISSION_TO_QUAD": In this method triangles are merged to quad elements. The method also considered the case when by merging two triangles two subregions would form with odd number of elements in them. In this case the method inserts a fission element.
- "RANK_TO_QUAD": In this method triangles are also merged to quad elements, regardless of any other constraint. In this case there can be cases when individual triangles are left in the mesh surrounded by quad elements which cannot be merged with any other triangle. To solve this problem the method also halves every element in the mesh (quads into four and triangles into three quad elements) after there is no more triangle merge is possible.
- "HALF_TO_QUAD": This method simply halves every triangle element into three quad elements.
- Choose "NONE" if you don't want any triangle to quad conversion.

If you are directly generating quadrilaterals using the "PAVING" method, and you want a triangular mesh as output you can set the mesh option to "SPLIT_TO_TRIANG" to activate a splitting procedure that will cut each quadrilateral into two triangles along the shortest diagonal.

4.4.6 Filenames

Filename references have to be absolute or relative to the location of the executable. Filenames are always given without extension as it is assumed that the extension matches the filetype syntax. Filenames are enclosed in double quotes.

The following filenames may be defined:

- **GEOM_MODEL_FILE**: the location and name of the geometric model file without the `*.gmf` extension. This file is required for new mesh generation from geometric model only. It may be, but doesn't have to be supplied for the other tasks.
- **COARSE_MESH_FILE**: the location and name of the coarse mesh definition file without the `*.mdf` extension. This file is not required when performing new mesh generation from geometric model only. It is required in the other tasks.
- **RESULT_MESH_FILE**: the location and name of the resulting mesh to which an extension will be added according to the output format. If the file exists already, it will be overwritten. This filename always has to be supplied.
- **BKGRND_MESH_FILE**: the location and name of the background mesh definition file without the `*.mdf` extension. This file is required when background or sources controlled mesh generation is chosen.
- **BKGRND_MESH_PARAMS_FILE**: the location and name of the background mesh parameters file without the `*.mpr` extension. This file is required when background or sources controlled mesh generation is chosen.
- **COARSE_MESH_PARAMS_FILE**: the location and name of the mesh parameters file without the `*.mpr` extension. This file is required when mesh parameter controlled mesh generation is chosen.
- **LOCAL_MESH_CTRL_FILE**: the location and name of the local control sources file without the `*.ctl` extension. This file is required when local control sources are requested.

4.4.7 Smoothing

The mesh that results from any of the operations mentioned above can be smoothed. The only smoothing type that is currently supported is optimized Laplacian smoothing. Choose "LAPLACIAN" or "LAPLACIAN_NDE" if Laplacian smoothing with or without diagonal exchange is required. "NODAL_AVG" will activate pure nodal averaging. if "NONE" is chosen as `SMOOTHING_TYPE` setting no smoothing is carried out at all. If smoothing is activated, you also have to specify the number of smoothing iterations to be carried out. This number is set by `SMOOTHING_ITER` and has to be bigger than one. Two iterations are in most cases sufficient.

In the case of paving the smoothing type must be set to "PAVING" as the paving method has its own smoothing procedures.

4.4.8 Output format

The output format of the resulting mesh can be specified by `OUTPUT_FORMAT`. Currently only "E_LIB" and "SMSH" are supported. "E_LIB" will generate E_Library compatible output and will add the `*.mdf` extension to the resulting mesh filename. "SMSH" will generate meshes in the format for the acoustics software `son3d` and will add the `*.smsh` extension to the resulting mesh filename.

4.5 CTL local control sources syntax

The local mesh density control data file (`.ctl`) defines a set of local control points for point-wise and line-wise mesh size control. For the point-wise local control the point coordinates, the radius of inner circle, the radius of outer circle and nodal mesh size parameter are specified. For line-wise local control a chain of control points will be specified.

The syntax will be explained by an example file:

```
NCONTROL_SETS 2
NCONTROL_POINTS 2
COORDINATES 8.5 2.0 0.0
INNER_RADIUS 0.2
OUTER_RADIUS 1.0
MESH_PARAMETER 0.3
COORDINATES 1.5 2.0 0.0
INNER_RADIUS 0.2
OUTER_RADIUS 1.5
MESH_PARAMETER 0.2

NCONTROL_POINTS 1
COORDINATES 2.0 5.5 0.0
INNER_RADIUS 0.2
OUTER_RADIUS 1.0
MESH_PARAMETER 0.2
```

Chapter 5

2D Finite element analysis: FEM

5.1 Status

name of executable(s)	fem fem.exe
platform(s)	irix linux win32 command line
current version	v1.1
date	July 2003
release	unfinished
E_lib filetypes	*.mdf *.mat *.ste
own filetypes	-

5.2 Syntax

```
fem v1.1 SECT Research Group, Heriot-Watt University Edinburgh
usage: fem input output ds
note all filenames are referred to without extensions
  input      : name of the mesh definition and material file
  output     : name of the output mesh definition file
  ds         : displacement scale (e.g. 10)
```

you might want to renumber the mesh first!
and remove unused nodes!

5.3 Overview

`fem` is a very limited 2D finite element program which calculates stresses and displacements for meshes consisting of `TRIANG1` and `QUAD1` elements only. Loads, boundary conditions and material assignments should be specified in the input mesh file. Material information should be given in the material file with the same name as the mesh definition file.

As `TRIANG1` and `QUAD1` are constant stress elements the output stress file will consist of one stress-point per element. As `fem` is a 2D FE package only three stresses will appear in the output stress file: σ_x , σ_y and τ_{xy} .

In order to increase the efficiency of the computation and to reduce the calculation times you should not include any unused nodes in the coordinates array. Renumbering of the nodes to decrease the bandwidth of the stiffness matrix is also advised. The `renumber` program can be used for this

purpose. `renumber` is explained in Section 8.10.

5.4 Program limitations

- Only meshes consisting entirely of one of the two supported element types can be analysed.
- From the material file only `YMOD`, `POISSONS_RATIO` and `THICKNESS` will be taken into account.
- The program gives no status reports and might take ages if you have not renumbered your mesh.
- The program will crash if the mesh contains unconnected nodes. These are nodes which are not used in any of the element definitions. Use the `cleanup` program to remove unconnected nodes. `cleanup` is explained in Section 8.2

Chapter 6

Error analysis: ADAPT

6.1 Status

name of executable(s)	adapt adapt.exe
platform(s)	irix linux win32 command line
current version	v1.1
date	July 2003
release	development
E_lib filetypes	*.mdf *.mat *.ste *.fee *.mpr
own filetypes	-

6.2 Syntax

```
adapt v1.1 SECT Research Group, Heriot-Watt University Edinburgh
usage: adapt input d mprtype
note all filenames are referred to without extensions
  input    : name of the mesh definition and stress file
  d        : permissible error value
  mprtype  : 1 for element mesh params
            : 2 for nodal mesh params
```

6.3 Overview

`adapt` is a small error analysis program for the sort of meshes that can be handled by `fem`. It generates finite element errors on the basis of the stress file by comparing averaged and non averaged nodal stresses.

The finite element element errors are then converted into mesh parameters either nodal or element parameters as specified in the command line. Make sure you choose the right type depending on what remeshing or viewing you want to do next.

It is always advised to check the generated mesh parameters before a new remeshing run is launched. Some mesh parameters might be far to small to be realistic. The `floor` program might be used to set a lower limit to the mesh parameters. `floor` is explained in Section 8.4.

6.4 Program limitations

- Meshes may contain only one of the two supported element types.
- The material file should also exist as material information is required for processing the finite element errors.
- Both material and stress files are expected to have the same name as the mesh definition file.

Chapter 7

Viewing and printing: E_PLOT32

7.1 Status

name of executable(s)	e_plot32.exe
platform(s)	win32
current version	v2.52
date	July 2003
release	complete
E_lib filetypes	*.mdf *.mpr *.dom *.fee *.ste
own filetypes	-

7.2 Syntax

run e_plot32.exe from explorer or start menu - run...
or make a shortcut to e_plot32.exe

7.3 Overview

E_plot32 is a native windows 32bit graphical user interface program. It will run on Windows95, Windows98 and Windows NT 4. The program provides a viewer for all sorts of meshes in the E_Library mesh definition format.

The user can view the mesh in the following plot configurations:

- plain mesh plot: just shows the mesh connectivity.
- subdomain plot: colours the different subdomains according to the subdomain information stored in the appropriate subdomain file.
- stress plot: shows a colour representation of all the stresses that are available in the stress file.
- FEerror plot: plots the distributions of the finite element errors as specified in the FEerror file.
- mesh parameter plot: makes a plot of the mesh parameters as stored in the mesh parameter file.

The program can print directly from the menu and also has an export facility to the `*.wmf` format which can be read by many vector graphics packages such as CorelDraw and AutoCAD 2D.

7.4 Help

For a more detailed explanation of each menu option we refer to the online help system within the program. Help is available by selecting the Help menu or by simply pressing **F1** at any time in the program.

7.5 Program limitations

- All files should have the same name as the input `*.mdf` files with the appropriate extensions.
- All filetypes are accepted except `QUAD3` and `BLOCK1`.
- For meshes containing `TRIANG2` elements: only the stresses for `layer 0` are shown in a stress plot.
- When exporting a subdomain plot to `*.wmf` format the subdomain colours are converted to grayscale. This does not happen with the stress, `FError` or mesh parameter plots.
- An export to `*.wmf` might mess up the scaling of the image. Scale it by a factor 100 to obtain normal sizes. However this depends on the graphics package you use.
- `E_plot32` will only accept element mesh parameters for a mesh parameter plot.

7.6 Alternative platforms

The viewer technology of `E_plot32` was also ported into a Linux and Irix version. These unix versions of the viewer have limited features and are continuously updated. If required obtain the most recent information from the author concerning the status of `ep1x` and `qmv`.

Chapter 8

Editing and modifying E_Library files: MDFTOOLS

The MDFTOOLS are a series of checking, editing and conversion tools to help creating and maintaining E_Library files. Some of them are batch files, some are small programs. A good knowledge of scripting or shell languages is advised in the preparation of E_Library files as there is (so far) no graphical design utility for the E_Library standard.

8.1 CHKMDF

8.1.1 Status

name of executable(s)	chkmdf
platform(s)	irix linux win32 command line
current version	v1.0
date	July 2003
release	stable
E_lib filetypes	*.mdf
own filetypes	-

8.1.2 Syntax

Validity checking of an e_lib mesh definition

Usage: `chkmdf filename`

`filename` : name of mesh definition file

All files should be specified without extension!

8.1.3 Overview

`chkmdf` is a tool that checks the validity of a mesh definition file. It displays mesh statistics and performs the following checks:

- Are there any duplicate nodes?
- Are there any invalid elements, i.e. elements in whose definition the same node number appears twice or more?
- Are there any duplicate elements? Here the node order is not taken into account.

8.2 CLEANUP

8.2.1 Status

name of executable(s)	cleanup cleanup.exe
platform(s)	irix linux win32 command line
current version	v1.0
date	July 2003
release	stable
E_lib filetypes	mdf
own filetypes	-

8.2.2 Syntax

Remove unconnected nodes from a mesh.

Usage: cleanup input output

input : name of input mesh definition file

output : name of resulting mesh definition file

All files should be specified without extension!

8.2.3 Overview

When meshes are adaptively remeshed with `mgn` using the task `REMESH` all the nodes of the input coarse mesh are copied into the resulting file though they might not be used in any of the new triangles. These unconnected nodes cause problems when renumbering the mesh for bandwidth reduction.

`cleanup` removes all unconnected nodes from a mesh definition file. The program only works with meshes containing only `TRIANG1` elements. The topology of the mesh is not altered, only the numbering of points and the point numbers in the element definition. Boundary conditions and loads are also renumbered appropriately.

8.3 EMP2MPR

8.3.1 Status

name of executable(s)	emp2mpr
platform(s)	irix linux win32 command line
current version	v1.0
date	July 2003
release	stable
E_lib filetypes	mdf mpr
own filetypes	-

8.3.2 Syntax

Convert element mesh parameters to nodal mesh parameters
by averaging all element mesh parameters connected to a node.

Usage: emp2mpr input output

input : name of input mesh file and mesh parameter file

output : name of output mesh parameter file

All files should be specified without extension!

8.3.3 Overview

emp2mpr converts an element mesh parameter file into a nodal mesh parameter file. The program requires, as an input, the mesh definition file and the element mesh parameter file, while the output is only a new nodal mesh parameter file. The program only works with meshes containing only TRIANGLE1 elements.

8.4 FLOOR

8.4.1 Status

name of executable(s)	floor floor.exe
platform(s)	irix linux win32 command line
current version	v1.0
date	July 2003
release	stable
E_lib filetypes	*.mdf *.mpr
own filetypes	-

8.4.2 Syntax

Limit the maximum and minimum values of the mesh parameter.

Usage: floor input output

input : name of input mesh file and mesh parameter file

output : name of output mesh parameter file

All files should be specified without extension!

8.4.3 Overview

floor is a little utility to alter mesh parameter files. Both nodal and element mesh parameter file are accepted. The file is parsed and minimum, maximum and average mesh parameter are printed on the screen. Then the user is asked to specify a new minimum and maximum parameter for this file.

8.5 MAKEDOM

8.5.1 Status

name of executable(s)	makedom
platform(s)	irix linux win32 command line
current version	v1.0
date	July 2003
release	stable
E_lib filetypes	mdf dom
own filetypes	-

8.5.2 Syntax

Create a domain decomposition file (*.dom).

```
Usage: makedom [-v|-h|-n<domain>] mdf_file
       -v - print version information
       -h - print this message
       -n<domain> - set domain number (default 1)
```

8.5.3 Overview

`makedom` creates a domain decomposition file which is compatible with the input mesh definition file. By default it generates a domain decomposition where all elements are in one domain. However it is also possible to specify that all elements should be assigned to a user defined subdomain. For example `makedom -n 2 mesh` will generate a domain decomposition file where all elements belong to subdomain two and the maximum number of subdomains is also equal to two.

8.6 MAKEMAT

8.6.1 Status

name of executable(s)	makemat
platform(s)	irix linux win32 command line
current version	v1.0
date	July 2003
release	stable
E_lib filetypes	mdf mat
own filetypes	-

8.6.2 Syntax

Creates material information and an example material file

Usage: makemat input output

input : name of input mesh file

output : name of output mesh and material file

All files should be specified without extension!

8.6.3 Overview

`makemat` creates material information for a mesh definition file and it also generates an example material file (`.mat`). The program only accepts meshes which contain a single element type. Moreover the acceptable element types are TRIANG1, TRIANG3 and LINK1.

8.7 MAKEMPR

8.7.1 Status

name of executable(s)	makempr
platform(s)	irix linux win32 command line
current version	v1.0
date	July 2003
release	stable
E_lib filetypes	mdf mpr
own filetypes	-

8.7.2 Syntax

Creates mesh parameter file (element or nodal)

```
Usage: makempr [-v|-h|-n<meshparam>|-e<meshparam>|-a] mdf_file
-v - print version information
-h - print this message
-n<meshparam> - set value of nodal mesh parameter
-e<meshparam> - set value of element mesh parameter
-a - automatic calculation of nodal mesh parameter (default)
```

8.7.3 Overview

`makempr` can create a nodal or element mesh parameter file. When the mesh parameter is explicitly specified on the command line all mesh parameters in the file will be the specified value. In the case of the `-a` option the program determines the minimum edge length at each point and this minimum edge length will become the nodal mesh parameter at each point.

8.8 MDFMERGE

8.8.1 Status

name of executable(s)	mdfmerge
platform(s)	irix linux
current version	v1.0
date	July 2003
release	development
E_lib filetypes	mdf
own filetypes	-

8.8.2 Syntax

Merges two mdf files.
Only merges geometry and domain decomposition,
but no loads, bc, etc.

Usage: mdfmerge mesh1 mesh2 out [d]
mesh1: first mesh definition file
mesh2: second mesh definition file
out: output mesh definition file
d : when not zero it also merges decomposition data

8.8.3 Overview

The mdfmerge program merges the geometry of two mesh definition files. When the optional argument is specified and it is not zero then the program also merges the domain decomposition data in the mesh. However the program does not merge loads, boundary conditions, materials, etc.

8.9 MPR2EMP

8.9.1 Status

name of executable(s)	mpr2emp
platform(s)	irix linux win32 command line
current version	v1.0
date	July 2003
release	stable
E_lib filetypes	mdf mpr
own filetypes	-

8.9.2 Syntax

Convert nodal mesh parameters to element mesh parameters by averaging all nodal mesh parameters of an element.

Usage: mpr2emp input output

input : name of input mesh file and mesh parameter file

output : name of output mesh parameter file

All files should be specified without extension!

8.9.3 Overview

mpr2emp converts a nodal mesh parameter file into an element mesh parameter file. The program requires, as an input, the mesh definition file and the nodal mesh parameter file, while the output is only a new element mesh parameter file. The program only works with meshes containing only TRIANGLE1 elements.

8.10 RENUMBER

8.10.1 Status

name of executable(s)	renumber renumber.exe
platform(s)	irix linux win32 command line
current version	v1.0
date	July 2003
release	stable
E_lib filetypes	*.mdf
own filetypes	-

8.10.2 Syntax

Renumbers the mesh nodes to reduce the matrix bandwidth.

Usage: renumber input output

input : name of the input mesh definition file

output : name of the resulting mesh definition file

All files should be specified without extension!

8.10.3 Overview

`renumber` is a small program that changes the node indices of a mesh in order to minimize the bandwidth of the stiffness matrix when this one is build in any finite element package. It changes the node numbers accordingly in loads and boundary condition nodes.

8.10.4 Program limitations

- `renumber` works only for single element type `TRIANG1` or `QUAD1` meshes.
- Only the information within the mesh definition file is altered. Subdomain, stress, finite element error or mesh parameter information is left untouched.
- The program will crash if the mesh contains unconnected nodes. These are nodes which are not used in any of the element definitions. Use the `MDFTOOL cleanup` to remove unconnected nodes. `cleanup` is explained in Section 8.2

Bibliography

- [1] B. H. V. Topping, J. Muylle, P. Iványi, R. Putanowicz, and B. Cheng. *Finite Element Mesh Generation*. Saxe-Coburg Publications, Stirling, 2004.
- [2] J. Muylle, P. Iványi, and B. H. V. Topping. A new point creation scheme for uniform Delaunay triangulations. *Engineering Computations, International Journal for Computer Aided Engineering and Software*, 19(6):707–735, 2002.
- [3] P. L. Chew. Guaranteed-quality delaunay triangulations. Technical Report TR-89-983, Dept. of Computer Science, Cornell University, 1989.