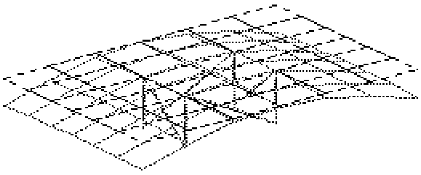
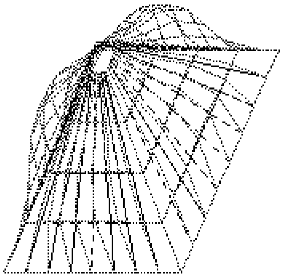
UNI KASSEL
VERSITÄT

Fachgebiet Leichtbau und
Laboratorium für Strukturmechanik

Prof. Dr.-Ing. Michael Link

Phone : +49 (0)561 804 2654 / 2632
Fax : +49 (0)561 804 3631
Mail: link@uni-kassel.de
Internet: www.uni-kassel.de\fb14\leichtbau

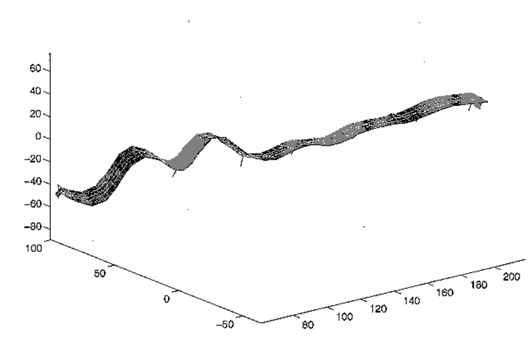




MATFEM 04

User's Guide

Revision 09-Aug-2005



Contents:

Appendix:

Abbreviations

w.r. to	with respect to
DOF	degree of freedom
UKL	University of Kassel, Department of Civil Engineering, Laboratory of Lightweight Structures and Structural Mechanics
FEM	finite element method
MATLAB	high performance numeric computation and visualization software, The MATHWORKS Inc. Natick, MA, USA
MDB	MATFEM data basis
BNP	basic nodal point
LNP	local nodal point
X, Y, Z, XX, YY, ZZ	DOF direction w.r. to the global coordinate system
x, y, z, xx, yy, zz	DOF direction w.r. to the local coordinate system

U_X, U_Y, U_Z	translational DOF
U_{XX}, U_{YY}, U_{ZZ}	rotational DOF
F_X, F_Y, F_Z	nodal forces
F_{XX}, F_{YY}, F_{ZZ}	nodal forces (moments)
σ_x, σ_y, σ_z	normal stresses
σ_{xy}, σ_{xz}, σ_{yz}	shear stresses
NP	nodal point
CS	coordinate system

Frequently used MATFEM Fixed Name Variables

nbalk	overall number of beam elements
nshel	overall number of shell elements (nshel = nshel3 + nshel4)
nshel3	overall number of 3 node shell elements
nshel4	overall number of 4 node shell elements

Right-handed screw rule

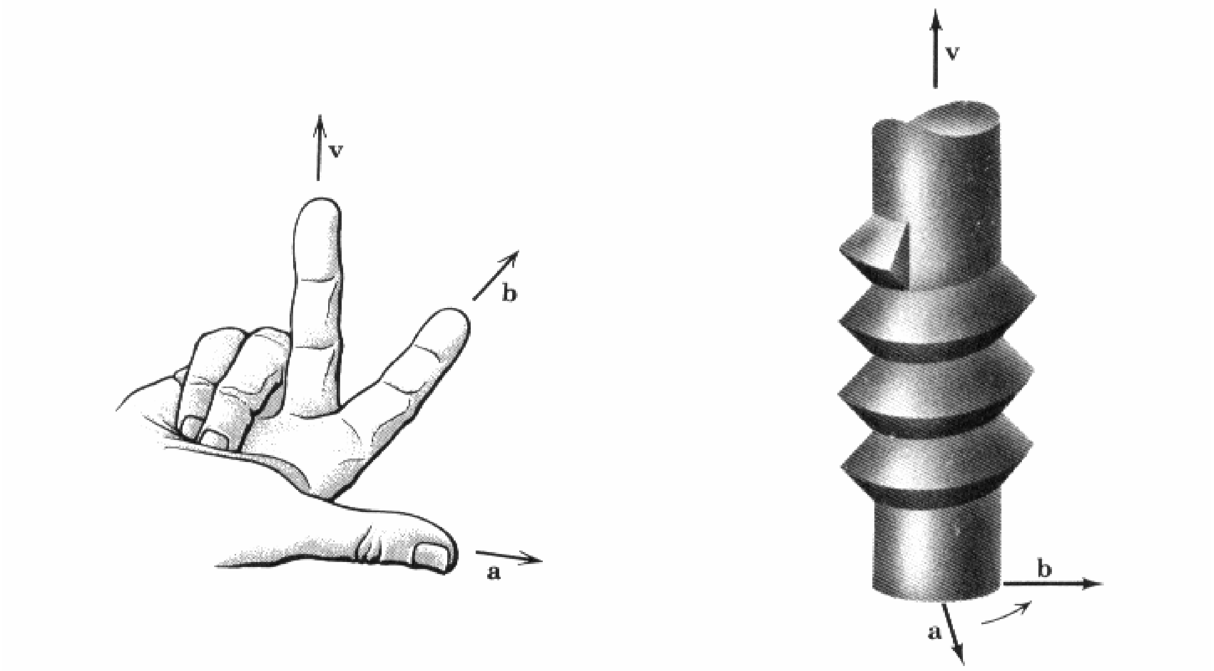


Fig. 0.1: Right-handed triple of vectors, right-handed screw rule, [Kreyszig 1]

General

Generally, the term ‘nodal forces’ can either imply nodal forces or nodal moments. The term ‘degree of freedom DOF’ can imply translational and/or rotational degrees of freedom.

One-dimensional arrays must generally be specified as row vectors.

Due to the original program development of MATFEM is done by German authors and therefore variable names used in MATFEM are often based on German expressions, this documentation uses both German and English terms where helpful.

Main New Features of MATFEM02

preprocessor:

analysis:

- calculation of non-linear frequency dynamic response using modal condensation during response calculation to speed up the calculation time (wahl = 8) (using the 'Harmonic Balancing' linearization approach)
- dynamic condensation has been improved for non-linear response calculation (wahl = 8) (using the 'Harmonic Balancing' linearization approach)
- additional non-linear elements (Friction type, clearance type, arctan-stiffness, arcsinh-stiffness, Gaul element (2 parameters)) (wahl = 8) (using the 'Harmonic Balancing' linearization approach)

postprocessor:

general:

- the subdirectory \issp has been removed from the MATFEM path. All former functions of this subdirectory are now included in the MATFEM main path \src

- the subdirectory \toolbox has been removed from the MATFEM path. All former functions of this subdirectory are now included in UKL's TBOX software package. The TBOX is general UKL toolbox used by different UKL software packages like ISSPA and UPDATE

The user can use the input file structure of older MATFEM versions but it is recommended to use the actual input file structure, as given in the *template* directory, for future applications.

Main New Features of MATFEM04

preprocessor:

analysis:

- real and complex eigensolutions
- MATLAB build-in eigensolution solver *eigs* is now available for dynamic analysis (ieig=5)
- eigenvector 'blowup' feature now available for external data (real or complex eigensolutions) given in Universal File format (type 15, 82,55)
- BNP labels
- accumulation of damping input (damptyp= 4) in the case of proportional damping $\mathrm{D} = \alpha \mathrm{K} + \beta \mathrm{M}$ and direct D matrix specification (e.g. used for dashpot dampers)

postprocessor:

- The *MATFEM Plot Panel* helps to easily modify the plot of the undeformed structure and to gain detailed information about nodes, elements, boundary conditions, etc.
- complex mode visualization (animation)

general:

- the UKL TBOX is now divided into the subdirectory \src, \exe, \pcode . Some new routines have been added to handle data given in Universal File Format. The TBOX is a general toolbox used by different UKL software packages like ISSPA and UPDATE

The user can use the input file structure of older MATFEM versions but it is recommended to use the actual input file structure, as given in the *template* directory, for future applications.

Upgrading from earlier MATFEM releases: Changes and Enhancements

1. MATFEM Installation

MATFEM requires that MATHWORKS/MATLAB Version 5.3 or higher is installed on the computer (PC or UNIX workstation).

MATFEM is distributed on a media using the following directory structure:

```
                              ukl\matfem
   --------------------------------I----------------------------
  I         I           I          I          I          I
 src       demo      template     docu      pcode       link
```

contents:

src	MATFEM main source (MATLAB script files)
demo	MATFEM input files for several demonstration examples (ref. Appendix D)
template	Templates of MATFEM input files (ref. Appendix C)
docu	MATFEM Documentation (Microsoft WORD 97 documents)
pcode	Precompiled MATFEM source files to be used within MATLAB5.3 or higher
link	Special MATLAB script files to link MATFEM to different UKL software packages e.g. UKL/UPDATE_B

In addition the media includes the UKL TBOX **,** a special software package which includes general data handle tools which are commonly used by several UKL software packages. The UKL TBOX has the following directory structure:

```
                      ukl\tbox
               -------------I---------------
                 I          I           I
                src       pcode        exe
```

contents:

src	TBOX main source (MATLAB script files)
pcode	Precompiled TBOX source files to be used within MATLAB5.3 or higher
exe	TBOX FORTRAN executables

Note:
Your implementation of MATFEM and the UKL TBOX may differ from this general directory structure due to individual requirements.

To run MATFEM within MATLAB the MATLABPATH has to be extended to include at least the following directories:

- ukl\matfem\src or ukl\matfem\pcode
- ukl\tbox\src or ukl\tbox\pcode

Note:
If only the *pcode edition* or the *limited edition* of MATFEM is available the user must link \pcode directory to the MATLABPATH.

Refer to the MATLAB User’s Guide to set the MATLABPATH.

The user must also set the operating system program search path to include ukl\tbox\exe. Refer to the User's Manual of your computer to properly set the program search path.

To check the MATFEM installation it is recommend to copy the *demo* directory to a temporary working directory and start the auto-sequence of demonstration examples by calling *demo_all* within MATLAB from the actual demo directory.

Setting the MATFEM editor

The user must set the MATFEM editor in the file *ed.m* in the ukl\tbox\src or the ukl\tbox\pcode directory. The editor is invoked by the preprocessor to edit the MATFEM input files or directly from the MATFEM main window to edit the MATFEM listing file *.aus. The user must edit the *ed.m* file and set the correct destination of the editor of his system w.r. to the operating system of his computer.

```
% Edit a MATLAB m-file with a system Editor.

function ed(filename)

system = computer;

if (nargin == 0)

  [filename,curpath] = uigetfile('*.m','Please select a m-file');

  if isstr(filename)
    if ~strcmp(system,'VAX_VMSD')
      curpath  = lower(curpath(1:length(curpath)-1));
    end
    filename = lower(filename);
  else
    return
  end

else

  curpath = cd;
  curpath = lower(curpath);

  if isempty(find(filename=='.'))
    filename = lower([filename,'.m']);
  end

end

if     strcmp(system,'PCWIN')                                % PC with Windows
  eval(['!notepad ',curpath,'\',filename])

elseif strcmp(system,'IBM_RS')                               % IBM RS6000
%  eval(['!nedit ',curpath,'/',filename])
  eval(['!xedit ',curpath,'/',filename])

elseif strcmp(system,'SGI')                                  % Silicon Graphics
%  eval(['!ieditor ',curpath,'/',filename])
  eval(['!editor ',curpath,'/',filename])

elseif strcmp(system,'SOL2')                                 % DLR Sun Solaris 2
  eval(['!textedit ',curpath,'/',filename])

elseif strcmp(system,'VAX_VMSD')                             % VAX
  eval(['!edit ',curpath,filename])

else
  error('Please add your system''s editor to file: ed.m !')
end

return
```

Fig. 1.1: Setting the MATFEM editor in the file ukl\tbox\src\ed.m or ukl\tbox\pcode\ed.m

2. General

MATFEM is a Finite Element code for linear static and dynamic analysis. It is developed by the department of lightweight structures and structural mechanics at the department of civil engineering at the University of Kassel (UKL), Prof. Dr. Ing. M. Link. MATFEM is written using MATHWORKS/MATLAB [MATLAB 1] computation and visualization software. MATFEM is aimed to be used within the fields of research and education to solve low to medium scaled static and dynamic problems up to ≈ 8000 degrees of freedom (DOF) which yields reasonable CPU times for Pentium IV PC’s. The developers of MATFEM put more emphasis on the solution algorithms and the variety of applications than on graphical user interface programming and the evaluation of routines to check the conformity and completeness of input data specification. Due to this, the user has to take a high share on specifying all needed input data correctly.
MATFEM expects the user to be a so called ‘friendly user’, i.e.

- he/she has a basic knowledge in mechanics and FEM to supply the needed input data and interpret the output results
- he/she has a basic knowledge in using MATHWORKS/MATLAB

MATFEM is certainly not designed for users who want to proof how stupid software can be.

Like any FEM program MATFEM consists of three main modules:

preprocessor (editing of input files, no graphical support)
analysis
postprocessor (visualization of analysis results)

To specify the input for a MATFEM analysis a maximum of 20 ASCII input files can be provided. These files are standard MATLAB script files. They all share a common file name (marked by * throughout this documentation) but have different file extensions. The common file name is specified by the user and may be any name, e.g: demo1 . The file extensions are fixed to *.m01 ... *.m20, e.g:

```
demo1.m01
    ...
    ...
demo1.m20
```

It is recommended to use the MATFEM input file templates to specify all needed data.
MATFEM is started within MATLAB by using the command: **matfem** or **mfstart**

For very special purposes (e.g. data debugging) it may be helpful to start MATFEM using the command: **mfein .** If so, the user has direct access to some main MATFEM variables after the analysis has terminated.

The commands **matfem** or **mfstart** invoke the main MATFEM menu from which the three main modules

preprocessor (editing of input files)
analysis
postprocessor (visualization of analysis results)

are accessible.

At each analysis run an ASCII file is created in which parts of the input data, information about the program flow and the analysis results are listed. This file serves as the major results listing file and is accessible directly from the MATFEM main window (button: *edit .aus*). The common file name is assigned to this file and the file extension is fixed to *.aus, e.g: *demo1.aus* .

In addition, MATFEM creates data files which include additional and intermediate results. These files also carry the common file name but with different file extensions depending on the contents of each file, e.g.

demo1.mtx (system matrices)
demo1.mod (modal parameters)

The files serve as a data basis for the interface analysis/postprocessor or provide access to analysis results for further user defined calculations. These files are referred as the MATFEM data basis (MDB). In general, these files are not used by the user. For detailed information about the contents, type and structure of the files refer to Appendix B. The files are written in MATLAB *.mat binary format. Refer to the MATLAB User's Guide for detailed information about the *.mat file format.

Using the command **matfem** or **mfstart,** all variables used in MATFEM are local variables, which are cleared after the analysis has terminated. However, the variables defined in any of the input files will be treated as local variables of the MATFEM analysis and therefore be part of the analysis. Therefore, care should be taken to properly choose variable names and define input parameters within the input files. Refer to chapter 4 and Appendix A for more details.

NOTE:

The user must take care about the unit system used for input and output data. Therefore it is strongly recommended to specify all input data in SI units i.e. mass [kg], dimensions [m], time [s], force [N]. Hence, all output quantities will also be in SI units i.e displacement [m], stresses [N/m^2], eigenfrequency [Hz], ... etc.

3. Input Data Files

The input parameters of a MATFEM analysis can be specified in up to 20 ASCII input files. These files share a common file name but have different file extensions (*.m01 ... *.m20) and must be within the same directory. For detailed information about the contents of the input files, refer to Appendix C and D and also to the templates and examples given in the TEMPLATE and DEMO directory.

Contents of input data file

*.m01	general MATFEM control parameters
*.m02	--- *user defined* ---
*.m03	--- *user defined* ---
*.m04	meshing parameters
*.m05	--- *user defined* ---
*.m06	material properties
*.m07	beam properties
*.m08	shell properties
*.m09	boundary conditions
*.m10	static condensation/ CRAIG-BAMPTON substructure coupling
*.m11	grounded spring elements
*.m12	--- *user defined* ---
*.m13	static analysis, loads
*.m14	dynamic analysis, loads, response
*.m15	--- *user defined* ---
*.m16	--- *user defined* ---
*.m17	--- *user defined* ---
*.m18	--- *user defined* ---
*.m19	--- *user defined* ---
*.m20	direct element parameter and matrix modification. Only to be used by highly experienced MATFEM users and MATFEM program developers)

This input file structure is different to older MATFEM releases where always sixteen input files had to be specified for any application. In addition, all of the MATFEM input variables had to be specified, regardless of the application type or whether they were needed or not. In MATFEM04 all variables are automatically set to initial values and the user only has to specify the input needed for the actual application. However, input files according the old input file structure can be used with MATFEM04 and should run without any modifications.

Due to the input file handling of MATFEM, the structures of the input files can be specified by the user to a large extend. The input files are read in ascending order in a two step procedure.

1. Input files *.m01 - *.m05 are read. They must contain some basic control parameters (e.g. wahl, iplt) and all meshing parameters. The files are read one by one and there is no data processing of any kind during the read process. Input file *.m01 must always exist. If one of the other input files does not exists the read process skips to the next file. Therefore it would be possible to specify all needed input in a single input file (e.g. *.m01), but this would be rather confusing. After reading these five input files the mesh is generated and plotted, if specified.
2. Input files *.m06 - *.m20 are read. These files contain all other MATFEM input (e.g. element properties, loads,...) The files are read one by one and there is no data processing of any kind during the read process. If any of the input files does not exist the read process skips to the next file. Again, it would be possible to specify all needed input in a single input file (e.g. *.m06), but this would be even more confusing.

To gain a clear structure for the input data it is best to use the file structure presented in the TEMPLATE directory. The MATFEM preprocessor supports this file structure in first place. But it also supports the older file structure by demand. However, it is up to the user to modify this file structure if needed, as long as the two step input file read procedure is kept.

Using already existing input files as templates for a new application

For a new application it is always recommended to fill out the respective template files of TEMPLATE directory. But it is also possible to use already existing input files of a different application, e.g. of the DEMO directory, as templates for a new application. This can be managed in two different ways:

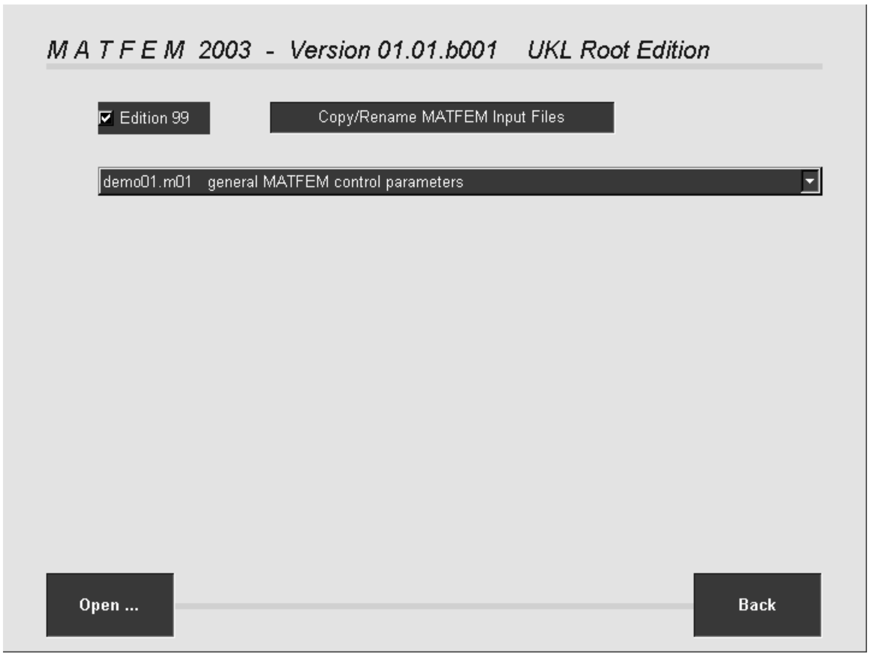
1. a.) Copy the template input files or the already existing input files to the actual working directory using the respective tools of the operating system, e.g. WINDOWS Explorer

   b.) Rename these files so that they all share the common file name of the new application

   c.) Edit the input files using the MATFEM Preprocessor and specify/modify all needed input for the new application.

2. a.) Start MATFEM and select an already existing application as the actual model file (e.g. from the TEMPLATE or DEMO directory).

   b.) Start the MATFEM Preprocessor so that the main MATFEM Preprocessor window becomes accessible



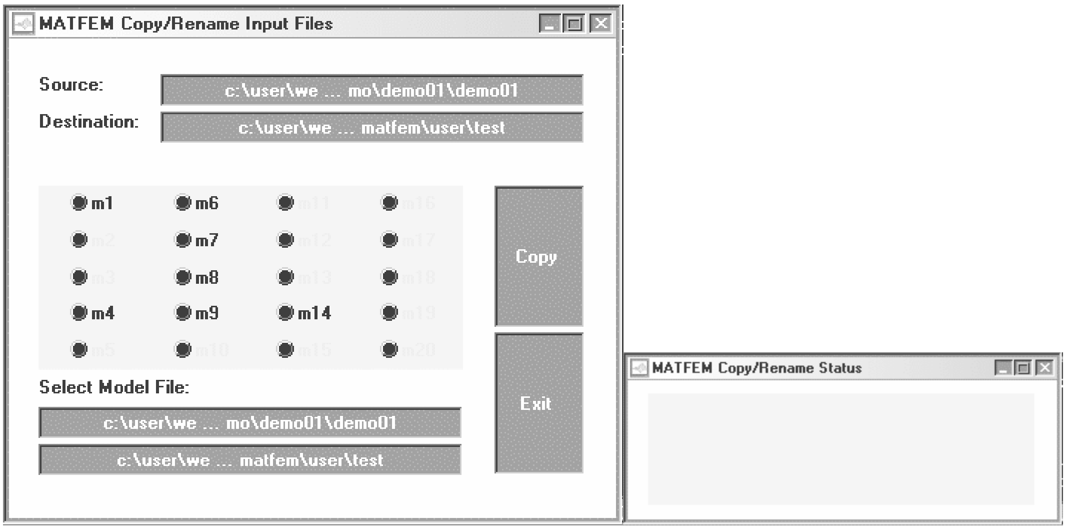
M A T F E M 2003 - Version 01.01.b001 UKL Root Edition

☑ Edition 99

demo01.m01 general MATFEM control parameters

Fig. 3.1: Main window of the MATFEM preprocessor

   c.) Press the "Copy/Rename MATFEM Input Files" button. The "MATFEM Copy/ Rename Input Files" window and a status window will appear.



MATFEM Copy/Rename Input Files

Source: c:\user\we ... mo\demo01\demo01

Destination: c:\user\we ... matfem\user\test

Select Model File:

c:\user\we ... mo\demo01\demo01

c:\user\we ... matfem\user\test

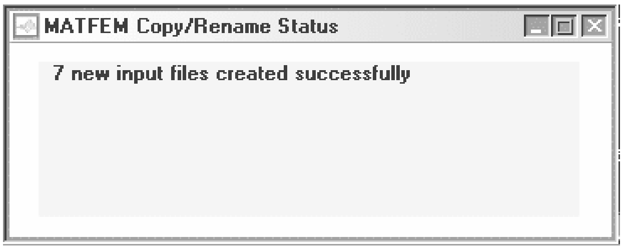
MATFEM Copy/Rename Status

Fig. 3.2: The MATFEM Copy/Rename Input Files main and status window

d.) Select a source file by pressing the *source* button. The actual model file is selected by default

e.) Select the destination i.e. the new model file name and the respective directory by pressing the *destination* button (the actual model file is selected by default). At this destination the new input files will be created. They will all share the specified common file name.

f.) Press the *copy* button to start the copy/rename task. The *status* window will notify the status of the copy task.



7 new input files created successfully

Fig. 3.3: The MATFEM Copy/Rename Status window after the successful termination of the copy task

g.) Exit from the copy/rename task by

1. pressing the *exit* button. The original working directory and model file is <u>not</u> altered i.e. the actual working directory and model file is retained as it was set for the MATFEM preprocessor.
2. pressing one of the *select model file* buttons (fig. 3.2). The actual working directory and model file will be altered to the respective destination:

   a.) upper button - source file specification

   b.) lower button - destination file specification

d.) Edit the input files using the MATFEM Preprocessor and specify/modify all needed input for the new application.

Note: The user must take high care to specify/modify all needed input for the new application properly. If he/she uses a copy of already existing input files chances are high that he/she will miss the setting of at least one important parameter. So the MATFEM listing file **.aus* of the new application should always be checked first in every detail that all settings of the new application are correct. To keep this error source in mind will definitely save a large amount of time in the error localization process if the MATFEM analysis fails or the MATFEM results are obviously incorrect.

4. Fixed Name Variables used within MATFEM

MATFEM uses a large number of variables which names are fixed and must not be modified by the user. To specify the input for a MATFEM analysis the parameters of some of these variables must be assigned. The input is assigned directly to the fixed name variables within the twenty input files. The input files are treated as standard MATLAB *.m script files. Hence it is possible to perform additional computing of any kind within the input files. But this additional computing is restricted:

> The user **must not** assign or delete MATFEM fixed name variables for subsequent computing in any other context than the assignment of MATFEM input.

However, in any application it is advisable to use the MATLAB programming language intensively to create or modify the input data conveniently. This implies the call of user specified subroutines or MATLAB script files within the input files. If applied, the user has to take high care that the user defined variable names do **not** match any of the fixed name variables of MATFEM. Therefore, it is recommended to use special characters in the user defined variable names, e.g *h__up* or *m_j_k*. The user should also check the table in Appendix A which summarizes the most important fixed name variables used in MATFEM. The list is not complete and subject to change with every future MATFEM update.

Note

It is recommended that the user should not use the MATLAB *load* command in the MATFEM input files as the user will not be notified which variables are actually read in. This is especially important if the user imports data from an existing MATFEM data basis in a MATFEM input file via the MATFEM load command. In this special case the user should use the special MATFEM input routines which exist for every file of the data basis, e.g *mf_r_mtx*, *mf_r_pld*.

5. Meshing

Generally, it requires three basic steps to generate a FEM mesh

1. Allocation of nodal points and elements
2. Allocation of a material and geometrical property data base
   (e.g. YOUNG's modulus, thickness, cross section, torsional moment of inertia)
3. Assignment of material and geometrical data to elements

To create a FEM mesh MATFEM first uses geometry elements which in a second step will be broke up into a number of structural FEM elements like beams and shells.

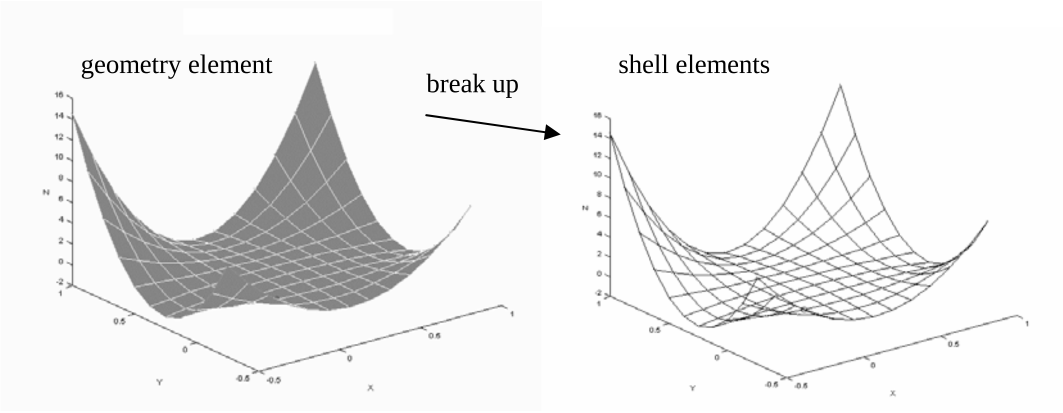


Fig. 5.1: Geometry element / FEM element

In MATFEM a FEM mesh is created according the following steps

1. Allocation of nodal points and elements *(specified in *.m04)*

a) Allocation of **basic nodal points** (BNP), which will be used to define the geometry elements (Bereichskoordinaten)

b) Allocation of the **geometry elements** (e.g. fla9, linien) which then will be broken up to generate nodal points and structural FEM elements (beam, shell). Geometry elements are defined by BNP's. There are

- patches
- axis-symmetry areas

- grillages
- axis-symmetry grillages
- rib areas
- lines.

Generally, patches and axis-symmetry areas are related to 3 and/or 4 node <u>shell</u> elements. Grillages, axis-symmetry grillages, rib areas and lines are related to <u>beam</u> elements.

c) Specification of fraction ratios to break up the overall geometry of each geometry element into a set of refined subdivisions which finally yield the **structural FEM elements** (beams, shells). The fraction ratios refer to the local coordinate system (η and ξ axis) of each geometry element. In addition, special parameter vectors are defined to specify general geometry element parameter and element break up rules.

d) Specification of the coincidence of geometry elements and element cards by vectors ec_*_koi. Therefore all elements derived from a geometry element are assigned to the same element card. There is an assignment vector for each type of geometry element (e.g. fla9 patch: ec_f_koi, ribs: ec_rip_koi)

The overall mesh is then derived from the geometry elements and their respective refinement. MATFEM automatically manages node and element numbering from input a)-c).

2. *Allocation of a material and geometrical property data base*
*(specified in *.m06, *.m07, *.m08)*
MATFEM uses parameter vectors to define geometrical and material properties, e.g. YOUNG’s modulus *E*, shell element thickness *dicke*. They serve as a kind of data base which will be referenced in step 3. The user may even specify parameters which are not needed in the actual application.

3. *Assignment of material and geometrical properties to element cards*
*(specified in *.m07, *.m08)*
MATFEM uses the matrices *ec_beam* and *ec_shell* to specify element cards for beam and shell elements respectively. Each column represents an element card which is assigned to the beam and shell elements by the geometry element coincidence vector *ec_*_koi*.

Previous MATFEM releases used beam and shell assignment vectors for each element property, e.g. *ihooke*, *idicke*. The introduction of element cards does not change this basic idea. In fact, all properties assigned in an element card are assigned automatically to the corresponding beam and shell assignment vectors. The element cards are especially useful for a mesh refinement of a model which consists of several beam and shell areas with different properties. However, the user may either use the beam and shell assignment vectors or the element cards.

5.1. Allocation of nodal points and elements *(specified in *.m04)*

5.1a) Allocation of basic nodal points (BNP)

The only coordinates of nodal points a user has to specify are those of the BNP’s (Bereichskoordinaten). These nodal points are common to any subsequent meshing. The BNP’s are specified in a (n,3) matrix called *berkoord* (file *.m04), where n specifies the overall number of BNP’s . The user may also specify *berkoord* as a (n,4) cell array where the 4-th column is the BNP label. The coordinates refer to a global Cartesian coordinate system. Each row specifies the X,Y,Z coordinates of a BNP. The row number of *berkoord* is automatically assigned to the BNP as the nodal number. So the first row in *berkoord* defines nodal point #1, the second nodal point #2, ... etc. The BNP label is automatically set to BNP 1, BNP 2, ...etc, if not specified. The user may also specify more BNP's then actually needed to define the element geometry. BNP's may also be coincident, i.e. may have equal X,Y,Z coordinates.

The mesh generating routines independently create the matrix of node coordinates *koord* from the geometry elements i.e. the specified BNP’s will not generally be included into *koord*. However *koord* will include nodal points which coordinates coincide with the BNP’s of *berkoord,* if the BNP’s are consistent with the geometry element break up rule.

The user can force MATFEM to renumber the nodal points of *koord* so that the nodal points of *berkoord* are kept by setting *bkeep* in *.m04.

Note: All nodal referred input and output, e.g forces or displacement, refer to the nodal points and numbering of *koord*. The BNP’s and *berkoord* are only used to generate the mesh.

Therefore the user always has to check that all nodal assignment is specified properly. The listing file *.aus provides detailed information about the nodal point coincidence. It is also recommended to set *bkeep= 1* in *.m04 to keep the BNP numbering of *berkoord* in *koord*. Any modifications of the original BNP sequence will then be prompted to the user and listed in detail in the *.aus file.

5.1b) Allocation of geometry elements

There are eight types of geometry elements in MATFEM to conveniently support complex meshing:

• *fla9*	patch	to generate 3 and 4 node shell elements
• *flarot*	axis-symmetry area	to generate 3 and 4 node shell elements
• *tria6*	patch	to generate 3 node shell elements
• *triarot*	axis-symmetry areas	to generate 3 node shell elements
• *roste*	grillage	to generate linear 2 node spring and beam elements and non-linear 2DOF truss elements
• *rostrot*	axis-symmetry grillage	to generate linear 2 node spring and beam elements and non-linear 2DOF truss elements
• *linien*	line	to generate linear 2 node spring and beam elements and non-linear 2DOF truss elements
• *rippen*	rib	to generate linear 2 node spring and beam elements and non-linear 2DOF truss elements

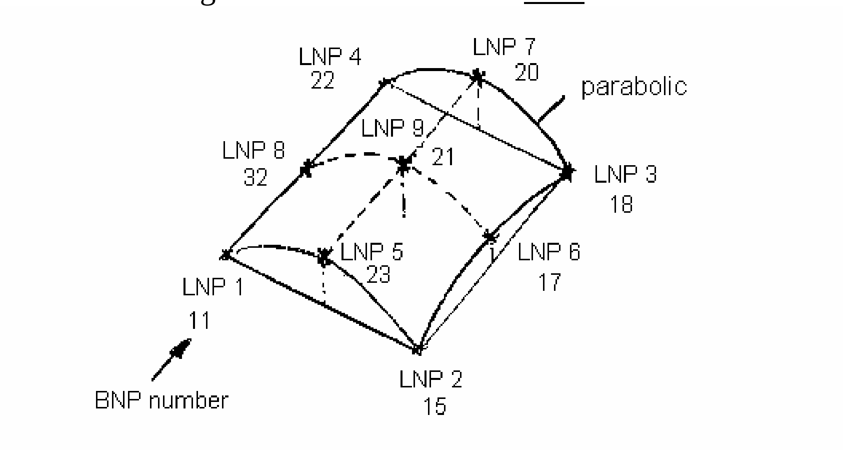
Note: It is recommended to use *linien* for definition of non-linear 2DOF truss elements !

The edges and respectively the generating lines of the geometry elements can generally be specified as straight or parabolic curved lines

Note: Do not use parabolic curved lines for definition of non-linear 2DOF truss elements !

- *fla9* patch

fla9 patches are defined by a *(m_p ,9)* matrix of BNP numbers called *fla9,* where m_p specifies the overall number of fla9 patches. *fla9* may also be defined as a *(m_p ,9)* cell array, if BNP labels are used. Each patch is defined by 1...9 local nodal points (LNP) which are selected by the user as a subset of the BNP's.
Note: Patches are used to generate 3 and/or 4 node shell elements.



Note: The partition geometry /structural element is set by *xsif, etaf* , refer to chapt. 5.1.c

Fig. 5.2: a single **fla9** patch specified by LNP 1 ... LNP9

A *fla9* patch can represent a parabolic curved surface in 3-D space. A quadrilateral patch is defined by specifying LNP (1 2 3 4) only and setting LNP (5 6 7 8 9) to zero. For a patch with parabolic shaped edges the LNP (5 6 7 8) of the corresponding edge must be specified. LNP 9 may be used to distort the mesh perpendicular to its curved surface. If LNP (5 6 7 8) are zero the LNP’s are located in the middle of the edges. If LNP (9) is zero LNP (9) is located at the coordinate average of the 4 corner nodes.

Note: A *fla9* patch must always be numbered in counterclockwise direction.

Specific parameters for the element generation e.g. type of elements are set for each element of *fla9* by specifying the parameter matrix *fla9par*:

fla9par(:,1) element type
= 3 3- node element
= 4 4- node element (default)
fla9par(:,2) type of 3 node break up pattern for the fla9 patch
= 0 none
= 1 quadrant
= 2 line by line w.r. to xsi/eta coordinate system
fla9par(:,3:6) 4 element integer vector to specify the pattern design
for each quadrant. Each integer specifies the left (1)
or right hand (2) break up rule of the corresponding
4 node sub patch

```
      4-------3  yields          4-------3  yields
      ! #     !  triangles:      !     # !  triangles:
  1   !   #   !  1 2 4       2   !   #   !  1 3 4
      !     # !   and            ! #     !   and
      1-------2  2 3 4           1-------2  1 2 3
```

Example 5.1:

Generate a *fla9* patch of a given structure:

1) a plane patch between BNP(11 15 18 22) = LNP(1 2 3 4),
4 node shell elements

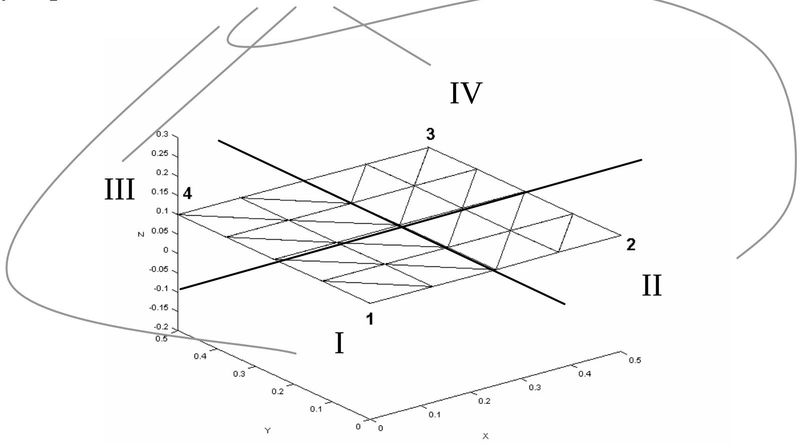
fla9 = [11 15 18 22 0 0 0 0 0]
fla9par = [4 0 0 0 0 0]

2) a patch between BNP(11 15 18 22) = LNP(1 2 3 4) with curved edges and the center point 21 above the average of the 4 corner points, as in fig. 5.2 ,
4 node shell elements

fla9 = [11 15 18 22 23 17 20 32 21]
fla9par = [4 0 0 0 0 0]

3) a plane patch between BNP((1 2 3 4) = LNP(1 2 3 4), 3 node shell elements, quadrant break up pattern, acc. fig. 5.3

fla9 = [1 2 3 4 0 0 0 0 0]
fla9par = [3 1 1 2 1 2]

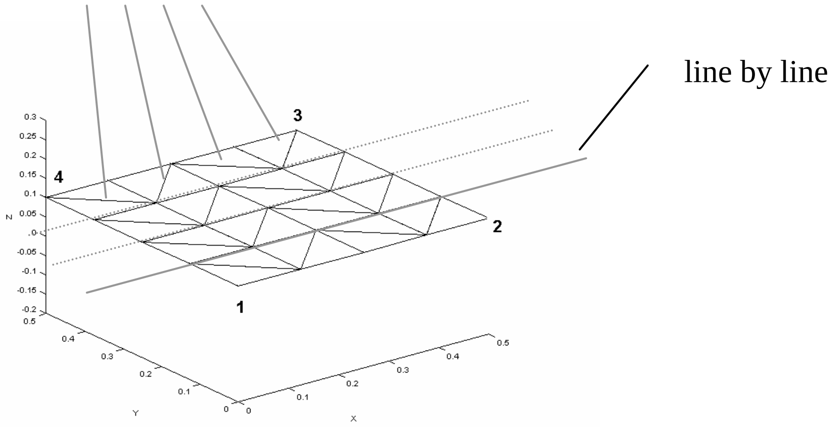


Note: The partition geometry /structural element is set by *xsif*, *etaf*, refer to chapt. 5.1.c

Fig 5.3: A plane patch to generate 3 node shell elements, quadrant break up pattern.

4) a plane patch between BNP((1 2 3 4) = LNP(1 2 3 4), 3 node shell elements, line by line break up pattern, acc. fig. 5.4

fla9 = [1 2 3 4 0 0 0 0]
fla9par = [3 2 1 2 1 2]



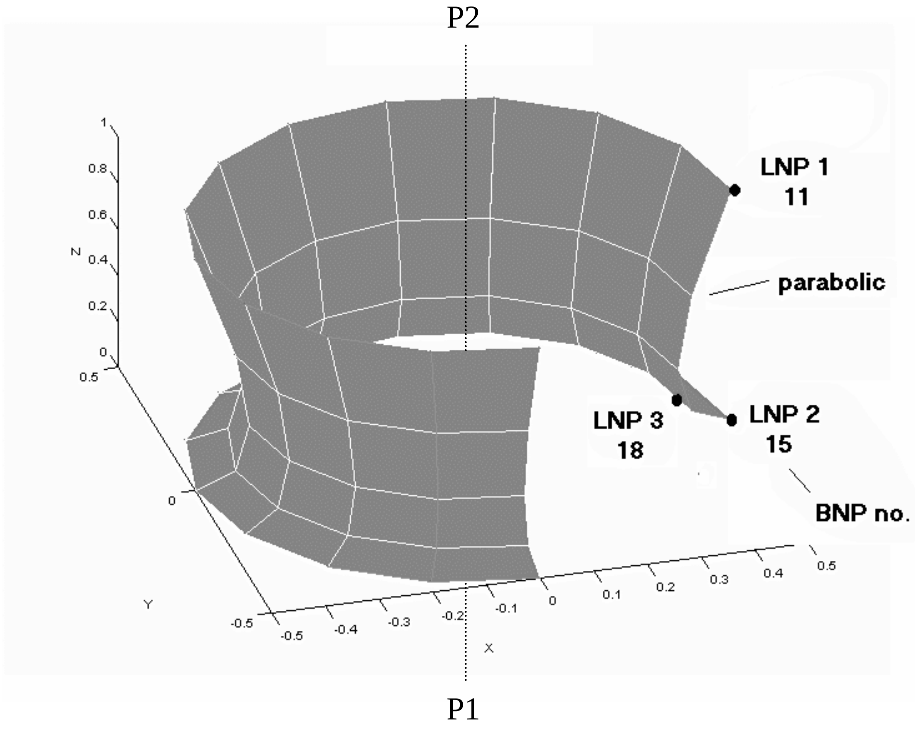
Note: The partition geometry /structural element is set by *xsif*, *etaf* , refer to chapt. 5.1.c

Fig 5.4: A plane patch to generate 3 node shell elements, line by line break up pattern

- *flarot* axis-symmetry area

flarot axis-symmetry areas are defined by a *(m_a ,3)* matrix of BNP numbers called *flarot,* where m_a specifies the overall number of axis-symmetry areas. . *flarot* may also be defined as a *(m_a ,3)* cell array, if BNP labels are used. Each area is defined by 1...3 local nodal points (LNP) which are selected by the user as a subset of the BNP's. These nodal points specify the contour of the area.

Note: *flarot* axis-symmetry areas are used to generate 3 or/and 4 node shell elements.



Note: The partition geometry /structural element is set by *xsimantel, etaumfang* , refer to chapt. 5.1.c

Fig. 5.5: a single *flarot* axis-symmetry area specified by LNP1 .. LNP3

An *flarot* axis-symmetry area can represent a curved surface in 3-D space. A straight contour is defined by specifying LNP (1 2) only and setting LNP (3) to zero. A parabolic shaped contour can be defined using LNP (1 2 3).
Specific parameters for the element generation e.g. rotation axis, type of element are set for each element of *flarot* by specifying the parameter matrix *flarotpar*:

flarotpar(:,1)	X coordinate of P1 to define rotation axis
flarotpar(:,2)	Y " of P1 " " " "
flarotpar(:,3)	Z " of P1 " " " "
flarotpar(:,4)	X coordinate of P2 to define rotation axis
flarotpar(:,5)	Y " of P2 " " " "
flarotpar(:,6)	Z " of P2 " " " "

flarotpar(:,7) assignment direction of element node coincidence
= 0 negative to flarot(:,1) -> flarot(:,2) direction
= 1 in flarot(:,1) -> flarot(:,2) direction

flarotpar(:,8:10) -- reserved for future use --

flarotpar(:,11) element type
= 3 3- node element
= 4 4- node element (default)

flarotpar(:,12) type of 3 node break up pattern for the flarot patch
= 0 none
= 1 quadrant
= 2 line by line w.r.t. to xsi/eta coordinate system

Note: For the definition of the pattern the *flarot* patch is treated as unrolled and thereby equivalent to a *fla9* patch, ref. figs. 5.3, 5.4

flarotpar(:,13:16) 4 element integer vector to specify the pattern design for each quadrant. Each integer specifies the left (1) or right hand (2) break up rule of the corresponding 4 node sub patch

```
     4-------3  yields            4-------3  yields
     ! #     !  triangles:        !     # !  triangles:
1    !   #   !  1 2 4        2    !   #   !  1 3 4
     !     # !   and              ! #     !   and
     1-------2  2 3 4             1-------2  1 2 3
```

P1 is the origin of the rotation axis.

Example 5.2:

Generate a *flarot* axis-symmetry area of a given structure:

1) a *flarot* axis-symmetry area with a plane contour between
BNP(11 15) = LNP(1 2), rotation axis: global Z axis, 4 node elements

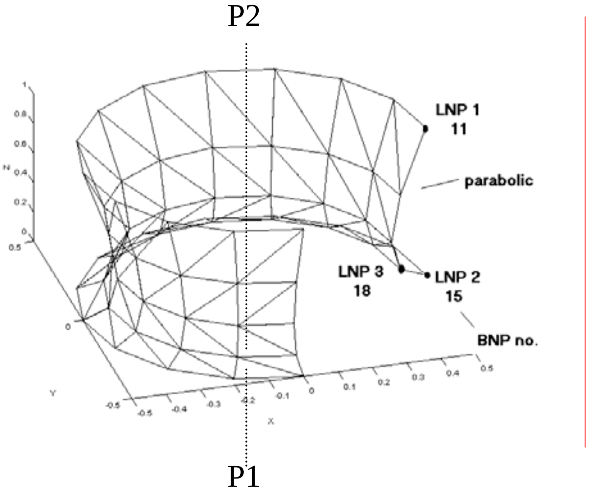
flarot = [11 15 0]
flarotpar = [0 0 0 0 0 1 1 0 0 0 4 0 0 0 0 0]

2) a *flarot* axis-symmetry area with a parabolic contour between
BNP(11 15 18) = LNP(1 2 3) , as in fig. 5.5
rotation axis: global Z axis, 4 node elements

flarot = [11 15 18]
flarotpar = [0 0 0 0 0 1 1 0 0 0 4 0 0 0 0 0]

3) a *flarot* axis-symmetry area with a parabolic contour between BNP(11 15 18) = LNP(1 2 3) , rotation axis: global Z axis, 3 node shell elements, quadrant break up pattern, acc. fig. 5.6

flarot = [11 15 18]
flarotpar = [0 0 0 0 0 1 1 0 0 0 3 1 1 2 1 2]

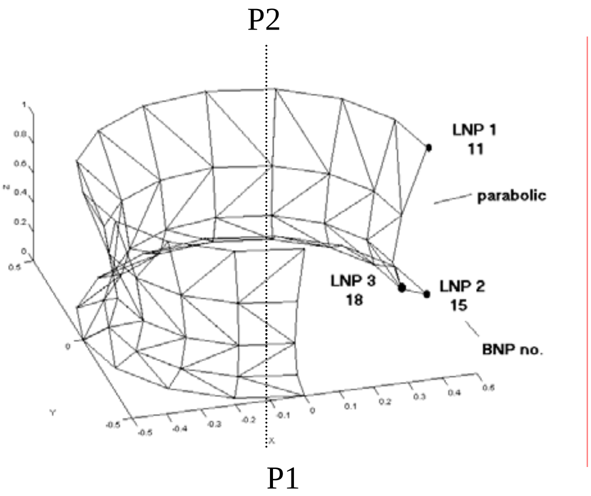


Note: The partition geometry /structural element is set by *xsimantel*, *etaumfang* , refer to chapt. 5.1.c

Fig 5.6: A *flarot* patch to generate 3 node shell elements, quadrant break up pattern

4) a *flarot* axis-symmetry area with a parabolic contour between BNP(11 15 18) = LNP(1 2 3) , rotation axis: global Z axis, 3 node shell elements, line by line break up pattern, acc. fig. 5.7

flarot = [11 15 18]
flarotpar = [0 0 0 0 0 1 1 0 0 0 3 2 1 2 1 2]



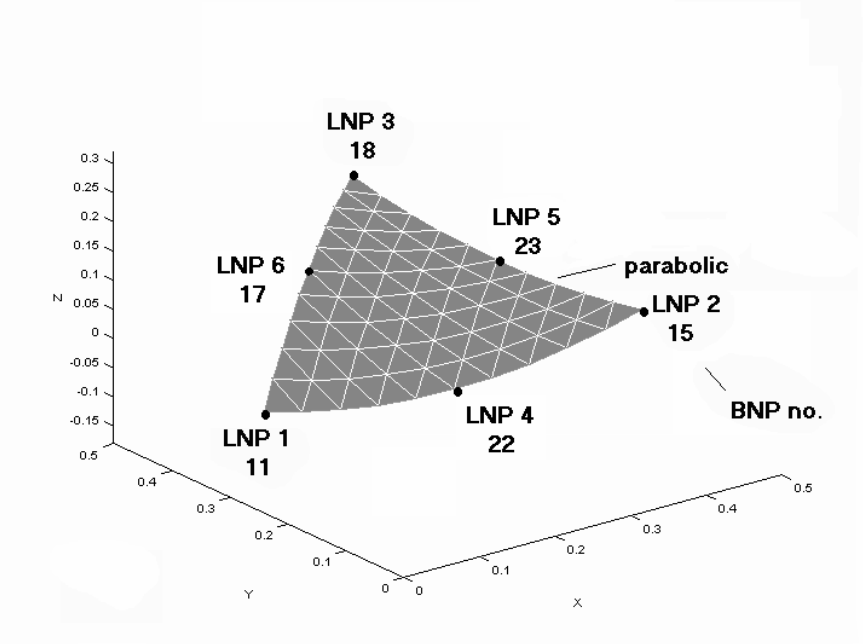
Note: The partition geometry /structural element is set by *xsimantel, etaumfang,* refer to chapt. 5.1.c

Fig 5.7: A flarot patch to generate 3 node shell elements, line by line break up pattern

- *tria6* patch

tria6 patches are defined by a $(m_p ,6)$ matrix of BNP numbers called *tria6*, where m_p specifies the overall number of tria6 patches. *tria6* may also be defined as a $(m_p ,6)$ cell array, if BNP labels are used. Each patch is defined by 1...6 local nodal points (LNP) which are selected by the user as a subset of the BNP's.

Note: *tria6* patches are used to generate 3 node shell elements. They are especially designed to allow local mesh refinements i.e. to link coarse mesh areas to fine mesh areas.



Note: The partition geometry /structural element is set by *xsit*, refer to chapt. 5.1.c

Fig. 5.8: a single *tria6* patch specified by LNP 1 ... LNP6

A *tria6* patch can represent a parabolic curved surface in 3-D space. A plane patch is defined by specifying LNP (1 2 3) only and setting LNP (4 5 6) to zero. For a patch with parabolic shaped edges the LNP (4 5 6) of the corresponding edge must be specified. If LNP (4 5 6) are zero the LNP's are located in the middle of the edges.

Note: A *tria6* patch must be numbered in counterclockwise direction.

Example 5.3:

Generate a *tria6* patch of a given structure:

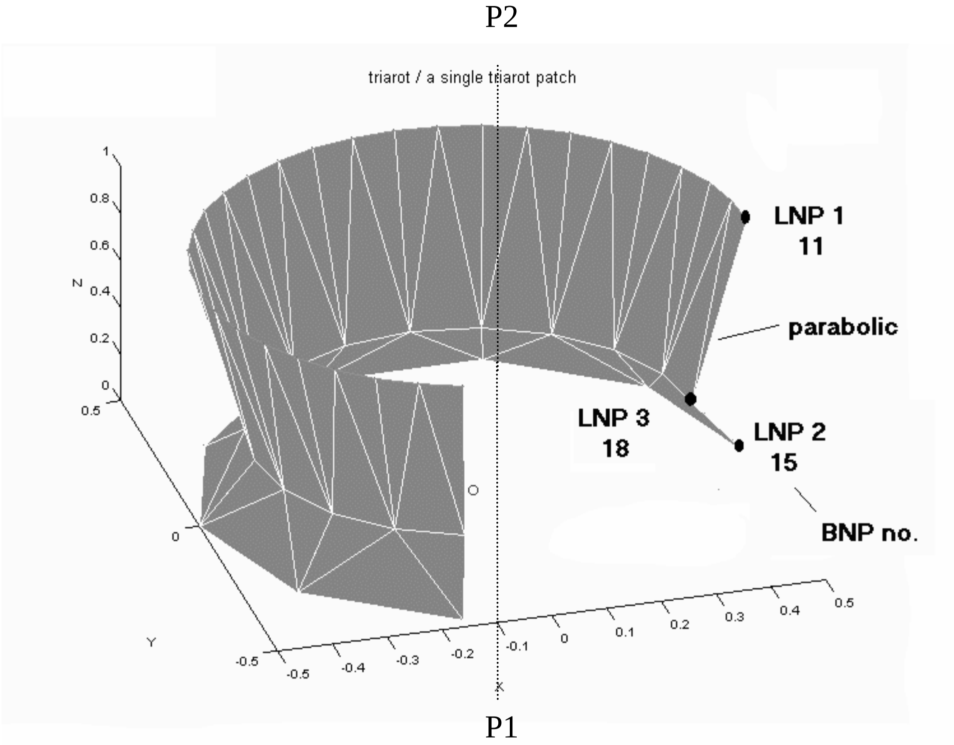
1) a plane patch between BNP(11 15 18) = LNP(1 2 3)

   tria6= [11 15 18 0 0 0]

2) a patch between BNP(11 15 18) = LNP(1 2 3) with curved edges, as in fig. 5.8
   .
   tria6= [11 15 18 22 23 17]

- *triarot* axis-symmetry area

triarot axis-symmetry areas are defined by a *(m_a ,3)* matrix of BNP numbers called *triarot,* where m_a specifies the overall number of axis-symmetry areas. *triarot* may also be defined as a *(m_a ,3)* cell array, if BNP labels are used. Each area is defined by 1...3 local nodal points (LNP) which are selected by the user as a subset of the BNP's. These nodal points specify the contour of the area.
Note: *triarot* axis-symmetry areas are used to generate 3 node <u>shell</u> elements. They are especially designed to allow local mesh refinements i.e. to link coarse mesh areas to fine mesh areas.



Note: The partition geometry /structural element is set by *eta_triarot_umfang*, refer to chapt. 5.1.c

Fig. 5.9: a single *triarot* axis-symmetry area specified by LNP1 .. LNP3

A *triarot* axis-symmetry area can represent a curved surface in 3-D space. A straight contour is defined by specifying LNP (1 2) only and setting LNP (3) to zero. A parabolic shaped contour can be defined using LNP (1 2 3).
Specific parameters for the element generation, e.g. rotation axis, are set for each element of *triarot* by specifying the parameter matrix *triarotpar*:

triarotpar(:,1)	X coordinate of P1 to define rotation axis
triarotpar(:,2)	Y " of P1 " " " "
triarotpar(:,3)	Z " of P1 " " " "
triarotpar(:,4)	X coordinate of P2 to define rotation axis
triarotpar(:,5)	Y " of P2 " " " "
triarotpar(:,6)	Z " of P2 " " " "

triarotpar(:,7) assignment direction of element node coincidence
= 0 negative to triarot(:,1) -> triarot(:,2) direction
= 1 in triarot(:,1) -> triarot(:,2) direction

Example 5.4:

Generate a *triarot* axis-symmetry area of a given structure:

1) a *triarot* axis-symmetry area with a straight contour between BNP(11 15) = LNP(1 2), as in fig. 5.19, rotation axis: global Z axis

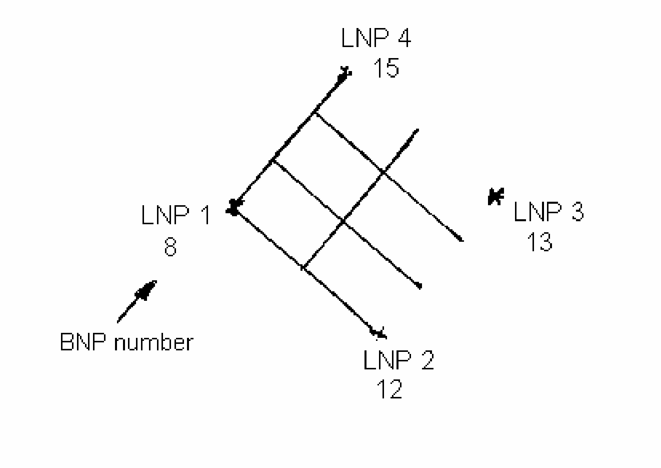
   triarot = [11 15 0]
   triarotpar = [0 0 0 0 0 1 1]

2) a *triarot* axis-symmetry area with a parabolic contour between BNP(11 15 18) = LNP(1 2 3) , as in fig. 5.9, rotation axis: global Z axis

   triarot = [11 15 18]
   triarotpar = [0 0 0 0 0 1 1]

- grillage

Grillages are defined by a *(m$_g$,4)* matrix of BNP numbers called *roste*, where *m$_g$* specifies the overall number of grillages. *roste* may also be defined as a *(m$_g$,4)* cell array, if BNP labels are used. Each grillage is defined by 1...4 local nodal points (LNP) which are selected by the user as a subset of the BNP's.
Note: *roste* grillages are used to generate beam elements.



Note: The partition geometry /structural element is set by *xsir* and *etar*, refer to chapt. 5.1.c

Fig. 5.10: A single *roste* grillage specified by LNP1 ... LNP4

A grillage can represent a skew surface in 3-D space. Grillages are generally not curved. The edges along LNP (2 3) and LNP (3 4) will not be included in the beam allocation process to allow for appending grillages without duplicating beam elements at these edges. However, if at these edges beam elements are required the user must use the *line* geometry element.

Example 5.5:

Generate a *roste* grillage of a given structure

1) between BNP(8 12 13 15), acc. fig 5.10

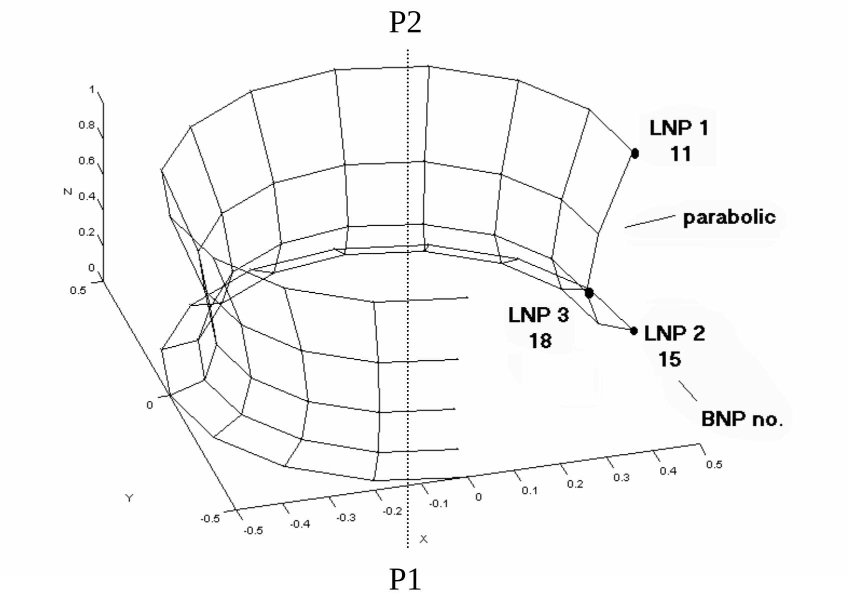
   roste = [8 12 13 15]

Note: It is recommended to define non-linear 2DOF truss elements using *lines* !

- *rostrot* axis-symmetry grillage

rostrot axis-symmetry grillages are defined by a *(m_a ,3)* matrix of BNP numbers called *rostrot,* where m_a specifies the overall number of axis-symmetry areas. *rostrot* may also be defined as a *(m_a ,3)* cell array, if BNP labels are used. Each area is defined by 1...3 local nodal points (LNP) which are selected by the user as a subset of the BNP's. These nodal points specify the contour of the area.

Note: *rostrot* axis-symmetry grillages are used to generate <u>beam</u> elements.



Note: The partition geometry /structural element is set by *xsimantelrost and etaumfangrost*, refer to chapt. 5.1.c

Fig. 5.11: a single *rostrot* axis-symmetry area specified by LNP1 .. LNP3

An axis-symmetry grillage can be defined on a curved surface in 3-D space. A plane contour is defined by specifying LNP (1 2) only and setting LNP (3) to zero. A parabolic shaped contour can be defined using LNP (1 2 3).

Specific parameters for the element generation, e.g. rotation axis, are set for each element of *rostrot* by specifying the parameter matrix *rostrotpar*:

rostrotpar(:,1) X coordinate of P1 to define rotation axis
rostrotpar(:,2) Y " of P1 " " " "
rostrotpar(:,3) Z " of P1 " " " "
rostrotpar(:,4) X coordinate of P2 to define rotation axis
rostrotpar(:,5) Y " of P2 " " " "
rostrotpar(:,6) Z " of P2 " " " "

rostrotpar(:,7) switch to generate ribs
rostrotpar(:,8) switch to generate stringers

rostrotpar(:9-10) = 0 (reserved)

Example 5.6:

Generate a *rostrot* axis-symmetry grillage of a given structure:

1) an axis-symmetry grillage with a plane contour between
BNP(11 15) = LNP(1 2), rotation axis: global Z axis, ribs and stringers

rostrot = [11 15 0]
rostrotpar = [0 0 0 0 0 1 1 1 0 0]

2) an axis-symmetry grillage with a parabolic contour between
BNP(11 15 18) = LNP(1 2 3) , as in fig. 5.11,
rotation axis: global Z axis, ribs and stringers

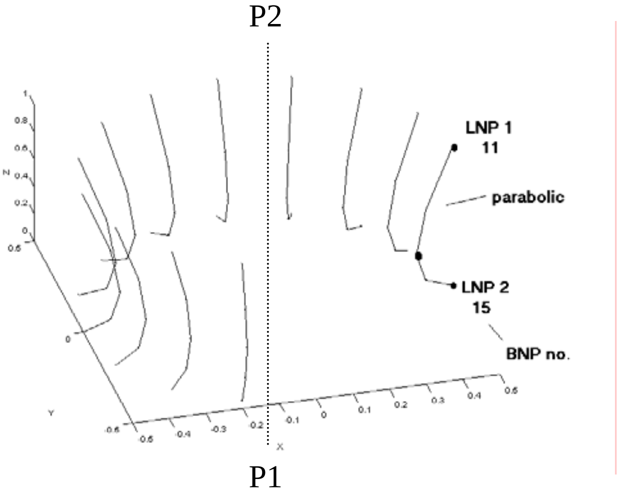
rostrot = [11 15 18]
rostrotpar = [0 0 0 0 0 1 1 1 0 0]

Note: Do not define non-linear 2DOF truss elements using parabolic contour!

3) an axis-symmetry grillage with a parabolic contour between
BNP(11 15 18) = LNP(1 2 3) , as in fig. 5.12,
rotation axis: global Z axis, ribs only

rostrot = [11 15 0]
rostrotpar = [0 0 0 0 0 1 1 0 0 0]

Note: Do not define non-linear 2DOF truss elements using parabolic contour!



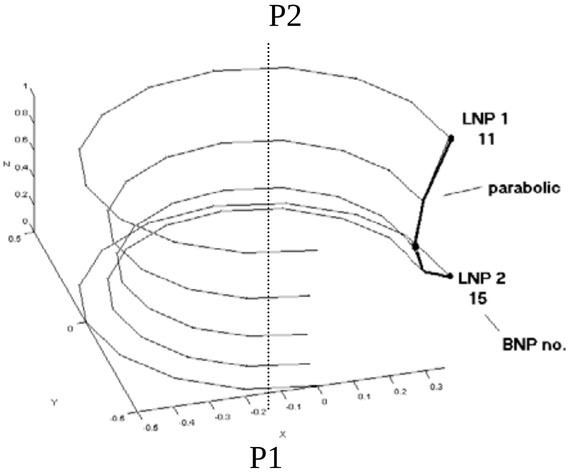
Note: The partition geometry /structural element is set by *xsimantelrost and etaumfangrost*, refer to chapt. 5.1.c

Fig. 5.12: Axis-symmetry grillage, ribs only

4) an axis-symmetry area with a parabolic contour between
BNP(11 15 18) = LNP(1 2 3) , as in fig. 5.13,
rotation axis: global Z axis, stringers only

rostrot = [11 15 0]
rostrotpar = [0 0 0 0 0 1 0 1 0 0]

Note: Do not define non-linear 2DOF truss elements using parabolic contour!



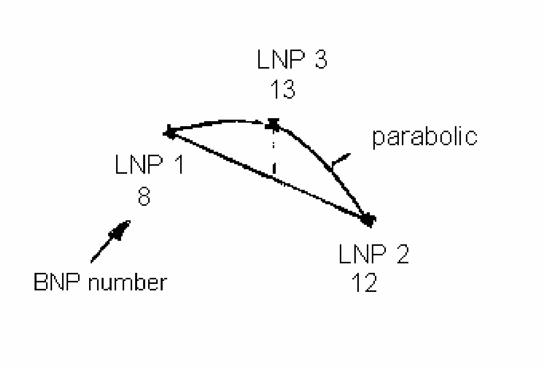
Note: The partition geometry /structural element is set by *xsimantelrost and etaumfangrost*, refer to chapt. 5.1.c

Fig. 5.13: Axis-symmetry grillage, stringers only

Note: It is recommended to define non-linear 2DOF truss elements using *lines* !

- line

Lines are defined by a *$(m_l, 3)$* matrix of BNP numbers called *linien*, where *m_l* specifies the overall number of lines. *linien* may also be defined as a *$(m_l,3)$* cell array, if BNP labels are used. Each line is defined by 1...3 local nodal points (LNP) which are selected by the user as a subset of the BNP's.
Note: *linien* lines are used to generate beam elements.



Note: The partition geometry /structural element is set by *xsil*, refer to chapt. 5.1.c

Fig. 5.14: A single *linien* line specified by LNP1 ... LNP3

A line can represent a parabolic curved connection line in 3-D space. If the LNP 3 is set to zero the connection line between LNP 1 and 2 is a straight line.

Example 5.7:

Generate a *linien* line of a given the structure

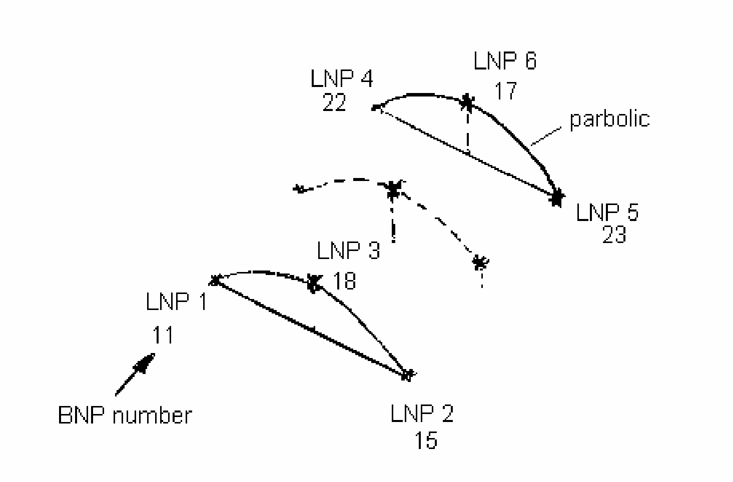
1) a straight line between BNP (8 12)
linien = [8 12 0]

1) a parbolic connection line between BNP (8 12), acc. fig 5.14.
linien = [8 12 13]

Note: Do not define non-linear 2DOF truss elements using parabolic connections!

- rib area

Rib areas are defined by a *(m$_r$,6)* matrix of BNP numbers called *rippen*, where *m$_r$* specifies the overall number of rib areas. *rippen* may also be defined as a *(m$_r$,3)* cell array, if BNP labels are used. Each rib area is defined by 1...6 local nodal points (LNP) which are selected by the user as a subset of the BNP's.
Note: *rippen* rib areas are used to generate beam elements.



Note: The partition geometry /structural element is set by *xsirip and etarip*, refer to chapt. 5.1.c

Fig. 5.15: A single *rippen* rib area specified by LNP1 ... LNP6

A rib area is defined by two straight or curved lines representing the first rib between LNP(1...3) and the last rib between LNP(4...6). If LNP(3, 6) are set to zero the connection lines between LNP(1 , 2) and LNP(4 , 5) are straight lines.

Example 5.8:

Generate a *rippen* rib area of a given structure:

1) a plane rib area between BNP (11 15 22 23)

   rippen= [11 15 0 22 23 0]

2) a rib area between BNP (11 15 22 23) with parabolic curved edges, acc. fig. 5.15

   rippen= [11 15 18 22 23 17]

Note: Do not define non-linear 2DOF truss elements using parabolic curved edges!
Note: It is recommended to define non-linear 2DOF truss elements using *lines*!

5.1c) Specification of fraction ratios for geometry elements

A local ξ,η coordinate system is assigned to each geometry element to conveniently support mesh refinement. The overall dimensions along each edge of the geometry elements are scaled to unity and can be partitioned in arbitrarily fractions, e.g.

ξ = [0 .25 .5 .75 1 NaN],
η = [0 .33 1 NaN] .

NaN (Not a Number) is generally used as the terminator of the fraction ratio specification.

To each type of geometry element up to two corresponding partition vectors are assigned:

fla9	-> *xsif, etaf*
flarot	-> *xsimantel, etaumfang*
tria6	-> *xsit*
triarot	-> *eta_triarot_quot, eta_triarot_umfang*
grillages	-> *xsir, etar*
rostrot	-> *xsimantelrost, etaumfangrost*
rostrot	-> *xsimantelrost, etaumfangrost*
linien	-> *xsil*
rib	-> *xsirip, etarip*

Each element of a geometry element must have its own fraction ratio specification, e.g. for two fla9 patches

```
fla9= [ 12  11  23  24  0  0  0  0  0
         9  16  23  17  0  0  0  0  0]
```

each fraction ratio vector must comprise two ranges

```
xsif = [ 0  .25  .5  .75  1  NaN   0  1  NaN]
etaf = [ 0  .25            1  NaN   0  1  NaN]
```

Each range is terminated by NaN.
If **no** refinement is required ξ,η are set to ξ = [0 1 NaN], η = [0 1 NaN]

Note:
The fraction ratios ξ , η for each type of geometry element are generally specified in a single row vector where the fraction ratios for each element are terminated by NaN. This is different to the specification of the geometry elements where matrices are used (e.g. rippen, linien). Each row of those matrices specifies an element of the geometry elements.

The partitions of the ξ,η axis are independent in general. Each cross point of fraction line / fraction line or fraction line / edge defines the location of a nodal point which will be further used for element allocation.

Generally, the user will use fraction ratios which vary within the range of 0 and 1. But the specification of fraction ratios is not restricted to this range. The user may even specify e.g. xsif= [–1 ... 2] or even xsif= [2 ... 4]. In these cases the left and right borders are beyond the definition range of the geometry element which is given by the spatial distribution of the respective BNP's. This definition is possible because the BNP's are used to define the geometry element only and are not used to define the structural FEM elements like beams and shells.
The partitioning of the geometry element determines the extend of mesh refinement. The overall mesh is derived from the geometry elements and their respective refinement. MATFEM automatically manages node and element numbering. The user then only has to assign geometry and material properties to each element (specified in *.m07, *.m08).

Note: It is recommended to specify a fraction ratio of [0:1 NaN] for non-linear 2DOF truss elements !

Note:

MATFEM allocates shell elements from geometry elements in the following order

fla9	(patch)	3 and/or 4 node shell elements
flarot	(axis-symmetry area)	3 and/or 4 node shell elements
tria6	(patch)	3 node shell elements
triarot	(axis-symmetry area)	3 node shell elements

Therefore shell elements specified using *flarot* always have a higher element number than those specified by *fla9*.

MATFEM allocates beam elements from geometry elements in the following order

roste	(grillage)
rostrot	(axis-symmetry grillage)
linien	(line)
rippen	(rib area)

Therefore beam elements specified using *linien* always have a lower element number than those specified by *rippen*. But they have a higher element number than those generated by *rostrot.*

MATFEM allocates shell elements first followed by the allocation of beam elements. Therefore shell elements always have a lower element number than beam elements.

For detailed examples about meshing please refer to Appendix D and to the MATFEM demonstration examples given in the DEMO directory.

- fla9 patch

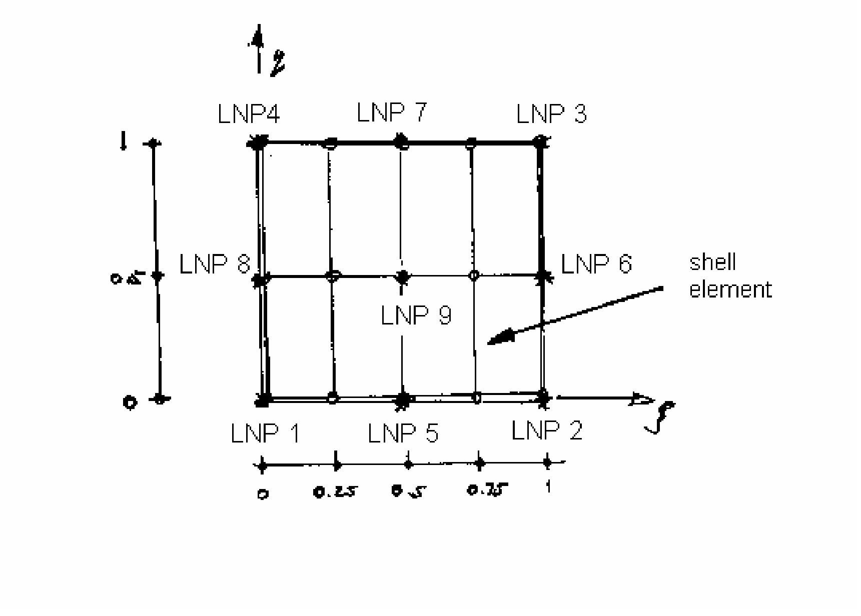


Fig. 5.16: fraction ratios ξ and η of a *fla9* patch

The local ξ axis of each patch (row) of *fla9* is specified by LNP 1 and LNP 2 i.e. *fla9*(:,1) and *fla9*(:,2) . The local η axis of each patch area of *fla9* is specified by LNP 1 and LNP 4 i.e. *fla9* (:,1) and *fla9*(:,4) .
The fraction ratios of the ξ and η axis are specified in the row vectors *xsif* and *etaf*. For each patch of matrix *fla9* two fraction ratios e.g. 0 and 1 have to be specified at least. If no partitioning is required, xsif and *etaf* only consist of mp * 3 elements
[0 1 NaN 0 1 NaN ....]. In this case each patch represents a single shell element.
Generally, the partitioning of *xsif* can be different to *etaf*

Note: *fla9* patches are used to generate 3 and/or 4 node shell elements

Example 5.9:

Generate shell elements from a given *fla9* patch acc. fig 5.16:

ξ divided into 4 elements η divided into 2 elements

xsif = [0 .25 .50 .75 1 NaN]

etaf = [0 .50 1 NaN]

-flarot axis-symmetry area

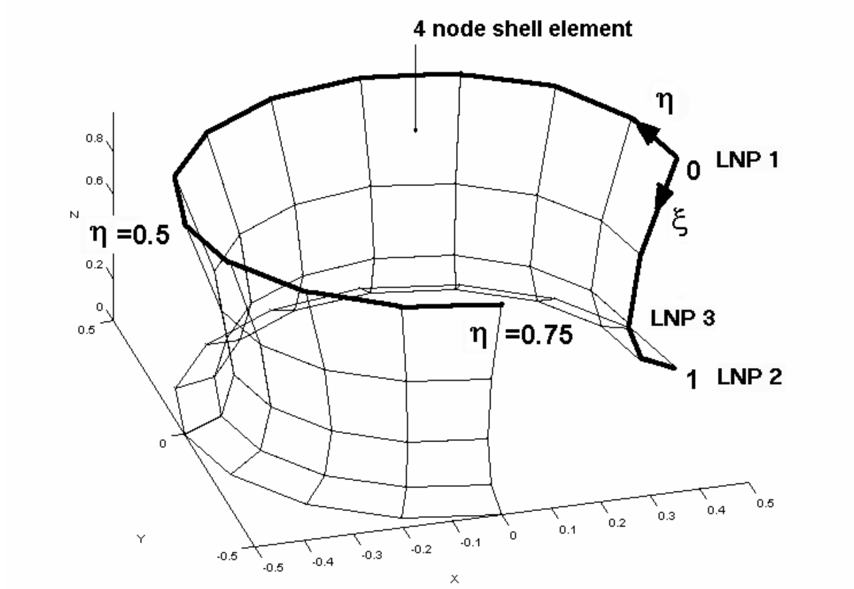


Fig. 5.17: fraction ratios ξ and η of a *flarot* axis-symmetry area

The local ξ axis of each axis-symmetry area (row) of flarot is specified by LNP 1, LNP 2 and LNP 3 i.e. *flarot*(:,1 : 3). The local η axis of each axis symmetry area of *flarot* is specified counterclockwise in circumferential direction.
The fraction ratios of the ξ and η axis are specified in the row vectors *xsimantel* (Teilung in Mantelrichtung) and *etaumfang (*Teilung in Umfangsrichtung). For each axis-symmetry area of matrix *flarot* two fraction ratios e.g. 0 and 1 have to be specified at least. If no partitioning is required, *xsimantel* and *etaumfang* only consist of m_p * 3 elements [0 1 NaN 0 1 NaN ....]. In this case each axis-symmetry area represents a single shell element, which for axis-symmetry purposes does not make much sense. Therefore *etaumfang* should comprise at least 4 elements to represent a circumferential contour, e.g. etaumfang = [0 .25 .5 .75 1]. However, from the point of structural mechanics, *eta_triarot_umfang* should at least comprise 8 elements to represent a shell structure.
Generally, the partitioning of *xsimantel* can be different to *etaumfang.*

Note: *flarot* axis-symmetry areas are used to generate 3 and/or 4 node shell elements

Example 5.10:

Generate shell elements from a given *flarot* axis-symmetry area acc. fig 5.17:

ξ divided into 4 elements (Mantelrichtung)
η divided into 12 elements (Umfangsrichtung)

xsimantel = [0 .25 .50 .75 1 NaN]

etaumfang = [0 : 1/ 16 : .75 NaN]

- tria6 patch

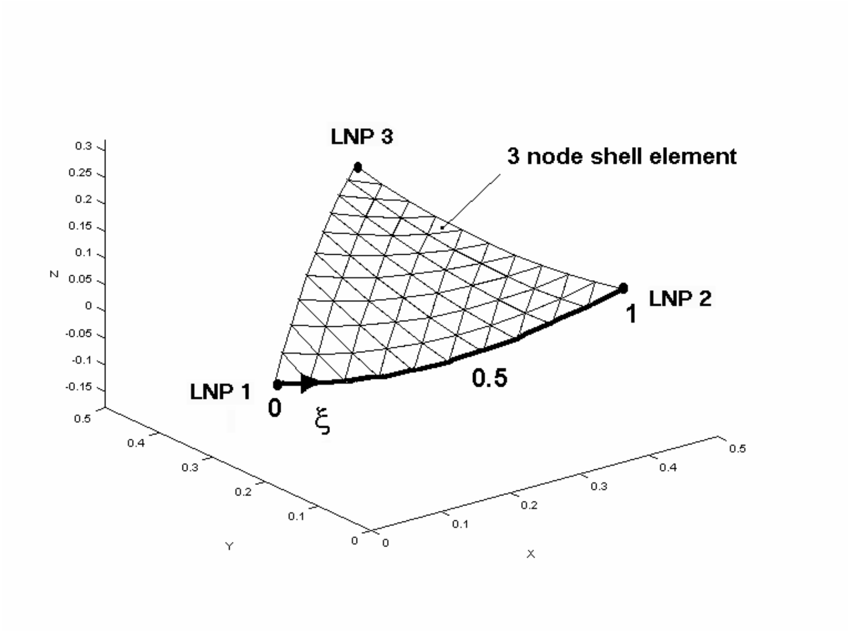


Fig. 5.18: fraction ratios ξ of a *tria6* patch

The local ξ axis of each patch (row) of *tria6* is specified by LNP 1 and LNP 2 i.e. *tria6*(:,1) and *tria6*(:,2).
The fraction ratios of the ξ axis are specified in the row vectors *xsit*. The fraction ratios of the η axis are automatically set equal to the ξ axis. For each patch of matrix *tria6* two fraction ratios e.g. 0 and 1 have to be specified at least. If no partitioning is required, xsit only consist of m_p * 3 elements [0 1 NaN 0 1 NaN ....]. In this case each patch represents a single shell element.

Note: *tria6* patches are used to generate 3 node shell elements

Example 5.11:

Generate shell elements from a given *tria6* patch acc. fig 5.18:

ξ divided into 10 elements

xsit = [0 : 0.1 : 1 NaN]

-triarot axis-symmetry area

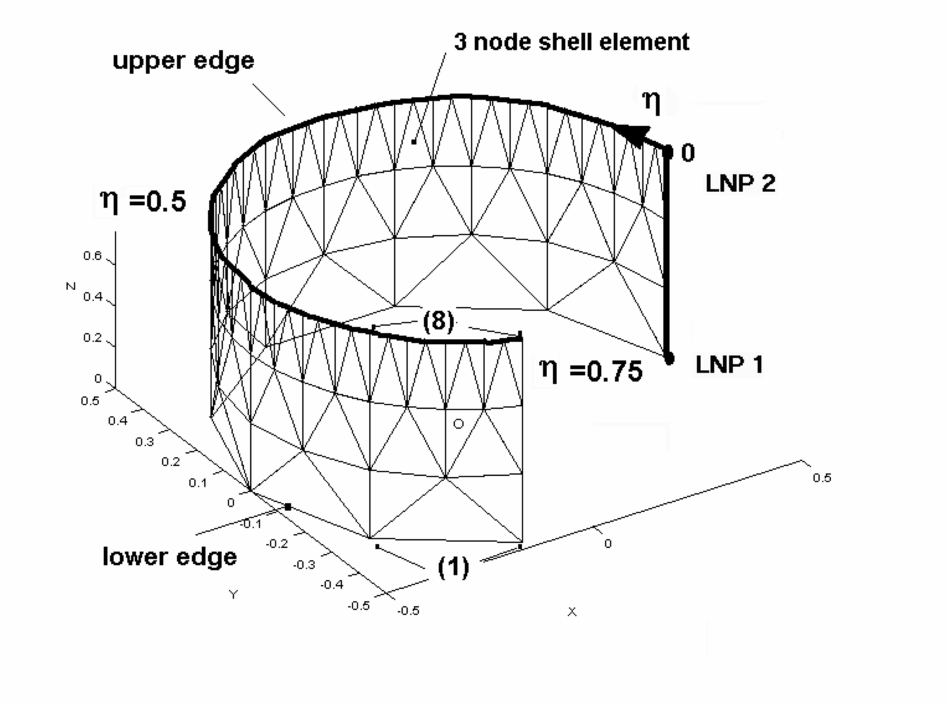


Fig. 5.19: fraction ratios η of a *triarot* axis-symmetry area

The local η axis of each axis symmetry area of *triarot* is specified counterclockwise in circumferential direction.

The fraction ratios of the η axis are specified in the row vector *eta_triarot_umfang (*Teilung in Umfangsrichtung). The fraction ratios of the xsi axis are even spaced and derived automatically from the partition ratio of eta w.r. to the upper and lower edge of the *triarot* element. The partition ratios must be of power 2 and are specified by the vector eta_triarot_quot.

Example acc. fig. 5.19: eta_triarot_quot = 8 : one fraction at the lower edge yields eight fractions at the upper edge.

For each axis-symmetry area of matrix *triarot* two fraction ratios e.g. 0 and 1 have to be specified at least. If no partitioning is required, *eta_triarot_umfang* only consist of mp * 3 elements [0 1 NaN 0 1 NaN ....]. In this case each axis-symmetry area represents a single shell element, which for axis-symmetry purposes does not make much sense. Therefore *eta_triarot_umfang* should comprise at least 4 elements to represent a closed circumferential contour, e.g.

eta_triarot_umfang = [0 .25 .5 .75 1 NaN].

However, from the point of structural mechanics, *eta_triarot_umfang* should at least comprise 8 elements to represent a shell structure.

Note: *triarot* patches are used to generate 3 node shell elements

Example 5.12:

Generate shell elements from a given *triarot* axis-symmetry area acc. fig 5.19:

η divided into 7 elements (Umfangsrichtung) at the lower edge and into 80 elements at the upper edge. There are (7+1) nodes at the lower edge and (8*7 + 1) nodes at the upper edge

```
eta_triarot_umfang = [ 0 : .1:  .7  NaN];

eta_triarot_quot   =  [ 8 ];
```

The respective fraction ratios of the ξ-axis for this example are automatically set to ξ = [0: 1/3: 1]

- grillage

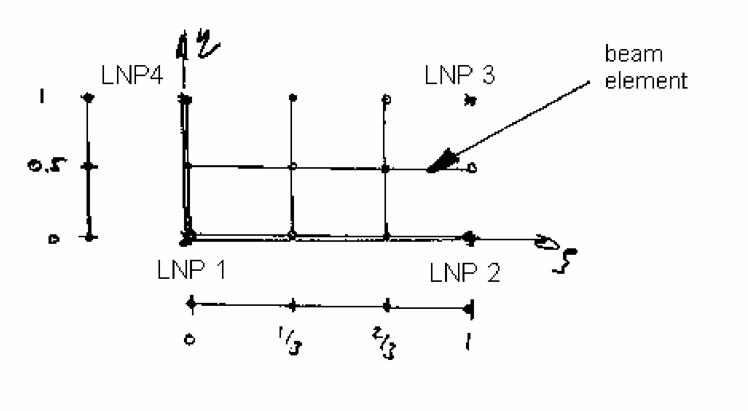


Fig. 5.20: fraction ratios ξ and η of a *roste* grillage

The local ξ axis of each grillage (row) of *roste* is specified by LNP 1 and LNP 2 i.e. *roste*(:,1) and *roste*(:,2) . The local η axis of each grillage of *roste* is specified by LNP 1 and LNP 4 i.e. *roste* (:,1) and *roste*(:,4) .
The fraction ratios of the ξ and η axis are specified in the row vectors *xsir* and *etar*. For each grillage of matrix *roste* the fraction ratios 0 and 1 have to be specified at least. If no partitioning is required, *xsir* and *etar* only consist of m_g * 3 elements
[0 1 NaN 0 1 NaN ....].
Generally, the partitioning of *xsir* can be different to *etar*

Note: *roste* grillages are used to generate beam elements

Example 5.13:

Generate beam elements from a given *roste* grillage acc. fig 5.20:

ξ divided into 3 elements η divided into 2 elements

xsir = [0 1/3 2/3 1 NaN]

etar = [0 .50 1 NaN]

-rostrot axis-symmetry grillage

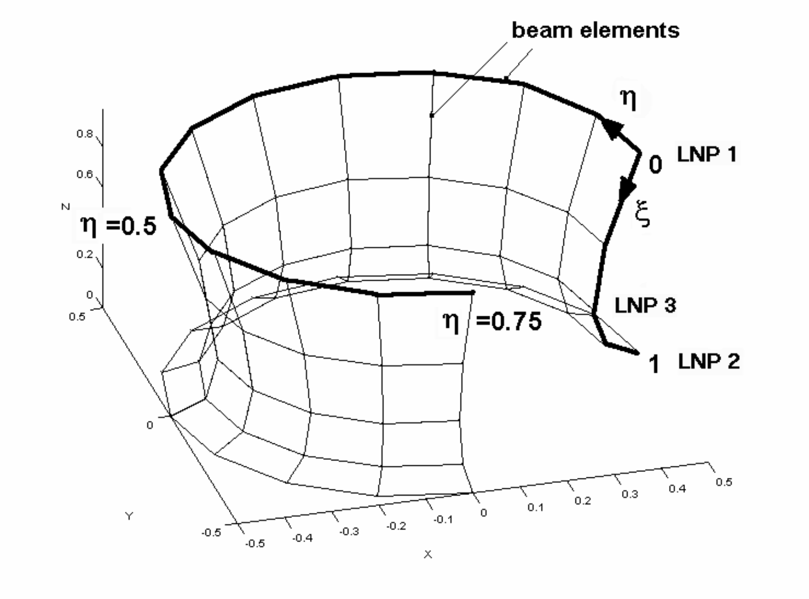


Fig. 5.21: fraction ratios ξ and η of a *rostrot* axis-symmetry area

The local ξ axis of each axis-symmetry grillage (row) of *rostrot* is specified by LNP 1, LNP 2 and LNP 3 i.e. *rostrot*(:,1 : 3). The local η axis of each axis symmetry area of *rostrot* is specified counterclockwise in circumferential direction.
The fraction ratios of the ξ and η axis are specified in the row vectors *xsimantelrost* (Teilung in Mantelrichtung) and *etaumfangrost (*Teilung in Umfangsrichtung).
For each axis-symmetry grillage of matrix *rostrot* two fraction ratios e.g. 0 and 1 have to be specified at least. If no partitioning is required, *xsimantelrost* and *etaumfangrost* only consist of m_p * 3 elements [0 1 NaN 0 1 NaN ....]. In this case each axis-symmetry grillage represents a single beam element, which for axis-symmetry purposes does not make much sense. Therefore *etaumfangrost* should comprise at least 4 elements to represent a closed circumferential contour, e.g. *etaumfangrost* = [0 .25 .5 .75 1 NaN].
Generally, the partitioning of *xsimantelrost* can be different to *etaumfangrost.*

Note: *rostrot* axis-symmetry grillages are used to generate beam elements

Example 5.14:

Generate beam elements from a given *rostrot* axis-symmetry grillage acc. fig 5.21:

ξ divided into 4 elements (Mantelrichtung)
η divided into 12 elements (Umfangsrichtung)

xsimantelrost = [0 .25 .5 .75 1 NaN]

etaumfangrost = [0 : 1/ 16 : .75 NaN]

- line

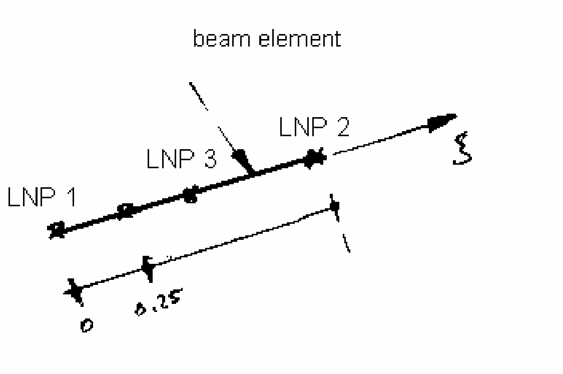


Fig. 5.22: fraction ratio ξ of a *linien* line

The local ξ axis of each line (row) of *linien* is specified by LNP 1 and LNP 2 i.e. *linien*(:,1) and *linien*(:,2) . The fraction ratio of the ξ axis of each line is specified in the row vector *xsil*. For each line of matrix *linien* the fraction ratios 0 and 1 have to be specified at least. If no partitioning is required, *xsil* only consists of m_l * 3 elements [0 1 NaN 0 1 NaN ....].

Note: *linien* lines are used to generate beam elements

Example 5.15:

Generate beam elements from a given *linien* line acc. fig 5.22:

ξ divided into 3 elements

xsil = [0 .25 .5 1 NaN]

Note: For non-linear 2DOF truss elements it is recommended to define *xsil*= [0 1 NaN]!

- rib area

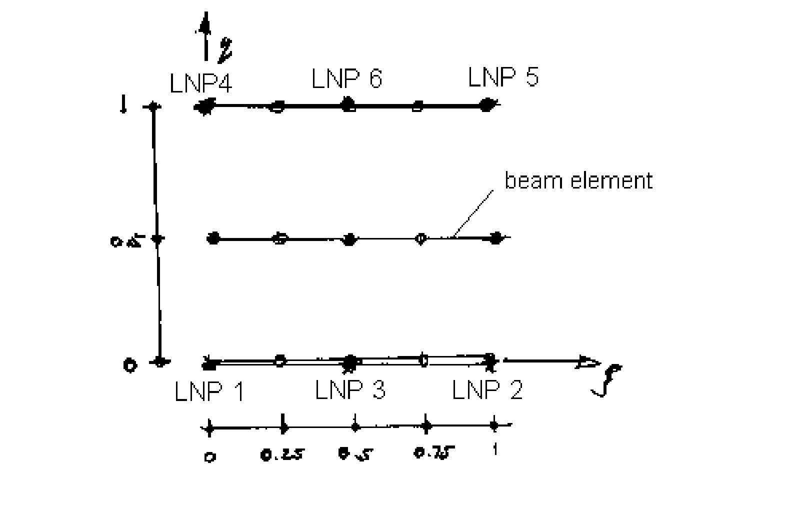


Fig. 5.23: fraction ratios ξ and η of a *rippen* rib area

The local ξ axis of each rib area (row) of *rippen* is specified by LNP 1 and LNP 2 i.e. *rippen*(:,1) and *rippen*(:,2) . The local η axis of each rib area of *rippen* is specified by LNP 1 and LNP 4 i.e. *rippen*(:,1) and *rippen*(:,4) .
The fraction ratios of the ξ and η axis are specified in the row vectors *xsirip* and *etarip*. For each rib area of matrix *rippen* the fraction ratios 0 and 1 have to be specified at least. If no partitioning is required, *xsirip* and *etarip* only consist of m_r * 3 elements [0 1 NaN 0 1 NaN ....].
Generally, the partitioning of *xsirip* can be different to *etarip*

Note: *rippen* rib areas are used to generate beam elements

Example 5.16:

Generate beam elements from a given *rippen* rib area acc. fig 5.23:

ξ divided into 4 elements
η divided into 2 elements

xsirip = [0 .25 .50 .75 1 NaN]

etarip = [0 .50 1 NaN]

To illustrate the mesh generating process consider the structure of the demonstration example demo1:

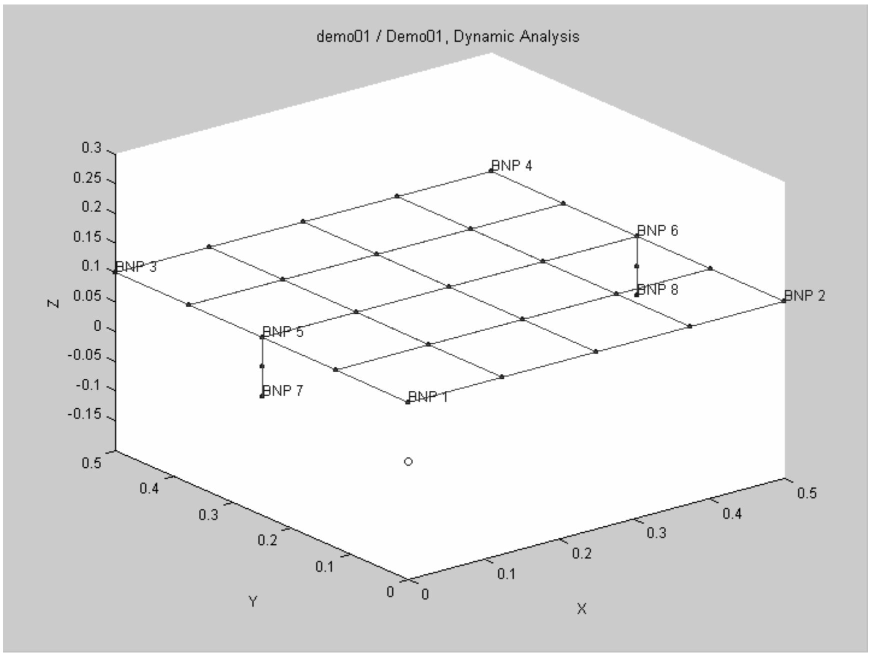


Fig. 5.24a: Illustration of mesh generating process, demo01

The mesh of this structure is generated as follows:

step 1: Nodes 1, 2, 3, 4, 5, 6, 7, 8 are specified as BNP's .

```
a = .5;    % Parameter for overall structure width and length dimensions
h = .1;    % Parameter for overall structure height dimension

 % ------------- X --- Y ------- Z -------------------global coordinates
berkoord = [  0      0        h     %  BNP 1
              a      0        h     %  BNP 2
              0      a        h     %  BNP 3
              a      a        h     %  BNP 4
              0      .5*a     h     %  BNP 5
              a      .5*a     h     %  BNP 6
              0      .5*a     0     %  BNP 7
              a      .5*a     0     %  BNP 8   ]
```

step 2: A *fla9* patch is defined using BNP(1, 2, 3, 4) .

```
fla9      = [1 2 4 3  0 0 0 0  0]
fla9par   = [ 4  0   0 0 0 0]
```

The partition vectors *xsif* and *etaf of* this patch are set to

```
xsif= [ 0 .25 .5 .75 1   NaN ]
etaf= [ 0 .25 .5 .75 1   NaN ]
```

This yields a 4 by 4 mesh of 4 node shell elements. The total number of shell elements is *nshel* = 16 .

step3: Two straight lines are defined using BNP(5, 7) and BNP(6, 8)

```
linien = [ 7 5 0 ;
           8 6 0 ]
```

The partition vector *xsil* of the two lines is set to

```
xsil= [ 0  .5  1   NaN      0  .5   1  NaN]
```

This yields 2 beam elements for each line. The total number of beam elements is *nbalk*= 4 .

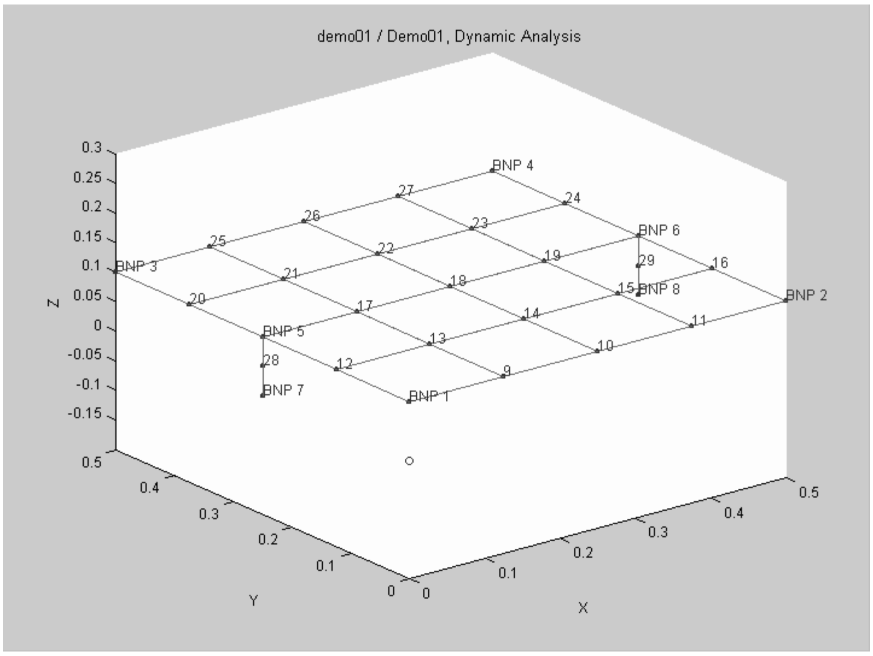


Fig. 5.24b: Nodes of the generated mesh
The overall structure is made of np = 29 nodal points:
Nodes 1 ... 8 correspond to BNP 1 ...8.
The BNP label is automatically set to 'BNP 1' ... 'BNP 8' .

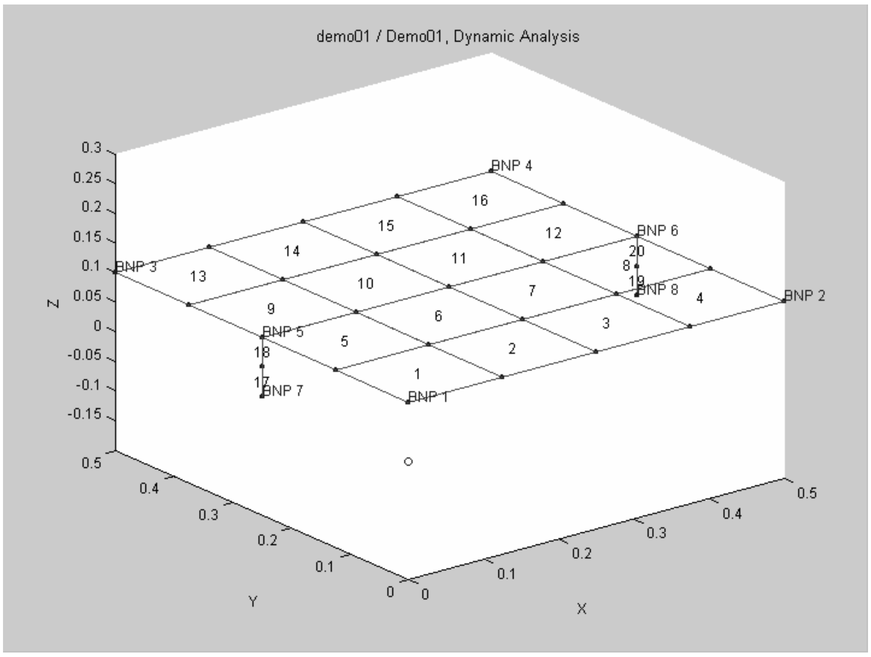


Fig. 5.24c: Elements of the generated mesh:
The overall structure is made of 20 elements:
16 shell elements(1 ...16), 4 beam elements(17 ...20)

Due to the mesh generating process used in MATFEM it is recommended to follow a three step procedure

1. The user should first generate the BNP's, the geometry elements and their respective fraction ratios. The program flow parameter *iplt* specified in *.m03 should be set to *iplt = 4 (generate FEM mesh , plot)*

2. The user should then start the analysis process which will terminate after the mesh is generated. No further calculations will be performed. The generated mesh can then be checked using the MATFEM postprocessor. Thereby the user can find out about node and element numbering, which is automatically performed by MATFEM.

3. The user can then supply all further needed input and start the analysis specifying the program flow parameter *iplt*
iplt = 2 (generate FEM mesh , plot, analysis)

During all steps the user should use the listing file **.aus* to check all input settings the program flow and the analysis results.

The complete set of MATFEM input files for demo01.m01, demo01.m04, ... is listed in the appendix D and is given in the \demo directory.

6. Element Types

MATFEM supports the following element types:

beam element

- general 2 node; 12 DOF, beam element (Timoshenko, Bernoulli)
- 2 node, 12 DOF spring element with user defined stiffness and mass properties
- 2 node, 12 DOF spring element (realized by decoupled beam element stiffness)
- non-linear 2DOF truss element (for non-linear dynamic response calculation only)

shell element

- 3 node, 18 DOF shell element
- 4 node, 24 DOF shell element

General:

All analysis results viewed in the postprocessor, listed in the *.aus file or stored in the MDB is given node by node or element by element respectively w.r. to the global or local coordinate system:
Displacements, mode shapes and nodal forces are given w.r. to the **global** coordinate system. Stresses are given w.r. to the **local** element coordinate system. The **local** element coordinate system of each element can be viewed within the MATFEM postprocessor.

The required input for beam and shell element properties is managed as follows:

The geometrical and material properties (e.g. YOUNG's modulus, mass density) of either beam or shell elements are specified in parameter vectors in the input files **.m06*, **.m07* and **.m08*. They serve as a kind of general data basis. There must be at least one parameter specified in each parameter vector. But parameter vectors may also include additional parameters which are not referenced in the actual application.

Example 6.1:

```
% --- YOUNG'S Modulus  Ek(1,:) ---------------------

            %    steel    aluminum
       Ek = [  2.10e+11    .7e+11   ];
```

Fig. 6.1: Parameter vector *Ek* to specify the YOUNG'S Modulus for beam elements in input file **.m06*

The user then has to assign the parameters of the parameter vectors to each shell or beam element. For this MATFEM uses the matrices *ec_beam* (file *.m07) and *ec_shell* (file *.m08) to specify element cards for beam and shell elements respectively.

Example 6.2:

```
% --- matrix of beam element cards  EC_beam( 21,:) -----------------------------
%
%     EC_beam( 1, :) =  element type
%
%                       = 100    linear    2 node, 12 DOF general spring element,
nodes may coincide
%
%                       = 101    linear    2 node, 12 DOF beam  element (BERNOULLI
beam theory)
%                       = 102    linear    2 node, 12 DOF beam  element (TIMOSHENKO
beam theory)
%
%                       = 105    linear    2 node, 12 DOF spring element, nodes
must not coincide
%
%                       = 111    nonlinear 2 node,  2 DOF truss element,
nonlinearity of 'power' type
%                       = 112    nonlinear 2 node,  2 DOF truss element,
nonlinearity of 'Signum/Coulomb' type
%                       = 113    nonlinear 2 node,  2 DOF truss element, stiffness
nonlinearity of 'arctan' type
%                       = 114    nonlinear 2 node,  2 DOF truss element, stiffness
nonlinearity of 'arcsinh' type
%                       = 115    nonlinear 2 node,  2 DOF truss element,
nonlinearity of 'clearance' type
%                       = 116    nonlinear 2 node,  2 DOF truss element, Gaul
element (2 Parameters)
%
%
%     Type 100    linear    2 node, 12 DOF spring element, coinciding nodes
%
%             EC_beam(  2, :) = auxiliary node            ( beam100_auxnode index
no.)
%             EC_beam(  3, :) = CS for element stiffness matrix input
%                             = 0 global Cs
%                             = 1 local element Cs
%             EC_beam(  4, :) = element stiffness matrix ( beam100_k       index
no.)
%             EC_beam(  5, :) = CS for element mass matrix input
%                             = 0 global Cs
%                             = 1 local element Cs
%             EC_beam(  6, :) = element mass matrix      ( beam100_m       index
no.)
%             EC_beam(  7, :) = CS for equivalent element force vector input
%                             = 0 global Cs
%                             = 1 local element Cs
%             EC_beam(  8, :) = equivalent element force vector
%             EC_beam(  9, :) = angle dgamma of principal axis w.r. to beam
reference system (radian)
%             EC_beam( 10, :) = reserved
%             EC_beam( 11. :) = reserved
%
%                               eccentric nodal point connections
%                               node A:
%             EC_beam( 12, :) = distance y-direction: eccentric nodal point/ center
of gravity
%             EC_beam( 13, :) = distance z-direction: eccentric nodal point/ center
of gravity
%             EC_beam( 14, :) = distance y-direction: eccentric nodal point/ shear
center
%             EC_beam( 15, :) = distance z-direction: eccentric nodal point/ shear
center
```

```
%                               node B:
%             EC_beam( 16, :) = distance y-direction: eccentric nodal point/ center
of gravity
%             EC_beam( 17, :) = distance z-direction: eccentric nodal point/ center
of gravity
%             EC_beam( 18, :) = distance y-direction: eccentric nodal point/ shear
center
%             EC_beam( 19, :) = distance z-direction: eccentric nodal point/ shear
center
%
%                               hinges at local nodes A and/or B  (A=1 or B=2)
%             EC_beam( 20, :) = hinges w.r. to local y axis
%             EC_beam( 21, :) = hinges w.r. to local z axis
%
%
%     Type 101, 102, 105    linear 2 node, 12 DOF beam element
%
%             EC_beam(  2, :) = material law (Ek index no.)
%             EC_beam(  3, :) = cross section (Ak index no.)
%             EC_beam(  4, :) = shear modulus (Gk index no.)
%             EC_beam(  5, :) = maximum moment of inertia (I1) (Jk  index no.)
%             EC_beam(  6, :) = minimum moment of inertia (I2) (Jkk index no.)
%             EC_beam(  7, :) = centroidal torsional moment of inertia (Tk index
no.)
%             EC_beam(  8, :) = mass density (Roh index no.)
%             EC_beam(  9, :) = angle dgamma of principal axis w.r. to beam
reference system (radian)
%
%                               distance: center of gravity / shear center
%             EC_beam( 10, :) = y-direction
%             EC_beam( 11. :) = z-direction
%
%                               eccentric nodal point connections
%                               node A:
%             EC_beam( 12, :) = distance y-direction: eccentric nodal point/ center
of gravity
%             EC_beam( 13, :) = distance z-direction: eccentric nodal point/ center
of gravity
%             EC_beam( 14, :) = distance y-direction: eccentric nodal point/ shear
center
%             EC_beam( 15, :) = distance z-direction: eccentric nodal point/ shear
center
%                               node B:
%             EC_beam( 16, :) = distance y-direction: eccentric nodal point/ center
of gravity
%             EC_beam( 17, :) = distance z-direction: eccentric nodal point/ center
of gravity
%             EC_beam( 18, :) = distance y-direction: eccentric nodal point/ shear
center
%             EC_beam( 19, :) = distance z-direction: eccentric nodal point/ shear
center
%
%                               hinges at local nodes A and/or B  (A=1 or B=2)
%             EC_beam( 20, :) = hinges w.r. to local y axis
%             EC_beam( 21, :) = hinges w.r. to local z axis
%
%
%     Type 111    nonlinear 2 node, 2 DOF beam element, nonlinearity of 'power'
type
%
%                               nonlinear stiffness parameters w.r. to local
coordinate system
%             EC_beam(  2, :) = type of nonlinear stiffness
%             EC_beam(  3, :) = kx
%             EC_beam(  4, :) = ky
%             EC_beam(  5, :) = kz
%             EC_beam(  6, :) = kxx
%             EC_beam(  7, :) = kyy
%             EC_beam(  8, :) = kzz
```

```
%
%                               nonlinear damping parameters w.r. to local
coordinate system
%             EC_beam(  9, :) = type of nonlinear damping
%             EC_beam( 10, :) = dx
%             EC_beam( 11, :) = dy
%             EC_beam( 12, :) = dz
%             EC_beam( 13, :) = dxx
%             EC_beam( 14, :) = dyy
%             EC_beam( 15, :) = dzz
%
%             EC_beam( 16, :) = angle dgamma of principal axis w.r. to beam
reference system (radian)
%
%             EC_beam( 17 : 21, :) = 0    reserved
%
%
%     Type 112    nonlinear 2 node, 2 DOF beam element, nonlinearity of
'Signum/Coulomb' type
%
%                               nonlinear stiffness parameters w.r. to local
coordinate system
%             EC_beam(  2, :) = 0   (reserved)
%             EC_beam(  3, :) = kx
%             EC_beam(  4, :) = ky
%             EC_beam(  5, :) = kz
%             EC_beam(  6, :) = kxx
%             EC_beam(  7, :) = kyy
%             EC_beam(  8, :) = kzz
%
%                               nonlinear damping parameters w.r. to local
coordinate system
%             EC_beam(  9, :) = 0   (reserved)
%             EC_beam( 10, :) = dx
%             EC_beam( 11, :) = dy
%             EC_beam( 12, :) = dz
%             EC_beam( 13, :) = dxx
%             EC_beam( 14, :) = dyy
%             EC_beam( 15, :) = dzz
%
%             EC_beam( 16, :) = angle dgamma of principal axis w.r. to beam
reference system (radian)
%
%             EC_beam( 17 : 21, :) = 0    reserved
%
%
%     Type 113    nonlinear 2 node, 2 DOF beam element, stiffness nonlinearity of
'arctan' type
%
%                               nonlinear stiffness parameters w.r. to local
coordinate system
%             EC_beam(  2, :) = 0   (reserved)
%             EC_beam(  3, :) = kx
%             EC_beam(  4, :) = ky
%             EC_beam(  5, :) = kz
%             EC_beam(  6, :) = bx
%             EC_beam(  7, :) = kyy
%             EC_beam(  8, :) = kzz
%
%                    horizontal deformation factor for arctan(factor*displacement)
%             EC_beam(  9, :) = 0   (reserved)
%             EC_beam( 10, :) = bx
%             EC_beam( 11, :) = by
%             EC_beam( 12, :) = bz
%             EC_beam( 13, :) = bxx
%             EC_beam( 14, :) = byy
%             EC_beam( 15, :) = bzz
%
```

```
%             EC_beam( 16, :) = angle dgamma of principal axis w.r. to beam
reference system (radian)
%
%             EC_beam( 17 : 21, :) = 0    reserved
%
%
%     Type 114    nonlinear 2 node, 2 DOF beam element, stiffness nonlinearity of
'arcsinh' type
%
%                               nonlinear stiffness parameters w.r. to local
coordinate system
%             EC_beam(  2, :) = 0   (reserved)
%             EC_beam(  3, :) = kx
%             EC_beam(  4, :) = ky
%             EC_beam(  5, :) = kz
%             EC_beam(  6, :) = bx
%             EC_beam(  7, :) = kyy
%             EC_beam(  8, :) = kzz
%
%                    horizontal deformation factor for arcsinh(factor*displacement)
%             EC_beam(  9, :) = 0   (reserved)
%             EC_beam( 10, :) = bx
%             EC_beam( 11, :) = by
%             EC_beam( 12, :) = bz
%             EC_beam( 13, :) = bxx
%             EC_beam( 14, :) = byy
%             EC_beam( 15, :) = bzz
%
%             EC_beam( 16, :) = angle dgamma of principal axis w.r. to beam
reference system (radian)
%
%             EC_beam( 17 : 21, :) = 0    reserved
%
%
%     Type 115    nonlinear 2 node, 2 DOF beam element, stiffness nonlinearity of
'clearance' type
%
%                               nonlinear stiffness parameters w.r. to local
coordinate system
%             EC_beam(  2, :) = 0   (reserved)
%             EC_beam(  3, :) = kxo - gap open
%             EC_beam(  4, :) = kxc - gap closed
%             EC_beam(  5, :) = bx  - half gap length [m]
%             EC_beam(  6, :) = kyo - gap open
%             EC_beam(  7, :) = kyc - gap closed
%             EC_beam(  8, :) = by  - half gap length [m]
%             EC_beam(  9, :) = kzo - gap open
%             EC_beam( 10, :) = kzc - gap closed
%             EC_beam( 11, :) = bz  - half gap length [m]
%
%             EC_beam( 12 : 15, :) = 0    reserved
%
%             EC_beam( 16, :) = angle dgamma of principal axis w.r. to beam
reference system (radian)
%
%             EC_beam( 17 : 21, :) = 0    reserved
%
%
%     Type 116    nonlinear 2 DOF truss element, Gaul element (2-Parameters)
%
%                               nonlinear spring stiffnesses w.r. to local
coordinate system
%             EC_beam(  2, :) = 0   (reserved)
%             EC_beam(  3, :) = kx
%             EC_beam(  4, :) = ky
%             EC_beam(  5, :) = kz
%             EC_beam(  6, :) = kxx
%             EC_beam(  7, :) = kyy
%             EC_beam(  8, :) = kzz
```

```
%
%                               nonlinear friction forces w.r. to local coordinate
system
%             EC_beam(  9, :) = 0   (reserved)
%             EC_beam( 10, :) = hx
%             EC_beam( 11, :) = hy
%             EC_beam( 12, :) = hz
%             EC_beam( 13, :) = hxx
%             EC_beam( 14, :) = hyy
%             EC_beam( 15, :) = hzz
%
%             EC_beam( 16, :) = angle dgamma of principal axis w.r. to beam
reference system (radian)
%
%             EC_beam( 17 : 21, :) = 0    reserved


% ----------     linear 2 node, 12 DOF --    non-linear 2node, 2 DOF

EC_beam =  [      102  102  102  102         111    111    111        %  1
                   1    1    1    1           3      0      2         %  2
                   1    2    3    4           0      0     1e7        %  3
                   1    1    1    1           0      0      0         %  4
                   1    2    3    4          1e11    0      0         %  5
                   1    2    3    4           0      0      0         %  6
                   1    2    3    4           0      0      0         %  7
                   1    2    3    4           0      0      0         %  8
                   0    0    0    0           0      2      3         %  9
                   0    0    0    0           0     10      5         % 10
                   0    0    0    0           0      0      0         % 11
                   0    0    0    0           0      0      0         % 12
                   0    0    0    0           0      0      0         % 13
                   0    0    0    0           0      0      0         % 14
                   0    0    0    0           0      0      0         % 15
                   0    0    0    0           0      0      0         % 16
                   0    0    0    0           0      0      0         % 17
                   0    0    0    0           0      0      0         % 18
                   0    0    0    0           0      0      0         % 19
                   0    0    0    0           0      0      0         % 20
                   0    0    0    0           0      0      0         % 21
                 ];
```

Fig. 6.2: Matrix of beam element cards *EC_beam* as specified in input file **.m07* for linear and non-linear beam elements. In this example there are 4 (= no. of columns) element cards specified for linear beam elements (type 102) and three element cards specified for non-linear 2DOF truss elements (type 111).

Example 6.3:

```
% --- matrix of shell element cards  EC_shell(5,:) -----------------------------
%
%     EC_shell(    1, :)   element concept
%                          = 1 plane anisotropic shell element, constant thickness
%                              Reissner/Mindlin plate theory
%     EC_shell( 2: 3, :)   index no. of respective parameter vector
%     EC_shell(    5, :)   shell reference input CS
%                          = 1 local element Cs
%                          = 2 uniform local/global element Cs
%                          = 3 uniform local element Cs

      EC_shell = [ [1 1];   % element concept
                   [1 1];   % material law (Es index no.)
                   [1 2];   % shell thickness  (dicke index no.)
                   [3 2];   % mass density  (Roh index no.)
                   [1 1];   % shell reference input CS
                 ];
```

Fig. 6.3: Matrix of shell element cards *EC_shell* as specified in input file **.m08*. In this example there are 2 (= no. of columns) element cards specified.

Element cards comprise complete sets of assignments to parameters. They comprise assignments to parameter vectors, e.g. material law, and in addition some parameters are set directly, e.g. angle of principal axis. Each column of the element card matrices represents a different element card. They are assigned to elements during meshing. The beam and shell properties specified in *ec_beam* and *ec_shell* are assigned to all the structural elements generated within the geometry elements via a coincidence vector *ec_*_koi specified in file *.m04.* There is a coincidence vector for each type of geometry element, e.g. *ec_flarot_koi* for *flarot* geometry elements, *ec_l_koi* for *linien* geometry elements.

Previous MATFEM releases used beam and shell assignment vectors for each element property, e.g. *ihooke*, *idicke*. The introduction of element cards does not change this basic idea. In fact, all properties assigned in an element card are assigned automatically to the corresponding beam and shell assignment vectors. The assignment vectors assign the beam/shell elements in ascending order to the corresponding parameters of the parameter vectors. Therefore, an assignment vector always has *nbalk* or *nshel* elements respectively.

Note: The user may either use the element cards or the beam and shell assignment vectors but he must not use them both in the same application.

6.1 Beam Element

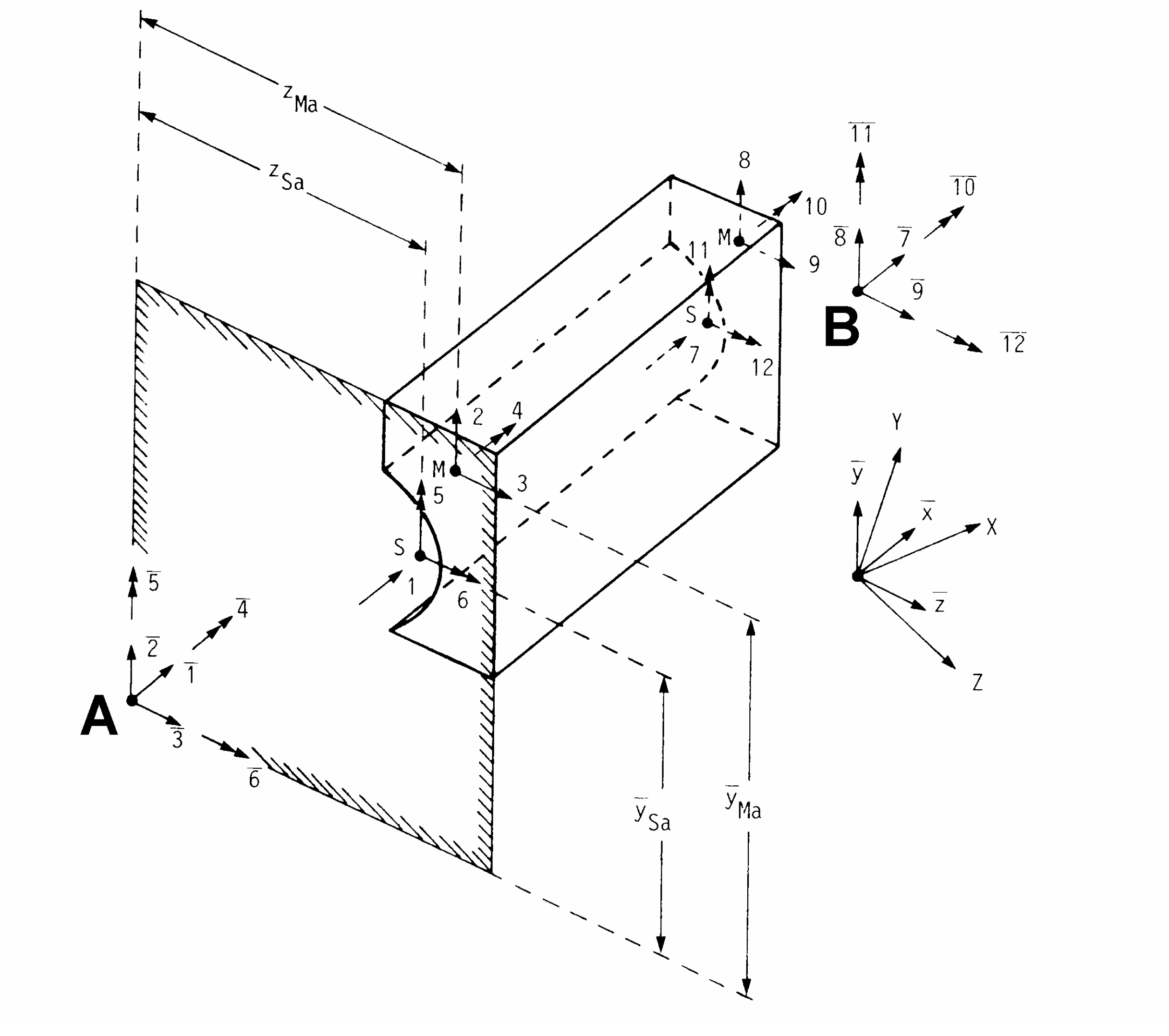


Fig. 6.1.1: General beam element with eccentric junction points A,B in 3-D-space

general: solid beam of constant cross section in 3-D space, eccentric nodal points, St Venant torsion

Beam types applicable to statics and dynamics:

BERNOULLI beam theory (no shear deformation)	beam_type 101
TIMOSHENKO beam theory (shear deformation)	beam_type 102
Two Node Spring Element (realized by decoupled beam stiffness terms)	beam_type 105
Two Node Spring Element with user specified stiffness and mass properties	beam_type 100

Non-linear 2DOF truss elements (non-linear dynamic response analysis only):

nonlinearity of 'power' type	beam_type 111
nonlinearity of 'Signum/Coulomb' type	beam_type 112
stiffness nonlinearity of 'arctan' type	beam_type 113
stiffness nonlinearity of 'arcsinh' type	beam_type 114
nonlinearity of 'clearance' type	beam_type 115
Gaul element (2 Parameters)	beam_type 116

reference:	*[Link 1], [Gruber 1], [Meyer 1]*	
number of nodes:	2	
number of DOF:	12	
degrees of freedom:	x, y, z	translational DOF's
	xx, yy, zz	rotational DOF's
beam orientation in 3-D space:	local element coordinate system:	
	x- axis	along the beam from node A to B
	y, z- axis	coincide with the principal axis of the cross section

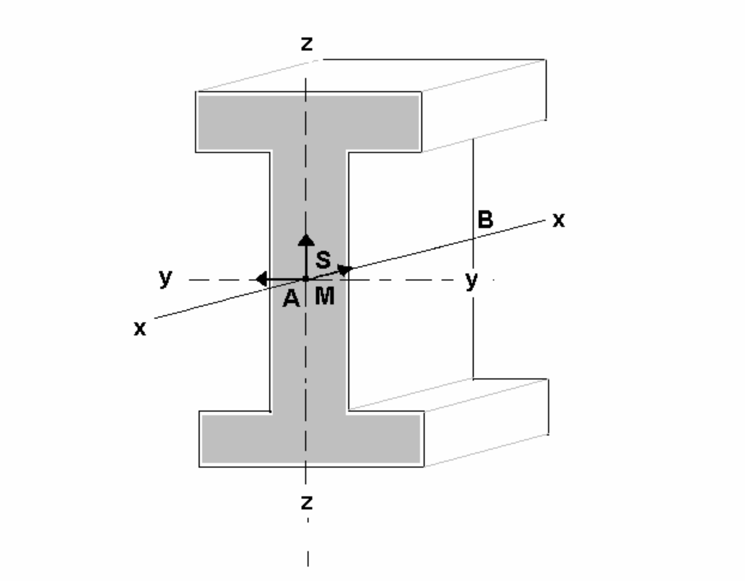


Fig. 6.1.2: beam cross section, principal axis, local beam xyz- axis

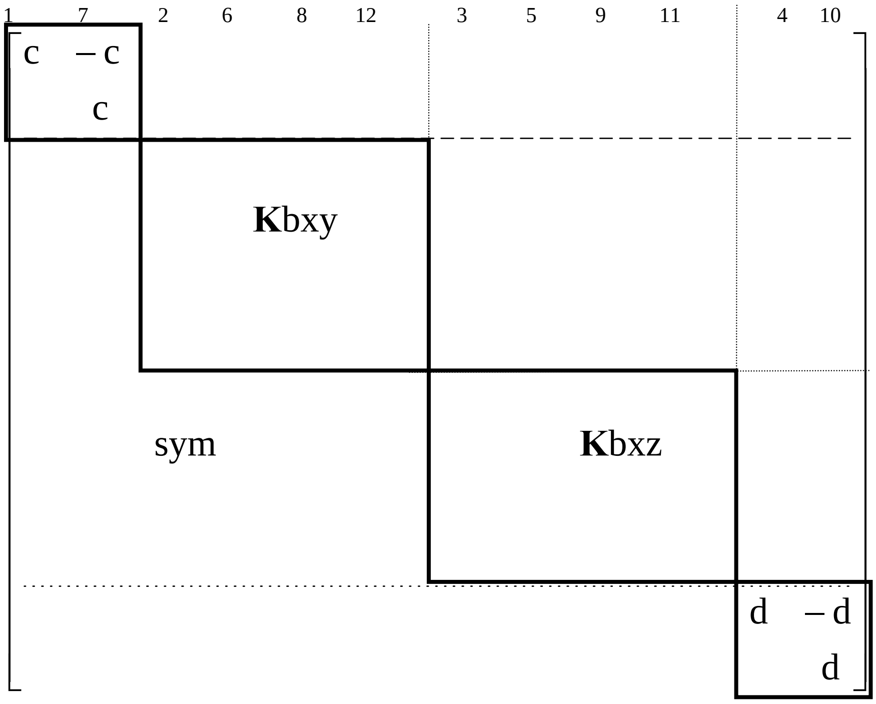
The principal axis and their respective principal moments of inertia I_1, I_2 are defined by

- y axis , 'starke Achse, Biegung um y, Biegung um y verursacht Verschiebung Uz und Verdrehung Uyy in der xz Ebene '
  'strong axis, bending w.r. to the y axis causes displacement Uz and rotation Uyy in xz plane'
  $I_1 = I_{yy} = I_{max}$ maximum moment of inertia (MATFEM parameter vector: Jk)

- z axis , 'schwache Achse, Biegung um z verursacht Verschiebung Uy und Verdrehung Uzz in der xy Ebene '
  'weak axis, bending w.r. to the z axis causes displacement Uy and rotation Uzz in xy plane'
  $I_2 = I_{zz} = I_{min}$ minimum moment of inertia (MATFEM parameter vector: Jkk)

Due that the y and z axis are the principal axis of inertia the respective centrifugal moment is zero (!). For each beam element the user must specify *I_1, I_2* and the angle *dgamma*, which specifies the orientation of the local/global coordinate system, and thereby specifies the spatial orientation of the cross section. He can check the orientation of each beam in MATFEM postprocessor, where *I_1* and *I_2* are displayed using the 'double T (H)' section symbol according fig. 6.1.2.

The beam element stiffness matrix in local coordinates is assembled from the longitudinal, bending and torsion stiffnesses as shown in fig. 6.1.3 .



where

$c = \frac{EA}{l} = K_x$ longitudinal stiffness

$d = \frac{GI_T}{l} = K_{xx}$ torsional stiffness

$\mathbf{K}_{xy}$ bending stiffness matrix w.r. to xy plane
(bending w.r. to z-axis, shear forces in y axis direction)

BERNOULLI beam theory (no shear deformation)

```
Kbxy = (E*Imin/l^3)*[ 12       6*l      -12       6*l
                        6*l     4*l^2    -6*l      2*l^2
                      -12      -6*l       12      -6*l
                        6*l     2*l^2    -6*l      4*l^2 ];
```

TIMOSHENKO beam theory (shear deformation)

```
phiy = 12*E*Imin/G/A(2)/l^2;  % A(2) == Asy (Schubfläche)
h1   = (1+phiy);

Kbxy = (E*Imin/l^3/h1)*[ 12       6*l          -12        6*l
                          6*l   (4+phiy)*l^2   -6*l     (2-phiy)*l^2
                        -12      -6*l          12        -6*l
                          6*l   (2-phiy)*l^2    -6*l     (4+phiy)*l^2 ];
```

Fig. 6.1.3: Element stiffness matrix of a beam element in local coordinates. Node numbering w.r. to fig. 6.1.1.

$\mathbf{K}_{xz}$ bending stiffness matrix w.r.t. to xz plane
(bending w.r.t. to y-axis, shear forces in z axis direction)

BERNOULLI beam theory (no shear deformation)

```
Kbxz = (E*Imax/l^3)*[ 12       -6*l       -12       -6*l
                      -6*l      4*l^2      6*l       2*l^2
                     -12        6*l       12        6*l
                      -6*l      2*l^2      6*l       4*l^2 ];
```

TIMOSHENKO beam theory (shear deformation)

```
phiz = 12*E*Imax/G/A(3)/l^2;   % A(3) == Asz (Schubfläche)
h1   = (1+phiz);

Kbxz = (E*Imax/l^3/h1)*[ 12       -6*l            -12       -6*l
                         -6*l   (4+phiz)*l^2      6*l   (2-phiz)*l^2
                        -12        6*l            12        6*l
                         -6*l   (2-phiz)*l^2      6*l   (4+phiz)*l^2 ];
```

A cross section
Axy shear area where shear forces in y direction apply, the x axis is perpendicular to this area
Axz shear area where shear forces in z direction apply the x axis is perpendicular to this area
A, Axy, Axz are set in the MATFEM parameter vector Ak(:, 1), Ak(:,2), Ak(:,3)

E YOUNG's modulus, set in the MATFEM parameter vector Ek
G shear modulus, " " Gk
I torsional moment of inertia, " " Tk
I_{max} maximum area moment of inertia, " " Jk
I_{min} minimum area moment of inertia, " " Jkk
l element length (calculated automatically from node distance A,B)

Fig. 6.1.3, continued: Element stiffness matrix of a beam element in local coordinates. Node numbering w.r.t. to fig. 6.1.1.

Two node spring element (beam_type 105):

The general beam element can also be used as a two node spring element with decoupled stiffness terms for longitudinal, torsional, bending and shear stiffness. Thereby the two node spring element can be used to model

a longitudinal spring with respect to the local x direction
a torsional spring with respect to the local xx direction
two bending springs with respect to the local yy and zz direction respectively
two shear springs with respect to the local y and z direction.

Therefore the stiffness matrix of the general beam element is modified acc. fig. 6.1.4.

Note:
All rotation related terms of fig. 6.1.4 which are affected by the shear stiffness K_{xy} and K_{xz} are needed to fulfill the equilibrium conditions of the element.

Note:
The stiffness terms of the two node spring element *beam_type 105* can not be specified directly as for *beam_type 100*. They have to be specified by the equivalent beam parameters like beam length, cross section, shear area, YOUNG's modulus, etc. The nodal points A,B of the two node spring element **must not** coincide.

$$
\begin{array}{cc|cccc|cccc|cc}
1 & 7 & 2 & 6 & 8 & 12 & 3 & 5 & 9 & 11 & 4 & 10 \\
c & -c & & & & & & & & & & \\
 & c & & & & & & & & & & \\
\hline
 & & e & \frac{le}{2} & -e & \frac{le}{2} & & & & & & \\
 & & & \frac{l^2e}{4}+a & \frac{-le}{2} & -a+\frac{l^2e}{4} & & & & & & \\
 & & & & e & \frac{-el}{2} & & & & & & \\
 & & & & & \frac{l^2e}{4}+a & & & & & & \\
\hline
 & & & & & & f & \frac{-lf}{2} & -f & \frac{-lf}{2} & & \\
 & & \text{sym} & & & & & \frac{l^2f}{4}+b & \frac{lf}{2} & -b+\frac{l^2f}{4} & & \\
 & & & & & & & & f & \frac{fl}{2} & & \\
 & & & & & & & & & \frac{l^2f}{4}+b & & \\
\hline
 & & & & & & & & & & d & -d \\
 & & & & & & & & & & & d
\end{array}
$$

where

$$c = \frac{EA}{l} = K_x$$ longitudinal stiffness

$$e = \frac{GA_{xy}}{l} = K_{xy}$$ shear stiffness (xy- plane)

$$f = \frac{GA_{xz}}{l} = K_{xz}$$ shear stiffness (xz-plane)

$$b = \frac{EI_{max}}{l} = K_{zz}$$ bending stiffness w.r. to local element z-axis

$$a = \frac{EI_{min}}{l} = K_{yy}$$ bending stiffness w.r. to local element y-axis

$$d = \frac{GI_T}{l} = K_{xx}$$ torsional stiffness

A cross section
Axy shear area where shear forces in y direction apply, the x axis is perpendicular to this area
Axz shear area where shear forces in z direction apply the x axis is perpendicular to this area
A, Axy, Axz are set in the MATFEM parameter vector Ak(:, 1), Ak(:,4), Ak(:,5)
E YOUNG’s modulus, set in the MATFEM parameter vector Ek
G shear modulus, " " Gk
I_T torsional moment of inertia, " " Tk
I_{max} maximum area moment of inertia, " " Jk
I_{min} minimum area moment of inertia, " " Jkk
l element length (calculated automatically from node distance A,B)

Fig. 6.1.4: Element stiffness matrix of a two node spring element. Node numbering w.r. to fig. 6.1.1.

Two node spring element with user specified stiffness and mass properties (beam_type 100):

For *beam_type 100* the element stiffness and mass matrices and the equivalent element forces must totally be specified by the user. These quantities are set in the *beam100_k*, *beam100_m* and *beam100_f* parameter matrices of input file **.m07*. Each column of *beam100_k* represents the 144 elements (12*12) of an element stiffness matrix. The columns of *beam100_m* represent different element mass matrices. The element forces at the 12 DOFs are specified as columns in the matrix *beam100_f*. The quantities in the parameter matrices can be given w.r. to the local or global coordinate system. The orientation of the element in 3 D space is equivalent to the general beam element. The local x-axis is determined as the straight line connection from node A to B. If nodes A and B coincide, an auxiliary node must be specified from the *beam100_auxnode* parameter vector to define the local x axis. The specification of the local y and z axis follows the definition for the general beam element. With respect to transformation, eccentric nodal points and hinges the element is also treated as a general beam element.

Note: The user must very carefully specify the parameter matrices, so that the equilibrium requirement is fulfilled for every *beam_type100* element.

```
% --- beam type [ 100 ] parameters ( 2 node, 12 DOF general spring element, nodes
may coincide)

      % --- auxiliary node coordinates in global CS, to specify the local element
      %     x axis, if element nodes coincide:      beam100_auxnode( :, 3)

            beam100_auxnode = [ berkoord( 5,:)+ [ 0 0 .1];
                                berkoord( 6,:)+ [ 0 0 .1] ];

      % --- element stiffness matrix -------------------------------------------
      %     beam100_k( 144, :)

            beam100_k = 1.0E8*[    eye( 6, 6)    -eye( 6, 6);  ...
                                  -eye( 6, 6)     eye( 6, 6)  ];
            beam100_k = reshape( beam100_k, 144, 1);


      % --- element mass matrix ------------------------------------------------
      %     beam100_m( 144, :)

            beam100_m= [];


      % --- equivalent element force vector ------------------------------------
      %     beam100_f( 12, :)

            beam100_f = [];
```

Fig. 6.1.4b: Parameter vectors to specify the element stiffness and mass matrices, the equivalent element forces and the auxiliary node coordinates to specify the beam orientation. Here, six springs are set to couple each respective DOF of Node A and B. The spring mass is zero.

Non-linear 2DOF truss element (beam_type: 111 - 116):

The geometrical definitions of the location of the general beam element in 3-D space can also be used to define non-linear 2DOF truss elements, with decoupled local non-linear stiffness and damping for the three translational and rotational DOF's. The stiffness and damping factors of these local non-linear elements can be specified directly in EC_beam. They do not depend on cross section parameters. The length of the element is only needed in case of local non-linear shear elements (local y-direction and local z-direction) where equilibrium is only obtained if additional moments are introduced at the nodal points. The nodal points A,B of the 2DOF truss element **must not** coincide.

As an example, the element stiffness matrices of a non-linear 2DOF truss element with non-linear stiffness defined for all coordinate directions are shown in Fig. 6.1.5.

local x-direction:

$$k_x \left[\begin{array}{cc} \scriptstyle 1 & \scriptstyle 7 \\ 1 & -1 \\ -1 & 1 \end{array}\right]$$

local xx-direction:

$$k_{xx} \left[\begin{array}{cc} \scriptstyle 4 & \scriptstyle 10 \\ 1 & -1 \\ -1 & 1 \end{array}\right]$$

local yy-direction:

$$k_{yy} \left[\begin{array}{cc} \scriptstyle 5 & \scriptstyle 11 \\ 1 & -1 \\ -1 & 1 \end{array}\right]$$

local zz-direction:

$$k_{zz} \left[\begin{array}{cc} \scriptstyle 6 & \scriptstyle 12 \\ 1 & -1 \\ -1 & 1 \end{array}\right]$$

local y-direction:

$$k_y \left[\begin{array}{cccc} \scriptstyle 2 & \scriptstyle 6 & \scriptstyle 8 & \scriptstyle 12 \\ 1 & l/2 & -1 & l/2 \\ & l^2/4 & -l/2 & l^2/4 \\ & & 1 & -l/2 \\ sym & & & l^2/4 \end{array}\right]$$

local z-direction:

$$k_z \left[\begin{array}{cccc} \scriptstyle 3 & \scriptstyle 5 & \scriptstyle 9 & \scriptstyle 11 \\ 1 & -l/2 & -1 & -l/2 \\ & l^2/4 & l/2 & l^2/4 \\ & & 1 & l/2 \\ sym & & & l^2/4 \end{array}\right]$$

l element length (calculated automatically from node distance A,B)

Fig. 6.1.5: Element stiffness matrices of a non-linear 2DOF truss element. Node numbering w.r. to fig. 6.1.1.

Note: These elements are only available in case of a non-linear dynamic response calculation (wahl = 8) but not for the calculation of static displacements (wahl = 3) or a linear dynamic response calculation (wahl = 2).

input data:

element allocation:	specified in *.m04	
material data:	specified in *.m06	
geometrical data:	specified in *.m07	
beam theory:	general beam:	BERNOULLI, → beam_type 101 TIMOSHENKO → beam_type 102
	two node spring element	isoparametric beam with one point integration of shear energy [Link 1] → beam_type 105
	two node spring element	user defined element stiffness and mass matrices, coinciding nodes → beam_type 100
	non-linear 2DOF truss element [Meyer 1]	→ beam_type 111 - 116

Beam theory selection:

The beam theory selection of the general beam element is controlled by cross section parameter matrix *Ak*:

BERNOULLI beam theory (no shear effects). *Ak* has to be specified as a (5, na) matrix, where rows 2,3,4,5 are set to zero

```
Ak = [  A(1,1)  ...  A(1,na) ;
         0       ...       0  ;
         0       ...       0  ;
         0       ...       0  ;
         0       ...       0 ]
```

where

na	number of different cross sections
A(1,i)	cross section

If for a given application only BERNOULLI beams are present it is sufficient to specify only the first row of *Ak*.

TIMOSHENKO beam theory (shear effects). *Ak* has to be specified as a (5, na) matrix, where rows 4,5 are set to zero

```
Ak = [   A(1,1)  ...   A(1,na) ;
         A(2,1)  ...   A(2,na) ;
         A(3,1)  ...   A(3,na) ;
            0       ...      0    ;
            0       ...      0       ]
```

where

na number of different cross sections

A(1,i) cross section

A(2,i) shear area Axy , where shear forces in y-direction (local coordinate system) apply

A(3,i) shear area Axz , where shear forces in z-direction (local coordinate system) apply

If for a given application only TIMOSHENKO and BERNOULLI beams are present it is sufficient to specify only the first <u>three</u> rows of *Ak*.

two node spring element. *Ak* has to be specified as a (5, na) matrix, where the two rows (row 2 and row 3, which are related to the shear areas of the TIMOSHENKO beam) must be set to zero

```
Ak = [   A(1,1)  ...   A(1,na) ;
           0         ...      0   ;
           0         ...      0   ;
         A(4,1)  ...   A(4,na) ;
         A(5,1)  ...   A(5,na)  ]
```

where

na number of different cross sections

A(1,i) cross section area. The longitudinal spring stiffness will be calculated by

$$K_x = \frac{EA_{1,i}}{l}$$

A(4,i) shear area $A_{xy,}$ where shear forces in y direction apply . The shear spring stiffness in xy will be calculated by

$$K_{xy} = \frac{GA_{4,i}}{l}$$

A(5,i) shear area $A_{xz,}$ where shear forces in z direction apply . The shear spring stiffness in xz will be calculated by

$$K_{xz} = \frac{GA_{5,i}}{l}$$

The bending stiffnesses of the two node spring element will be calculated by

$$K_{zz} = \frac{EI_{min}}{l} \qquad K_{yy} = \frac{EI_{max}}{l}$$

and the torsional stiffness by

$$K_{xx} = \frac{GI_{T}}{l}$$

NOTE: The user should be aware that the parameter *l* is common to all stiffness terms, (fig.6.1.4). The parameter *l* is the distance between the nodal points A,B (length of beam) which is automatically assigned within MATFEM.

Example 6.3:

A given structure with BERNOULLI beams, TIMOSHENKO beams and two node spring elements of *beam_type 105*:

```
Ak = [   A(1,1)   A(1,1)   A(1,3)  ;
            0     A(2,2)      0    ;
            0     A(3,2)      0    ;
            0        0     A(4,3)  ;
            0        0     A(5,3)     ]
```

Parameter vectors:

material law:

Ek YOUNG's modulus

Gk shear modulus

geometry

Ak cross section

Jk maximum moment of inertia I_{max}, inertia moment w.r. to the local yy axis of the principal axis coordinate system, ref. fig. 6.1.2

Jkk minimum moment of inertia I_{min} inertia moment w.r. to the local zz axis of the principal axis coordinate system, ref. fig. 6.1.2

Tk torsional area moment of inertia ($\equiv I_T$)

mass

Roh mass density

Beam element cards:

Beam element cards are specified in the matrix EC_beam(21, :). Each column of EC_beam represents a different element card which comprises assignments to parameter vectors, e.g. material law or a direct parameter input, e.g. angle of principal axis. Linear and non-linear elements can be defined.

a.) linear beam elements

EC_beam (1, :) element type→ here: linear: 101,102 or 105
(2, :) material law (*Ek* index no.)
(3, ;) cross section (*Ak* index no.)
(4, :) shear modulus (*Gk* index no.)
(5, :) maximum moment of inertia I_{max} (*Jk* index no.)
(6, :) minimum moment of inertia I_{min} (*Jkk* index no.)
(7, :) centroidal torsional I_T moment of inertia (*Tk* index no.)
(8. :) mass density (*Roh* index no.)
(9. :) angle of principal axis w.r. to beam reference system (radians)

distance: center of gravity / shear center:
(10. :) y-direction
(11. :) z-direction

eccentric nodal point connections
node A:
(12. :) distance y-direction: eccentric nodal point/ center of gravity
(13. :) distance z-direction: eccentric nodal point/ center of gravity
(14. :) distance y-direction: eccentric nodal point/ shear center
(15. :) distance z-direction: eccentric nodal point/ shear center

node B:
(16. :) distance y-direction: eccentric nodal point/ center of gravity
(17. :) distance z-direction: eccentric nodal point/ center of gravity
(18. :) distance y-direction: eccentric nodal point/ shear center
(19. :) distance z-direction: eccentric nodal point/ shear center

hinges at local nodes A or B or A,B (1= A, 2= B, 3 = A and B)
(20. :) hinges w.r. to local z axis
(21. :) hinges w.r. to local y axis

All input of EC_beam(10:21, :) for linear beam elements w.r. to the local coordinate system according fig 6.1.1

b.) linear two node spring element with user defined element stiffness and mass matrices

EC_beam (1, :) element type → here: 100
(2, :) auxilary node (beam100_auxnode index no.)
(3, ;) CS for element stiffness matrix input
= 0 global Cs
= 1 local element Cs
(4, :) element stiffness matrix (beam100_k index no.)
(5, :) CS for element mass matrix input
= 0 global Cs
= 1 local element Cs
(6, :) element mass matrix (beam100_m index no.)
(7, :) CS for equivalent element force vector input
= 0 global Cs
= 1 local element Cs
(8. :) equivalent element force vector
(9. :) angle of principal axis w.r. to beam reference system (radians)
(10. :) reserved
(11. :) reserved

eccentric nodal point connections
node A:
(12. :) distance y-direction: eccentric nodal point/ center of gravity
(13. :) distance z-direction: eccentric nodal point/ center of gravity
(14. :) distance y-direction: eccentric nodal point/ shear center
(15. :) distance z-direction: eccentric nodal point/ shear center

node B:
(16. :) distance y-direction: eccentric nodal point/ center of gravity
(17. :) distance z-direction: eccentric nodal point/ center of gravity
(18. :) distance y-direction: eccentric nodal point/ shear center
(19. :) distance z-direction: eccentric nodal point/ shear center

hinges at local nodes A or B or A,B (1= A, 2= B, 3 = A and B)
(20. :) hinges w.r. to local z axis
(21. :) hinges w.r. to local y axis

All input of EC_beam(12:21, :) w.r. to the local coordinate system according fig 6.1.1

c.) non-linear beam elements (2DOF truss elements)

Beam_type = 111

EC_beam (1, :) element type → here: non-linear: 111

nonlinear stiffness parameters w.r. to local coordinate system
(2, :) = type of nonlinear stiffness
(3, :) = kx
(4, :) = ky
(5, :) = kz
(6, :) = kxx
(7, :) = kyy
(8, :) = kzz

nonlinear damping parameters w.r. to local coordinate system
(9, :) = type of nonlinear damping
(10, :) = dx
(11, :) = dy
(12, :) = dz
(13, :) = dxx
(14, :) = dyy
(15, :) = dzz

(16, :) = angle dgamma of principal axis w.r. to beam reference system (radian)

(17 : 21, :) = 0 reserved

Beam_type = 112

EC_beam (1, :) element type → here: non-linear: 112

nonlinear stiffness parameters w.r. to local coordinate system
(2, :) = 0 (reserved)
(3, :) = kx
(4, :) = ky
(5, :) = kz
(6, :) = kxx
(7, :) = kyy
(8, :) = kzz

nonlinear damping parameters w.r. to local coordinate system
(9, :) = 0 (reserved)
(10, :) = dx
(11, :) = dy
(12, :) = dz
(13, :) = dxx
(14, :) = dyy
(15, :) = dzz

(16, :) = angle dgamma of principal axis w.r. to beam reference system (radian)

(17 : 21, :) = 0 reserved

Beam_type = 113

EC_beam (1, :) element type → here: non-linear: 113

nonlinear stiffness parameters w.r. to local coordinate system
(2, :) = 0 (reserved)
(3, :) = kx
(4, :) = ky
(5, :) = kz
(6, :) = kxx
(7, :) = kyy
(8, :) = kzz

horizontal deformation factor for arctan(factor*displacement)
(9, :) = 0 (reserved)
(10, :) = bx
(11, :) = by
(12, :) = bz
(13, :) = bxx
(14, :) = byy
(15, :) = bzz

(16, :) = angle dgamma of principal axis w.r. to beam reference system (radian)

(17 : 21, :) = 0 reserved

Beam_type = 114

EC_beam (1, :) element type → here: non-linear: 114

nonlinear stiffness parameters w.r. to local coordinate system
(2, :) = 0 (reserved)
(3, :) = kx
(4, :) = ky
(5, :) = kz
(6, :) = kxx
(7, :) = kyy
(8, :) = kzz

horizontal deformation factor for arcsinh(factor*displacement)
(9, :) = 0 (reserved)
(10, :) = bx
(11, :) = by
(12, :) = bz
(13, :) = bxx
(14, :) = byy
(15, :) = bzz

(16, :) = angle dgamma of principal axis w.r. to beam reference system (radian)

(17 : 21, :) = 0 reserved

Beam_type = 115

EC_beam (1, :) element type → here: non-linear: 115

nonlinear stiffness parameters w.r. to local coordinate system
(2, :) = 0 (reserved)
(3, :) = kxo - gap open
(4, :) = kxc - gap closed
(5, :) = bx - half gap length [m]
(6, :) = kyo - gap open
(7, :) = kyc - gap closed
(8, :) = by - half gap length [m]
(9, :) = kzo - gap open
(10, :) = kzc - gap closed
(11, :) = bz - half gap length [m]

(12 : 15, :) = 0 (reserved)

(16, :) = angle dgamma of principal axis w.r. to beam reference system (radian)

(17 : 21, :) = 0 (reserved)

Beam_type = 116

EC_beam (1, :) element type → here: non-linear: 116

nonlinear stiffness parameters w.r. to local coordinate system
(2, :) = 0 (reserved)
(3, :) = kx
(4, :) = ky
(5, :) = kz
(6, :) = kxx
(7, :) = kyy
(8, :) = kzz

nonlinear friction forces w.r. to local coordinate system
(9, :) = 0 (reserved)
(10, :) = hx
(11, :) = hy
(12, :) = hz
(13, :) = hxx
(14, :) = hyy
(15, :) = hzz

(16, :) = angle dgamma of principal axis w.r. to beam reference system (radian)

(17 : 21, :) = 0 reserved

All input of EC_beam for non-linear beam elements w.r. to the local coordinate system according fig 6.1.1

Direct beam assignment vectors:

Beam assignment vectors assign parameter vectors to beam elements or comprise the parameters directly for each element. Beam assignment vectors are always of length *nbalk.*

<u>*material law:*</u>

ihooke	material law (*Ek* index no.)

<u>*geometry:*</u>

iflaeche	cross section (*Ak* index no.)
ischubmod	shear section (*Gk* index no.)
itraegh	maximum moment of inertia I_{max} (*Jk* index no.)
itraeghk	minimum moment of inertia I_{min} (*Jkk* index no.)
itors	torsional area moment of inertia (*Tk* index no.)
dgamma	angle of principal axis w.r. to beam reference system (radians, details see below)

<u>*mass:*</u>

irohdi	mass density

<u>*hinges:*</u>

glbalk	beam numbers with hinges
glkmax	local node number (A or B) of hinges w.r. to local z axis
glkmin	local node number (A or B) of hinges w.r. to local y-axis

<u>*eccentric junctions:*</u>

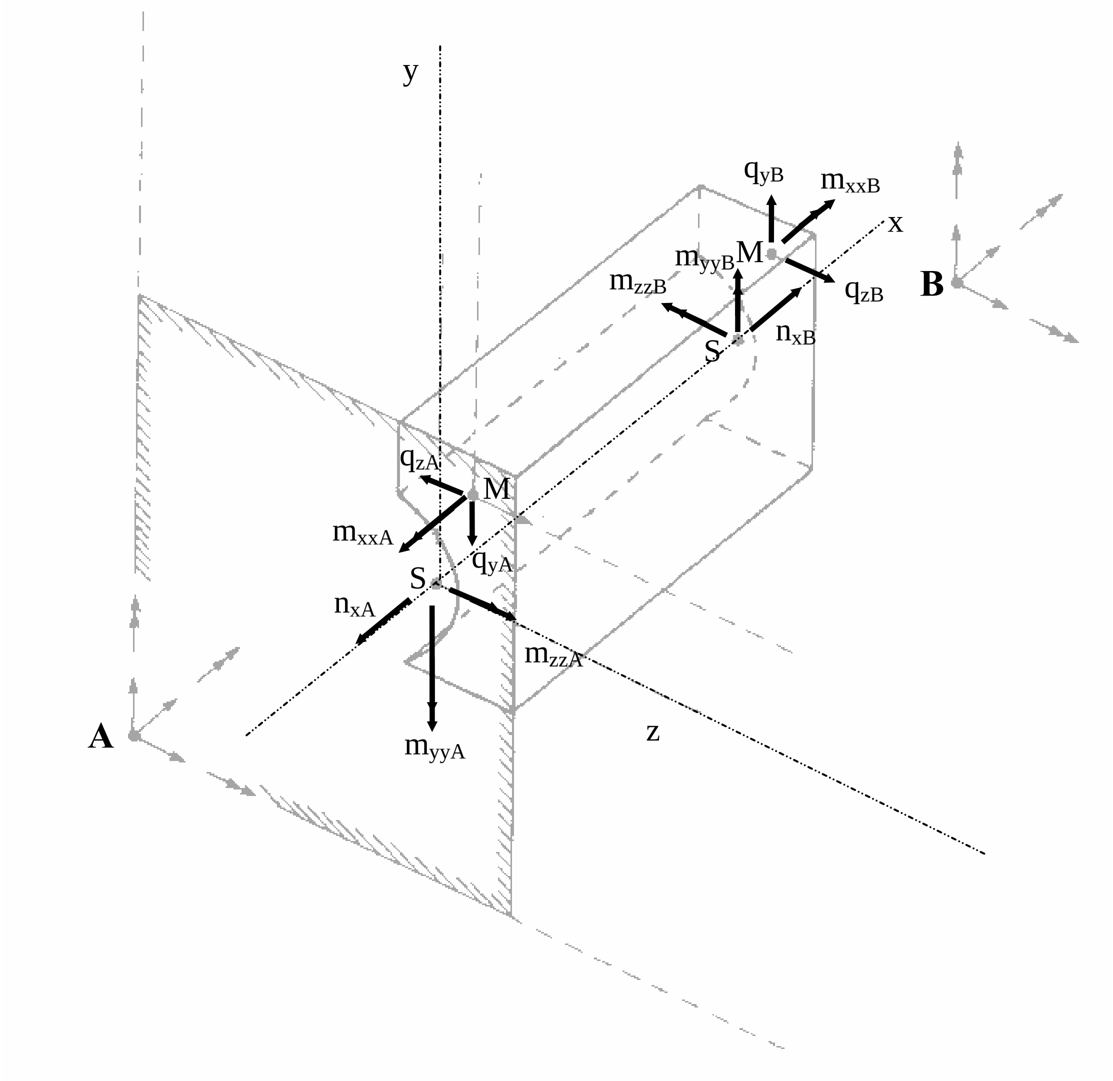
yMk	distance y-direction: center of gravity / shear center
zMk	distance z-direction: center of gravity / shear center
yMa	node A distance y-direction: eccentric nodal point/ shear center
yMb	node B distance y-direction: eccentric nodal point/ shear center
zMa	node A distance z-direction: eccentric nodal point/ shear center
zMb	node B distance z-direction: eccentric nodal point/ shear center
ySa	node A distance y-direction: eccentric nodal point/ center of gravity
ySb	node B distance y-direction: eccentric nodal point/ center of gravity
zSa	node A distance z-direction: eccentric nodal point/ center of gravity
zSb	node B distance z-direction: eccentric nodal point/ center of gravity

output data (static analysis):

stress resultants at nodal points A and B w.r.t. to local coordinate system (fig. 6.1.6)

n_{xA} , q_{yA} , q_{zA} , m_{xxA} , m_{yyA} , m_{zzA}

n_{xB} , q_{yB} , q_{zB} , m_{xxB} , m_{yyB} , m_{zzB}



M shear center, S center of gravity $m_{yy} = \int \sigma_{xx} z\, dA$ $m_{zz} = \int \sigma_{xx} y\, dA$

Fig. 6.1.6: Stress resultants of a beam element

Beam stresses:

The physical shape of a beam element cannot be specified in MATFEM. So MATFEM does not know whether the geometrical and material data refer e.g. to a solid beam with a rectangular cross section or to a U-shaped beam or an S-shaped beam with eccentric junction points. Therefore stresses can not be calculated within MATFEM. The user must calculate beam stresses externally using the beam stress resultants which are available from the MDB.

Center of gravity/ shear center, eccentric nodal points:

In the general case where the center of gravity *S* and the shear center *M* of a beam element do not coincide and the junction points are not on the center line of the beam area (like fig. 6.1.1) the user has to specify the appropriate distances:
 center of gravity /shear center
 eccentric nodal point/ center of gravity
 eccentric nodal point/ shear center
In the special case where the center of gravity and the shear center coincide and the junction points are on the center line of the beam area these quantities are all set to zero. In all other cases the quantities have to be specified for node A and node B of a beam (ref. fig. 6.1.1).

Beam orientation:

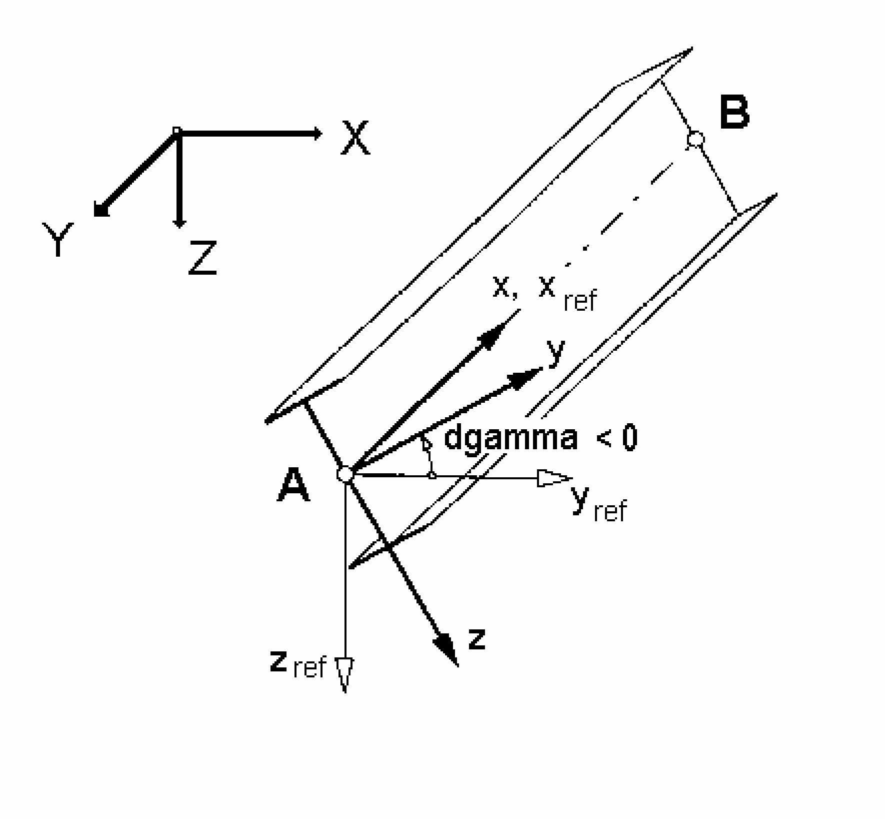


Fig. 6.1.7: Definition of angle *dgamma* to specify the orientation of a beam in 3-D space

The orientation of each beam in 3-D space has to be specified. In many FEM software packages the local x-axis y-axis and z- axis are specified by means of an auxiliary point within the x,y- plane thus fixing the orientation of the beam in 3-D space. In MATFEM the specification of the local y-axis and z-axis is made in a different manner:

The local coordinate system of a beam is always the principal axis coordinate system

In order to force the local y-axis and z-axis to coincide with the principal axis of the area moment an auxiliary cartesian coordinate systems is introduced. It is referred as the beam reference coordinate system (x_{ref} , y_{ref} , z_{ref}). The beam reference coordinate system is defined by

- x_{ref} coincides with the local x-axis. It points from node A to node B
- y_{ref} is calculated from the vector product x_{ref} and the global Z axis. The vector product yields y_{ref} which is perpendicular to the plane determined by x_{ref} and Z and therefore parallel to the global XY-plane. y_{ref} is sensed so that a right-handed screw turned from Z toward x_{ref} through the smaller of the angles determined by these vectors would advance in the direction of y_{ref} .
- z_{ref} is perpendicular to x_{ref} , y_{ref}. The orientation of z_{ref} follows the right-handed screw rule.

In the special case that the beam is parallel to the global Z axis the beam reference coordinate system is defined by

- x_{ref} coincides with the local x-axis. It points from node A to node B
- y_{ref} coincides with the global Y-axis
- z_{ref} is perpendicular to x_{ref} , y_{ref}. The orientation of z_{ref} follows the right-handed screw rule.

Using this definition it is possible to rotate the beam reference coordinate system around the local x, x_{ref} axis until the y_{ref} axis coincides with the y-axis of the principal axis coordinate system, i.e. the local beam coordinate system. The angle dgamma defines this rotation in radians. The angle dgamma is positive, if this rotation follows a rotation around the local x-axis in a positive sense.

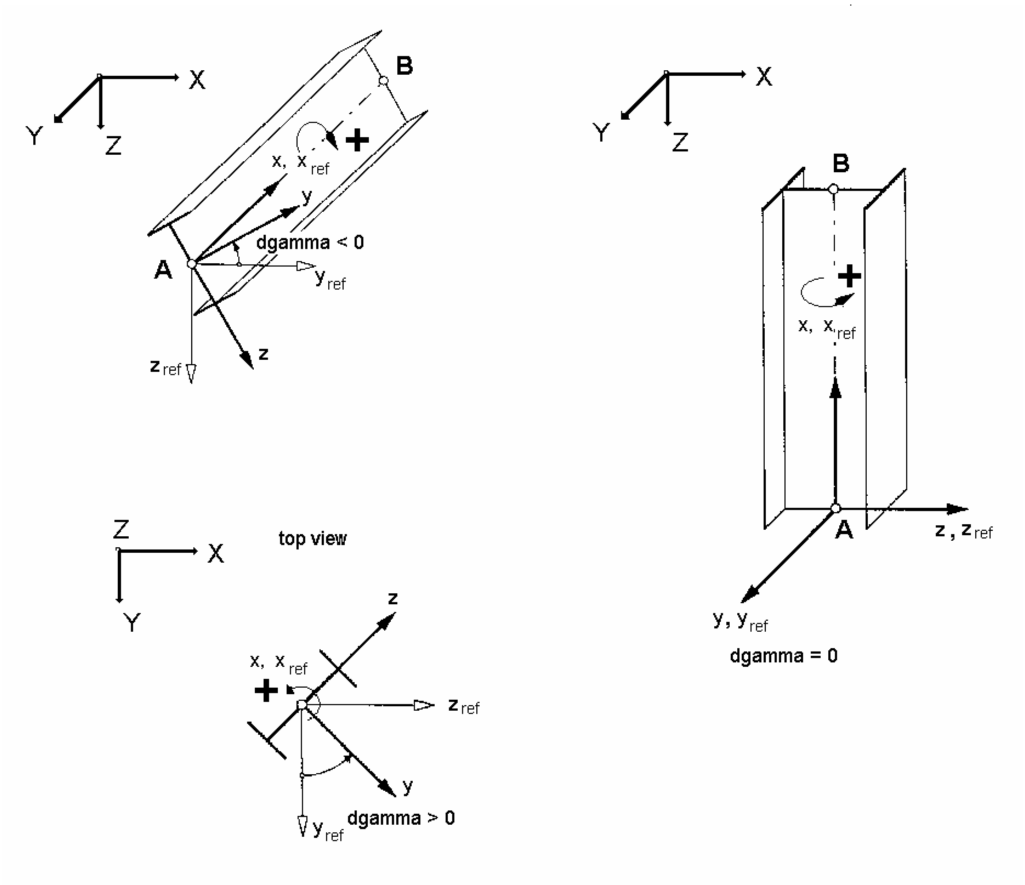


Fig 6.1.8: Definition of the angle *dgamma* to specify the orientation of a beam in 3 D space

In practice the angle *dgamma* is often not easy to specify for a given beam element in 3-D-space due to its rather complex definition which implies to imagine a rotation in 3-D space. Therefore it is recommended to always check the angle *dgamma* using the postprocessor of MATFEM where I_1 and I_2 are displayed using the double T section symbol according fig. 6.1.2.

NOTE:
All beam input data e.g. area moments of inertia, off-beam geometry ...etc. refer to the local coordinate system.

Example 6.4:

Consider the cross section of figure 6.1.9

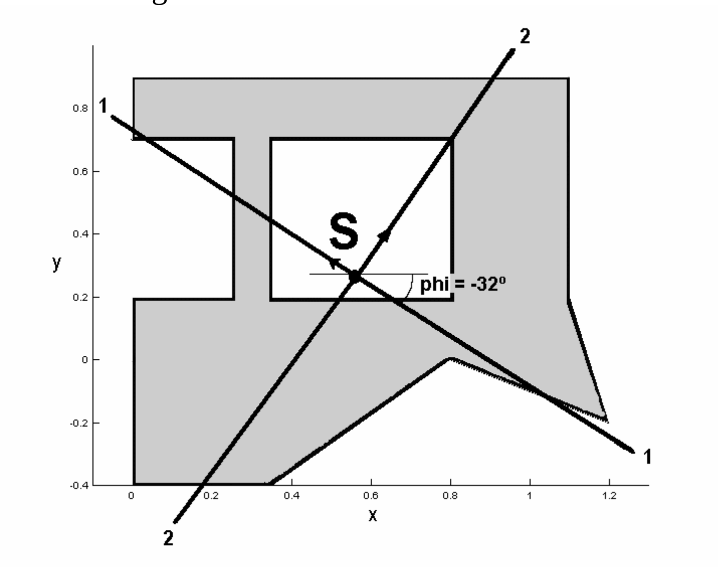
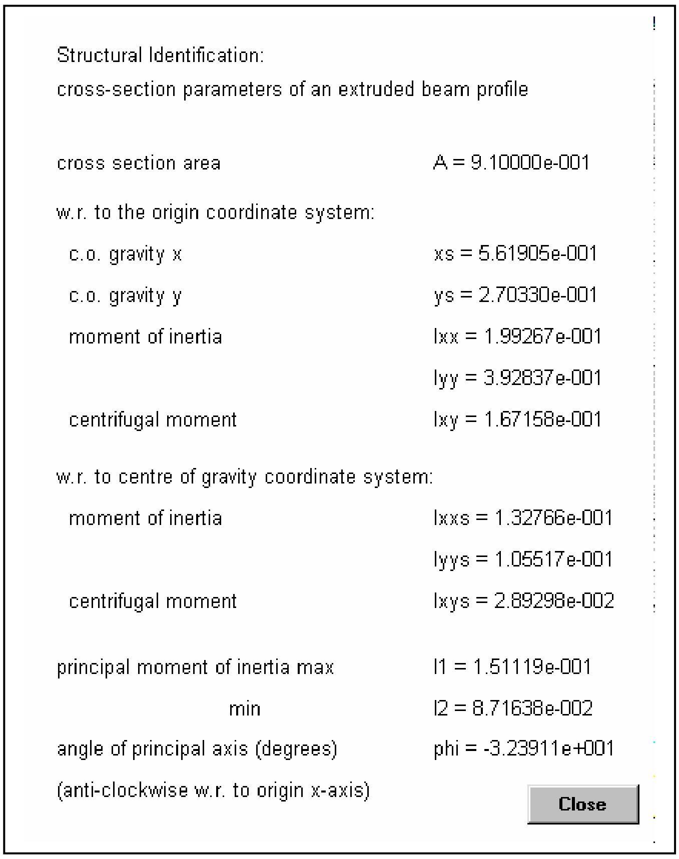


Fig. 6.1.9: cross section of an extruded beam profile. The cross section is specified in an arbitrary x, y coordinate system.

step 1:
The cross section parameters of this profile are calculated with MATFEM (wahl= 6)
The results are listed in fig 6.1.10.



```
Structural Identification:
cross-section parameters of an extruded beam profile

cross section area                        A = 9.10000e-001

w.r. to the origin coordinate system:
  c.o. gravity x                          xs = 5.61905e-001
  c.o. gravity y                          ys = 2.70330e-001
  moment of inertia                       Ixx = 1.99267e-001
                                          Iyy = 3.92837e-001
  centrifugal moment                      Ixy = 1.67158e-001

w.r. to centre of gravity coordinate system:
  moment of inertia                       Ixxs = 1.32766e-001
                                          Iyys = 1.05517e-001
  centrifugal moment                      Ixys = 2.89298e-002

principal moment of inertia max           I1 = 1.51119e-001
                            min           I2 = 8.71638e-002
angle of principal axis (degrees)         phi = -3.23911e+001
(anti-clockwise w.r. to origin x-axis)
```

Fig. 6.1.10: cross section parameters of an extruded beam profile of fig. 6.1.9.

The results are calculated w.r. to the origin of the x,y coordinate system of fig. 6.1.9 and with respect to the center of gravity (S). In addition, the principal moments of inertia I_1, I_2 and the respective angle *phi* of the principal axis w.r. to the x,y axis of fig. 6.1.9 are calculated.

step 2:
To assign this cross section to a clamped beam structure according fig 6.1.11 the user must specify the maximum of principal moments I_{max} to the parameter vector Jk and the minimum of principal moments I_{min} to Jkk:

$$Jk = I_1 \equiv I_{max} = 1.51E\text{-}01$$
$$Jkk = I_2 \equiv I_{min} = 0.87E\text{-}01$$

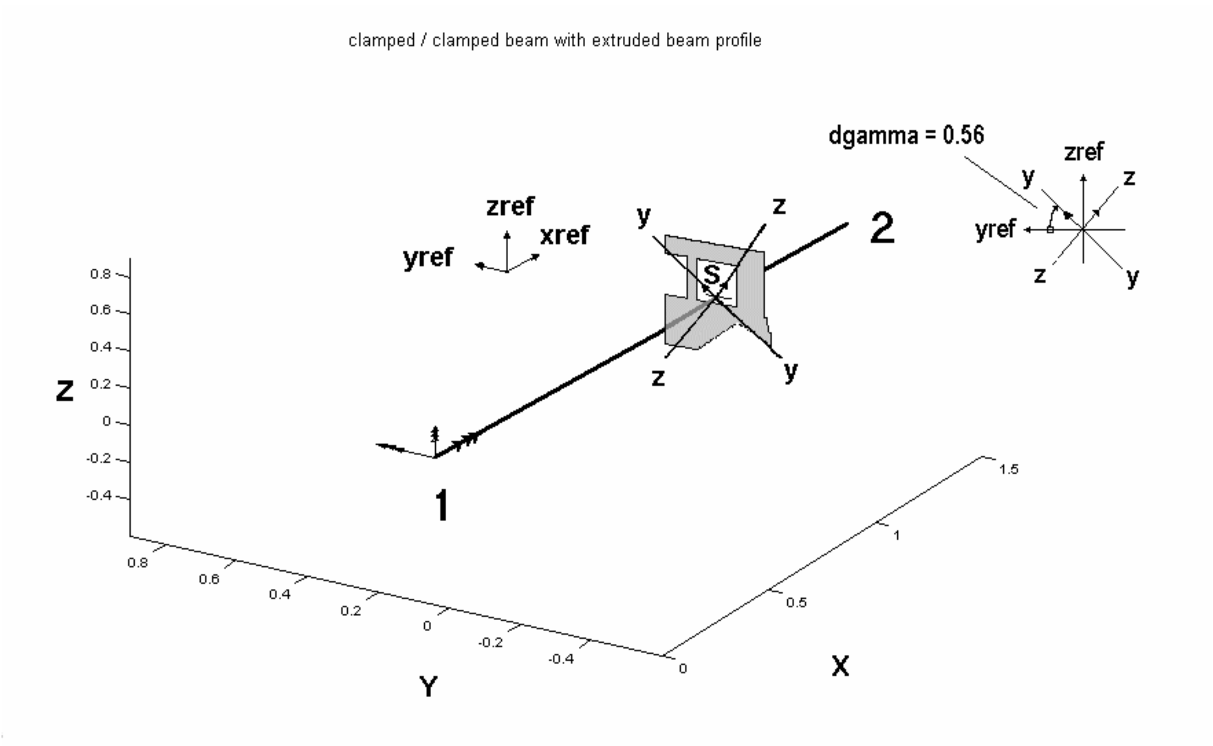


Fig. 6.1.11: Beam, clamped at node 1, extruded beam profile

The beam reference coordinate system in this case is defined by

- x_{ref} coincides with the local x-axis. It points from node A (1) to node B (2)
- y_{ref} is calculated from the vector product x_{ref} and the global Z axis. The vector product yields y_{ref} which is perpendicular to the plane determined by x_{ref} and Z and therefore parallel to the global XY-plane. y_{ref} is sensed so that a right-handed screw turned from Z toward x_{ref} through the smaller of the angles determined by these vectors would advance in the direction of y_{ref} .
- z_{ref} is perpendicular to x_{ref} , y_{ref}. The orientation of z_{ref} follows the right-handed screw rule.

To yield the angle *dgamma* the beam reference coordinate system must now be turned around the local x, x_{ref} axis until the y_{ref} axis coincides with the y-axis of the principal axis coordinate system, i.e. the local beam coordinate system. The angle *dgamma* defines this rotation in radians. In this case

dgamma = 32.39° * 2* pi / 360° = 0.565

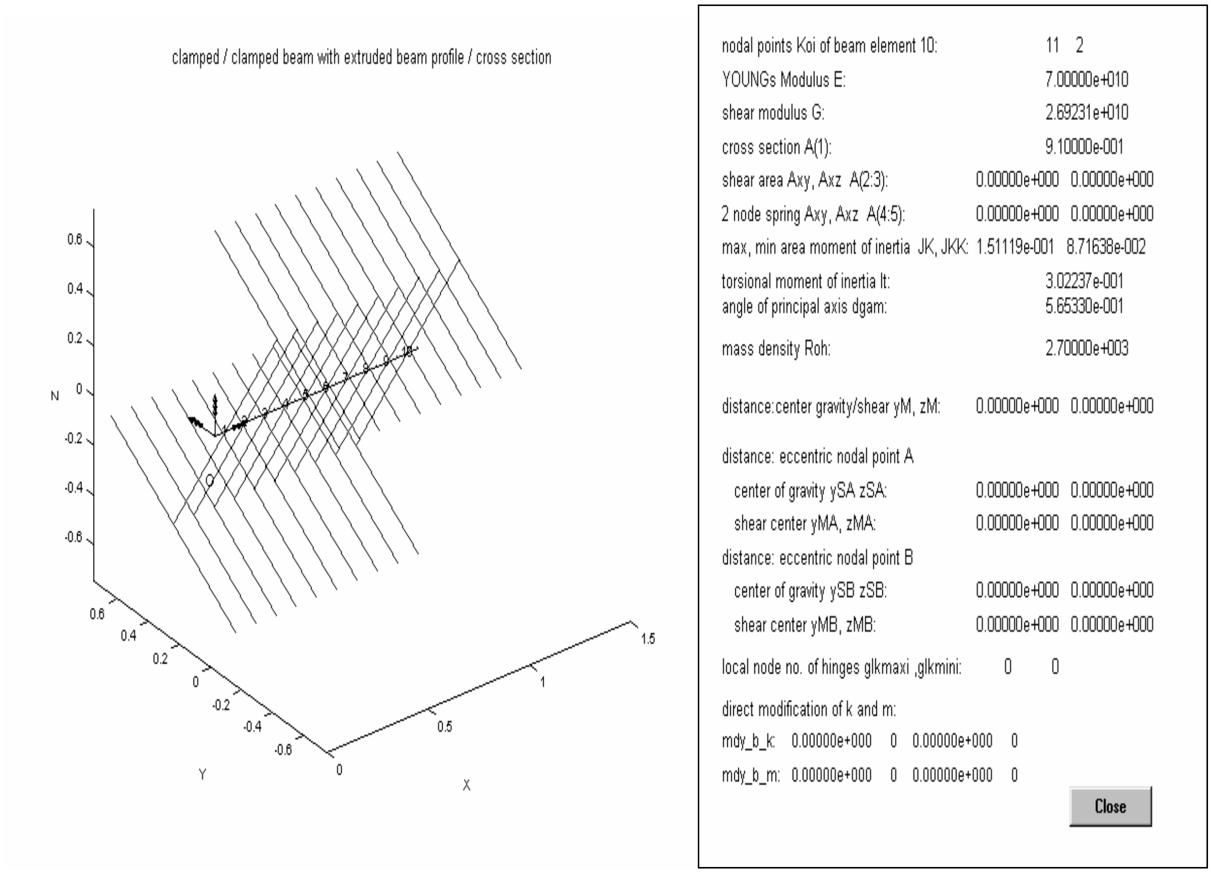
dgamma is positiv because this rotation follows a rotation around the local x-axis in a positive sense.

The user must then supply all other cross section parameters e.g. torsional moment of inertia, distance center of gravity/shear center.

Note: These additional parameters are yet <u>not</u> available from the MATFEM run with wahl = 6 and therefore must be calculated externally.

step 3:

The user can now start the MATFEM static (wahl = 3) or dynamic (wahl = 2) analysis. The orientation of the beam and the beam cross section parameters can be checked within the MATFEM postprocessor, fig. 6.1.12. The beam orientation is viewed using the double T section symbol according fig. 6.1.2.



```
nodal points Koi of beam element 10:                11   2
YOUNGs Modulus E:                                   7.00000e+010
shear modulus G:                                    2.69231e+010
cross section A(1):                                 9.10000e-001
shear area Axy, Axz  A(2:3):                0.00000e+000  0.00000e+000
2 node spring Axy, Axz  A(4:5):             0.00000e+000  0.00000e+000
max, min area moment of inertia  JK, JKK:  1.51119e-001  8.71638e-002
torsional moment of inertia It:                     3.02237e-001
angle of principal axis dgam:                       5.65330e-001

mass density Roh:                                   2.70000e+003

distance:center gravity/shear yM, zM:       0.00000e+000  0.00000e+000

distance: eccentric nodal point A
  center of gravity ySA zSA:                0.00000e+000  0.00000e+000
  shear center yMA, zMA:                    0.00000e+000  0.00000e+000
distance: eccentric nodal point B
  center of gravity ySB zSB:                0.00000e+000  0.00000e+000
  shear center yMB, zMB:                    0.00000e+000  0.00000e+000

local node no. of hinges glkmaxi ,glkmini:     0     0

direct modification of k and m:
mdy_b_k:  0.00000e+000   0   0.00000e+000   0
mdy_b_m:  0.00000e+000   0   0.00000e+000   0
```

Fig. 6.1.12: Clamped beam with an extruded beam profile. Input of cross section parameters View of beam orientation within the postprocessor

In the given example a static force in the global -Z direction is applied at the tip of the beam. Due to the principal axis of the beam cross section do not coincide with the global XYZ coordinate system the resulting displacement has not only a component in the global Z direction but components in both Y and Z direction, fig. 6.1.13.

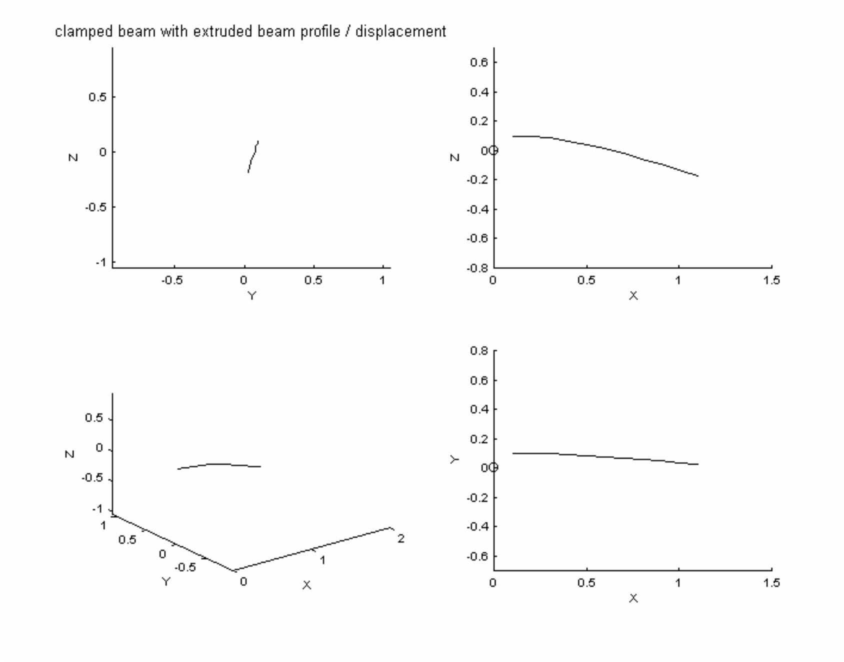


Fig. 6.1.13: Clamped beam with an extruded beam profile. Displacement due to a static force at the tip of the beam in global -Z direction.

6.2 Shell Element

MATFEM supports the following shell element types:

- 3 node, 18 DOF shell element

a) *element concept 1*
plane, anisotropic shell element, with constant thickness, consisting of a plane membrane constant strain triangle (CST) element (9 DOF’s) and a plane plate bending Discrete Kirchhoff Triangle (DKT) element (9 DOF’s), no coupling between bending and membrane forces (no transverse shear deformation)

- 4 node, 24 DOF shell element

a) *element concept 1*
plane, anisotropic shell element, with constant thickness, consisting of a plane membrane element (12 DOF’s) including rotational DOF's and a quasi conforming plane plate bending element (12 DOF’s), no coupling between bending and membrane forces,
Reissner-Mindlin plate theory, thermal loading

b) *element concept 2*
plane isoparametric plate element, independent shear formulation

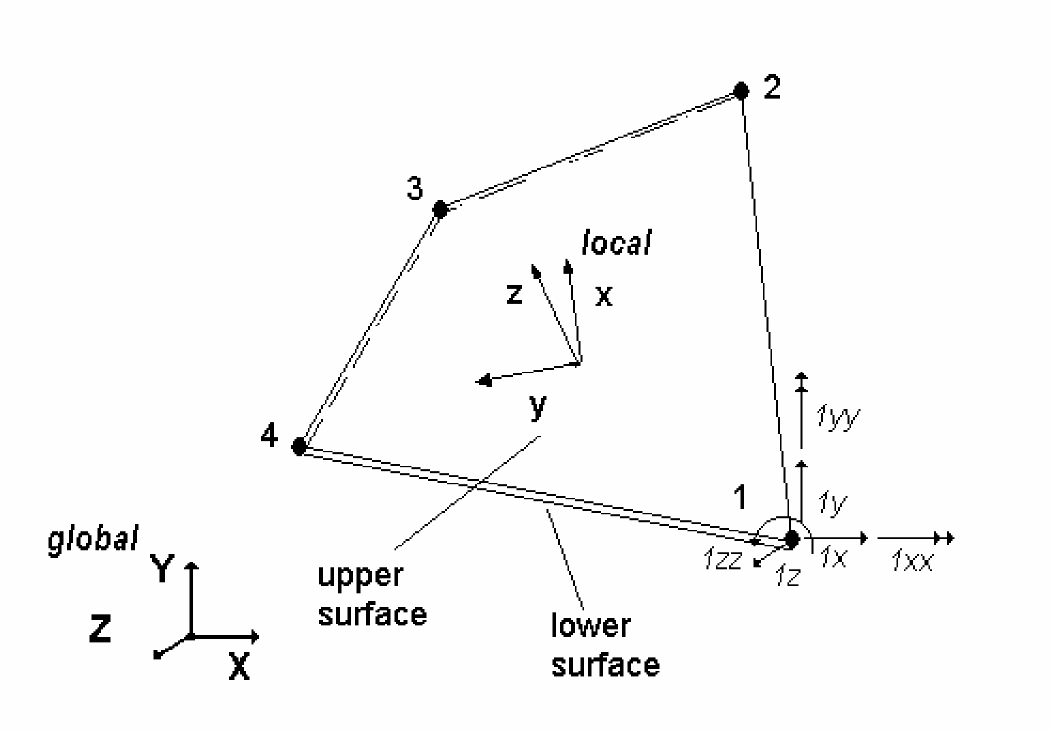
The different shell types are treated generally as 'shells' and therefore share the same variables for parameter vectors (e.g. *dicke*), element cards (*EC_shell*) and direct shell assignment vectors (e.g. *idicke*).
The overall number of shells *nshel* is given by

nshel = nshel3 + nshel 4

where

nshel3 overall number of 3 node shell elements
nshel4 overall number of 4 node shell elements

6.2.1 4 node Shell Element



The shell element has 4*6 = 24 local DOF's which are assembled in all related output as
[1x 1y 1z 1xx 1yy 1zz , 2x 2y 2z 2xx 2yy 2zz ,
3x 3y 3z 3xx 3yy 3zz , 4x 4y 4z 4xx 4yy 4zz]
The upper surface of a shell element is defined by the uppermost plane w.r. to the positive z direction of the local coordinate system
The local coordinate system of a shell element can be viewed within the MATFEM postprocessor

Fig. 6.2.1.1: 4-node shell element

a) ***element concept 1***

general: plane, anisotropic shell element, with constant thickness, consisting of a plane membrane element (12 DOF's) including rotational DOF's and a quasi conforming plane plate bending element (12 DOF's), no coupling between bending and membrane forces,
Reissner-Mindlin plate theory, thermal loading

reference: membrane element: [Long&Xu 1]
plate element: [Zou 1]
number of nodes: 4
number of DOF: 24
degrees of freedom: x, y, z translational DOF's
xx, yy, zz rotational DOF's

a) *element concept 2*

general: plane, anisotropic shell element, with constant thickness, independent shear formulation
Assumed Natural Strain (ANS) element, based on the Hughes T1 plate element.

reference: M. Fiolka: Entwicklung und Erprobung eines viereckigen ebenen Schalenelementes mit Schubverformung, Diplomarbeit, Universität Gh Kassel, Fachgebiet Leichtbau, Kassel 2001

T.J.R. Hughes: Finite Elements Based upon Mindlin Plate Theory with Particular Reference to the four Node bilinear isoparametric Element Journal of Applied Mechanics (1981) Vol.48 587-596

A.Tessler: An improved treatment of transverse shear in the Mindlin type four-node quadrilateral Element Computer Methods in applied Mechanics an Engineering 39 (1983) 311-335

number of nodes:	4	
number of DOF:	24	
degrees of freedom:	x, y, z	translational DOF's
	xx, yy, zz	rotational DOF's

local coordinate system:

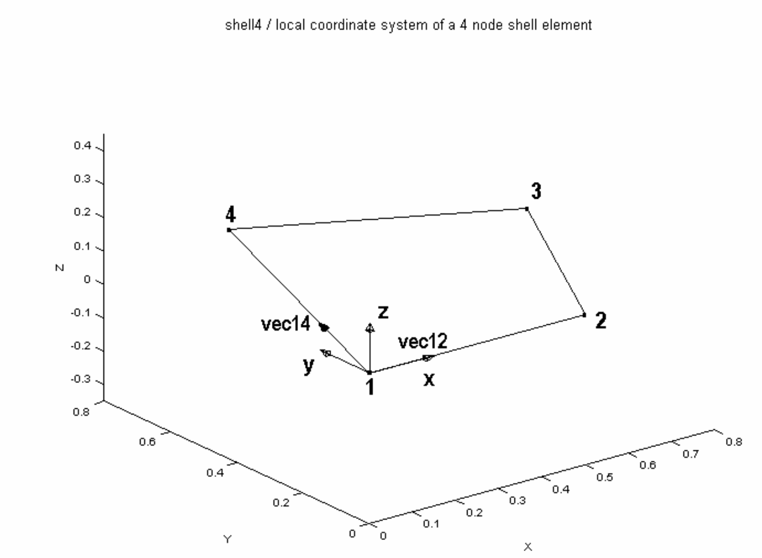


Fig. 6.2.1.2: local coordinate system of a 4-node shell element

The local coordinate system of a 4 node shell element is defined by:

x is given by the straight line between node 1 and node 2 (vec12). It points from node1 to node 2.

z is calculated from the vector product of *x* and the straight line between node 1 and node 4 (vec14). The vector product yields z which is perpendicular to the plane determined by *x* and *vec14*. The *z* axis is sensed so that a right-handed screw turned from *x* toward *vec14* through the smaller of the angles determined by these vectors would advance in the direction of *z*.

y is calculated from the vector product of *x* and *z*. The vector product yields y which is perpendicular to the plane determined by *x* and *z*. The *y* axis is sensed so that a right-handed screw turned from z toward *x* through the smaller of the angles determined by these vectors would advance in the direction of *y*.

The local coordinate system of each shell element can be viewed within the postprocessor

The overall shape of the shell element may be quadrilateral. As an extreme shape distortion, a triangular shape is possible if one node lies on the straight connection line between two other nodes. In this case an increase of the element stiffness must be expected.

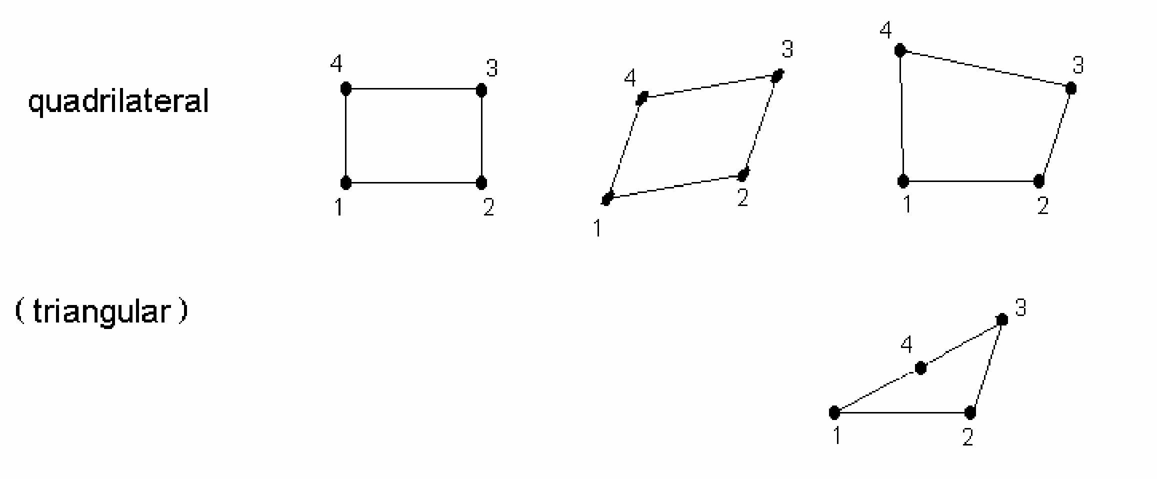


Fig. 6.2.1.3: Overall shapes of a 4-node shell element

The specification of the input parameters is given in chap. 6.2.3.

output data (static analysis):

stress resultants: at GAUß integration points, w.r.t. to local coordinate system

membrane forces	n_{xx} , n_{yy} , n_{xy}
bending moments	m_{xx} , m_{yy} , m_{xy}
shear forces	q_{xz} , q_{yz}

Note: All forces are related to a unit shell length (N/m)

element stresses: at GAUß integration points, w.r.t. to local coordinate system

membrane stresses (index m)	$\sigma_{xx,m}$, $\sigma_{yy,m}$, $\sigma_{xy,m}$
bending stresses (index b)	$\sigma_{xx,b}$, $\sigma_{yy,b}$, $\sigma_{xy,b}$
stresses at lower surface (index u)	$\sigma_{xx,u}$,$\sigma_{yy,u}$, $\sigma_{xy,u}$
principal stresses and angle of principal stresses at lower surface (index u)	$\sigma_{1,u}$, $\sigma_{2,u}$, alfa_u
v.-MISES stresses	$\sigma_{v,u}$
stresses at upper surface (index o)	$\sigma_{xx,o}$,$\sigma_{yy,o}$, $\sigma_{xy,o}$
principal stresses and angle of principal stresses at lower surface (index o)	$\sigma_{1,o}$, $\sigma_{2,o}$, alfa_o
v.-MISES stresses	$\sigma_{v,o}$

Optionally, stresses at nodal points can be calculated. The user has to set the switch *mitwert* in *.m13 to *mitwert* = 1. The stresses are simply averaged out of adjacent elements. Beam elements are <u>not</u> included in the averaging process. However, the results are only reasonable if **all** adjacent elements are defined in the **same local** coordinate system. The user has to take high care whether this simplified assumption is fulfilled for the parts of the structure under consideration. The averaged stresses can be viewed graphically on the structure using the MATFEM postprocessor.

Note: All stresses are related to a unit shell length $(\frac{N}{m^2\ m})$.

The angle of principal stresses is given in degrees.

The stress resultants are calculated from

- membrane forces n_{xx} , n_{yy} , n_{xy} (i = x, y)

$$n_{ii} = \int_t \sigma_{ii,m} dz$$

- bending moments m_{xx} , m_{yy} , m_{xy} (i = x, y)

$$m_{ii} = \int_t \sigma_{ii,b} z dz$$

- shear forces q_{xz} , q_{yz} (i = x, y)

$$n_{ij} = \int_t \sigma_{ij,s} dz$$

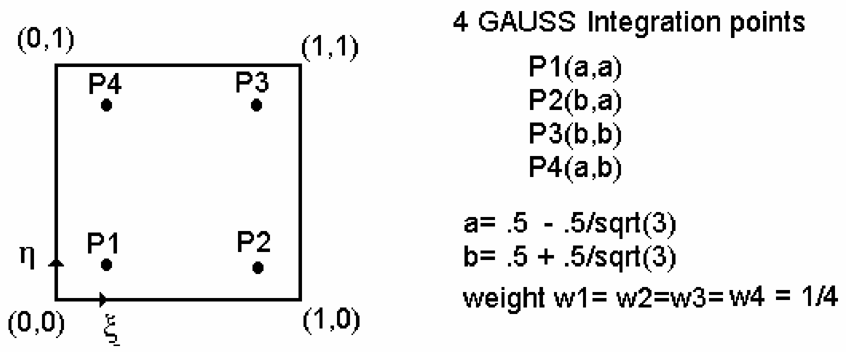


Fig. 6.2.1.4: GAUSS integration points of a 4 node shell element in the unit coordinate system

Due that the forces are calculated w.r. to a unit length and the section modulus (Widerstandsmoment) of the edges is given by $\frac{b * t^2}{6}$, where b specifies the actual edge length, the element stresses are calculated from

- stress at upper and lower surface of shell element

$$\sigma_o = \frac{n}{t} + \frac{6m}{t^2}$$ at upper surface

(index o derived from the German term oben)

$$\sigma_u = \frac{n}{t} - \frac{6m}{t^2}$$ at lower surface

(index u derived from the German term unten)

where
n normal forces
m bending forces

- main stresses $\sigma_{1,2}$ and main stress angle *alfa* (Hauptspannungen und Hauptspannungswinkel)

$$\sigma_{1,2} = \tfrac{1}{2}\left(\sigma_{xx} + \sigma_{yy}\right) \pm \sqrt{\tfrac{1}{4}\left(\sigma_{xx} - \sigma_{yy}\right)^2 + \sigma_{xy}^2}$$

$$\text{alfa} = \frac{1}{2}\arctan\left(\frac{2\sigma_{xy}}{\sigma_{xx} - \sigma_{yy}}\right)$$

- 'von MISES' stresses σ_V (Gestaltänderungsenergiehypothese)

$$\sigma_v = \sqrt{\sigma_1^2 + \sigma_2^2 - \sigma_1\sigma_2} \quad = \quad \sqrt{\sigma_x^2 + \sigma_y^2 - \sigma_x\sigma_y + 3\sigma_{xy}^2}$$

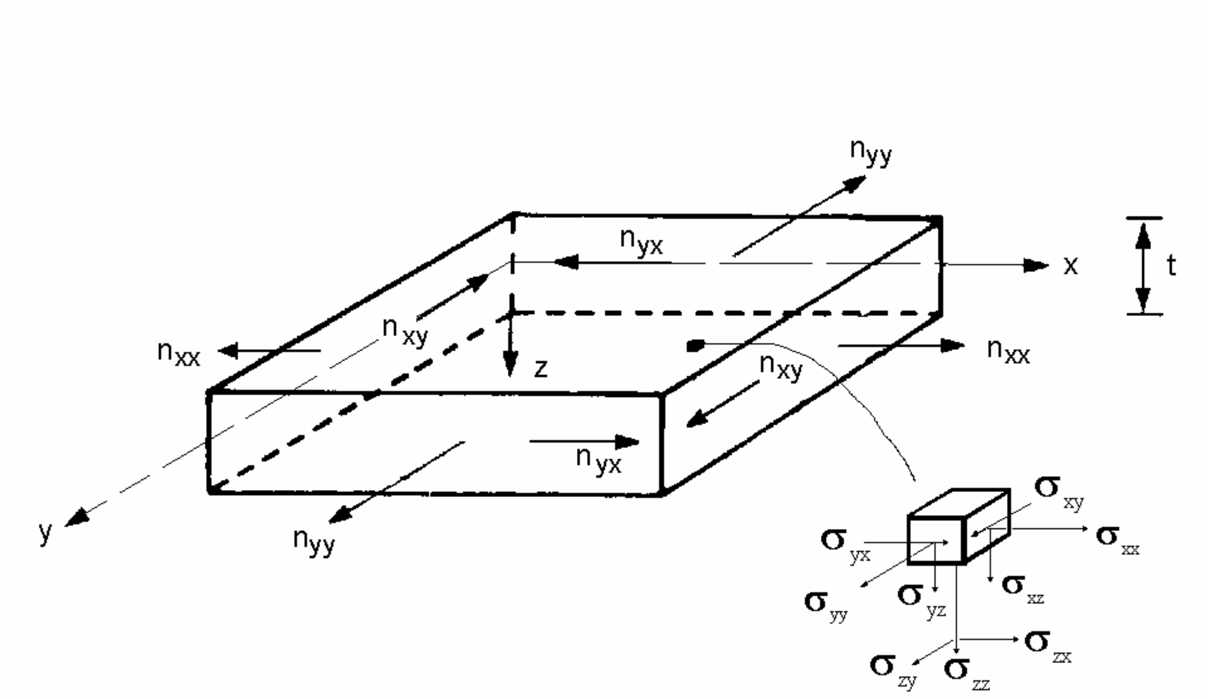


Fig. 6.2.1.5: Stress resultants of 4 node shell element, membrane forces

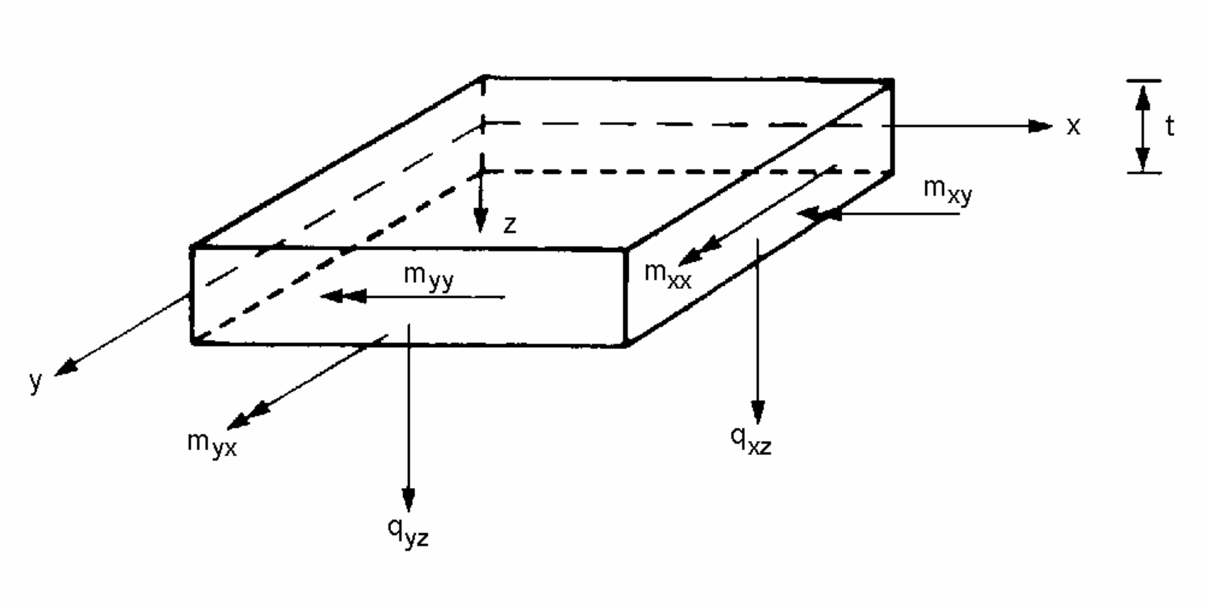


Fig. 6.2.1.6: Stress resultants of a 4 node shell element, bending moments and shear forces

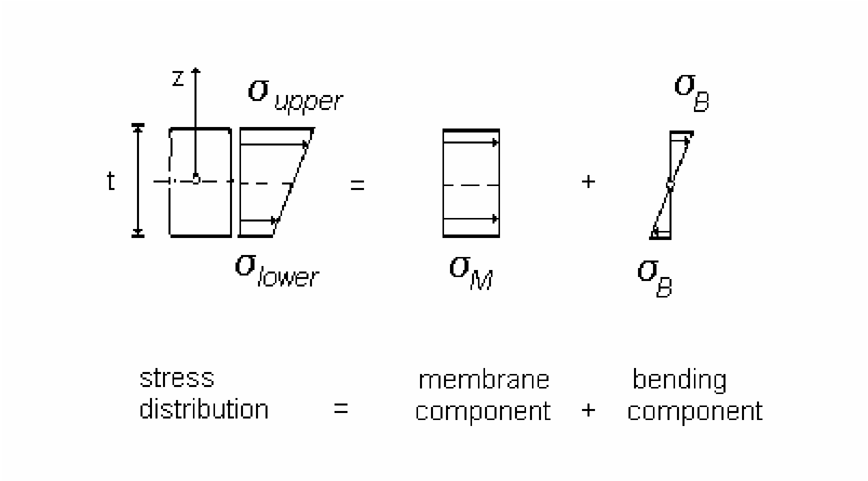
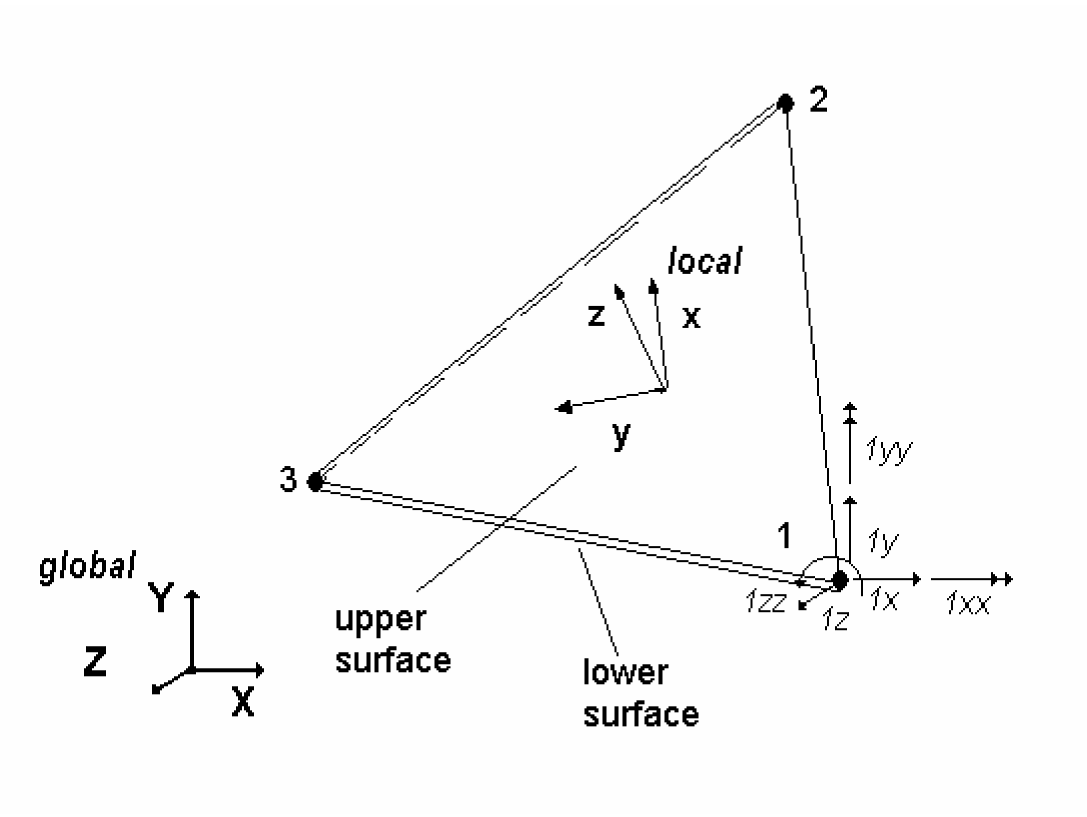


Fig. 6.2.1.7: Stresses at upper and lower surface of a 4 node shell element

6.2.2 3 node Shell Element



The shell element has 3*6 = 18 local DOF's which are assembled in all related output as
[1x 1y 1z 1xx 1yy 1zz , 2x 2y 2z 2xx 2yy 2zz , 3x 3y 3z 3xx 3yy 3zz]
The upper surface of a shell element is defined by the uppermost plane w.r. to the positive z direction of the local coordinate system
The local coordinate system of a shell element can be viewed within the MATFEM postprocessor

Fig. 6.2.2.1: 3-node shell element

general:	plane, anisotropic shell element, with constant thickness, consisting of a plane membrane constant strain triangle (CST) element (9 DOF's) and a plane plate bending Discrete Kirchhoff Triangle (DKT) element (9 DOF's), no coupling between bending and membrane forces (no transverse shear deformation)	
reference:	[Link 1], [Gröger 1], [Jeych&Kirk 1]	
number of nodes:	3	
number of DOF:	18	
degrees of freedom:	x, y, z	translational DOF's
	xx, yy, (zz)	rotational DOF's

Note:
The respective stiffness for the rotational DOF *zz*, which is not defined in the element evaluation, is set to 1.0E-12 by default.

local coordinate system:

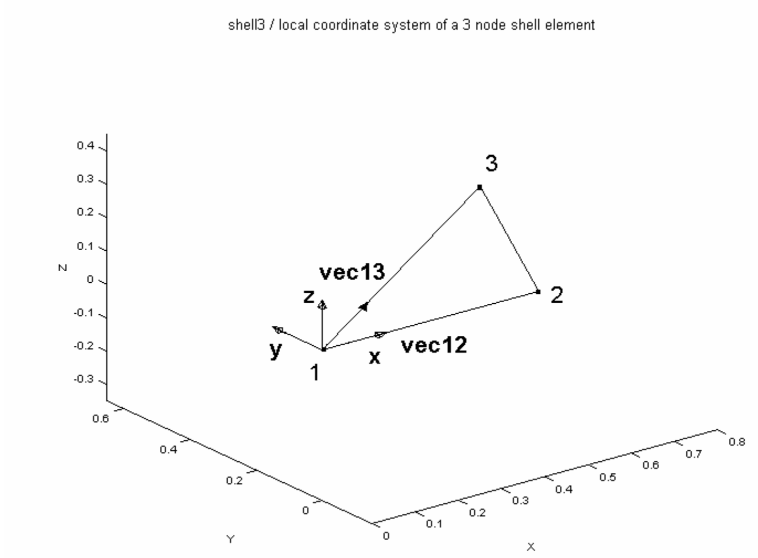


Fig. 6.2.2.2: local coordinate system of a 3-node shell element

The local coordinate system of a 3 node shell element is defined by:

x is given by the straight line between node 1 and node 2 (vec12). It points from node1 to node 2.

z is calculated from the vector product of *x* and the straight line between node 1 and node 3 (vec13). The vector product yields z which is perpendicular to the plane determined by *x* and *vec13*. The *z* axis is sensed so that a right-handed screw turned from *x* toward *vec13* through the smaller of the angles determined by these vectors would advance in the direction of *z*.

y is calculated from the vector product of *x* and *z*. The vector product yields y which is perpendicular to the plane determined by *x* and *z*. The *y* axis is sensed so that a right-handed screw turned from z toward *x* through the smaller of the angles determined by these vectors would advance in the direction of *y*.

The local coordinate system of each shell element can be viewed within the postprocessor

The specification of the input parameters is given in chap. 6.2.3.

output data (static analysis):

stress resultants: at GAUß integration points, w.r.t. to local coordinate system

membrane forces	n_{xx} , n_{yy} , n_{xy}
bending moments	m_{xx} , m_{yy} , m_{xy}
shear forces	q_{xz} , q_{yz}

Note: All forces are related to a unit shell length (N/m)

element stresses: at GAUß integration points, w.r.t. to local coordinate system

membrane stresses (index m)	$\sigma_{xx,m}$, $\sigma_{yy,m}$, $\sigma_{xy,m}$
bending stresses (index b)	$\sigma_{xx,b}$, $\sigma_{yy,b}$, $\sigma_{xy,b}$
stresses at lower surface (index u)	$\sigma_{xx,u}$, $\sigma_{yy,u}$, $\sigma_{xy,u}$
principal stresses and angle of principal stresses at lower surface (index u)	$\sigma_{1,u}$, $\sigma_{2,u}$, alfa_u
v.-MISES stresses	$\sigma_{v,u}$
stresses at upper surface (index o)	$\sigma_{xx,o}$, $\sigma_{yy,o}$, $\sigma_{xy,o}$
principal stresses and angle of principal stresses at lower surface (index o)	$\sigma_{1,o}$, $\sigma_{2,o}$, alfa_o
v.-MISES stresses	$\sigma_{v,o}$

Optionally, stresses at nodal points can be calculated. The user has to set the switch *mitwert* in *.m13 to *mitwert* = 1. The stresses are simply averaged out of adjacent elements. Beam elements are not included in the averaging process. However, the results are only reasonable if **all** adjacent elements are defined in the **same local** coordinate system. The user has to take high care whether this simplified assumption is fulfilled for the parts of the structure under consideration. The averaged stresses can be viewed graphically on the structure using the MATFEM postprocessor.

Note: All stresses are related to a unit shell length ($\frac{N}{m^2\ m}$).

The angle of principal stresses is given in degrees.

The stress resultants are calculated from

- membrane forces n_{xx} , n_{yy} , n_{xy} (i = x, y)

$$n_{ii} = \int_t \sigma_{ii,m} dz$$

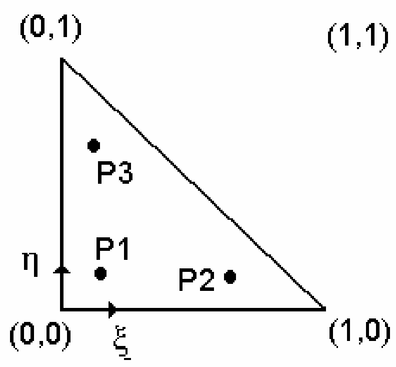
- bending moments m_{xx} , m_{yy} , m_{xy} (i = x, y)

$$m_{ii} = \int_t \sigma_{ii,b} z dz$$

- shear forces q_{xz}, q_{yz} are calculated from the derivatives of the bending moments

$$q_{xy} = \frac{\partial m_x}{\partial x} + \frac{\partial m_{xy}}{\partial y}$$

$$q_{yz} = \frac{\partial m_y}{\partial y} + \frac{\partial m_{xy}}{\partial x}$$



3 GAUSS Integration points
P1(1/6, 1/6)
P2(4/6, 1/6)
P3(1/6, 4/6)
weight w1= w2 = w3 = 1/3

Fig. 6.2.2.3: GAUSS integration points of a 3 node shell element in the unit coordinate system

Due that the forces are calculated w.r. to a unit length and the section modulus (Widerstandsmoment) of the edges is given by $\frac{b*t^2}{6}$, where b specifies the actual edge length, the element stresses are calculated from

- stress at upper and lower surface of shell element

$\sigma_o = \frac{n}{t} + \frac{6m}{t^2}$ at upper surface

(index o derived from the German term oben)

$\sigma_u = \frac{n}{t} - \frac{6m}{t^2}$ at lower surface

(index u derived from the German term unten)

- main stresses $\sigma_{1,2}$ and main stress angle *alfa* (Hauptspannungen und Hauptspannungswinkel)

$$\sigma_{1,2} = \tfrac{1}{2}\left(\sigma_x + \sigma_y\right) \pm \sqrt{\tfrac{1}{4}\left(\sigma_x - \sigma_y\right)^2 + \sigma_{xy}^2}$$

$$\text{alfa} = \frac{1}{2}\arctan\left(\frac{2\,\sigma_{xy}}{\sigma_x - \sigma_y}\right)$$

- 'von MISES' stresses σ_V (Gestaltänderungsenergiehypothese)

$$\sigma_V = \sqrt{\sigma_1^2 + \sigma_2^2 - \sigma_1\sigma_2} \quad = \quad \sqrt{\sigma_x^2 + \sigma_y^2 - \sigma_x\sigma_y + 3\sigma_{xy}^2}$$

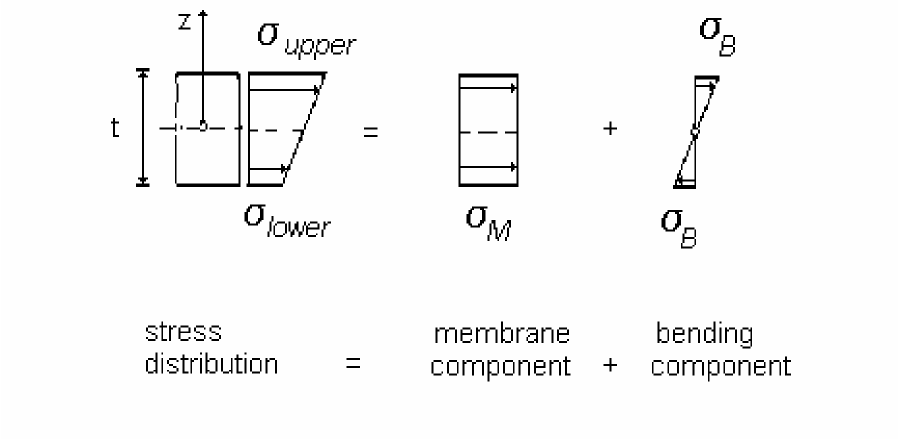


Fig. 6.2.2.6: Stresses at upper and lower surface of a 3 node shell element

6.2.3 Specification of shell input parameter

input data:

element allocation:	specified in *.m04
material data:	specified in *.m06
element type and geometrical data:	specified in *.m08

Parameter vectors:

material law:

Ast symmetric in-plane stiffness matrix

$$\begin{bmatrix} n_{xx} \\ n_{yy} \\ n_{xy} \end{bmatrix} = \begin{bmatrix} Ast(1,1) & Ast(1,2) & Ast(1,3) \\ & Ast(2,2) & Ast(2,3) \\ sym & & Ast(3,3) \end{bmatrix} \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{bmatrix}$$

where

ε	in-plane strain
γ	shear strain
n	membrane forces

only input of upper triangle of *Ast* required:

```
Ast(:,1:6) = [  Ast(1,1),  Ast(1,2), Ast(1,3), ...
                Ast(2,2),  Ast(2,3), Ast(3,3)  ]
```

Dst symmetric bending stiffness matrix

$$\begin{bmatrix} m_{xx} \\ m_{yy} \\ m_{xy} \end{bmatrix} = \begin{bmatrix} Dst(1,1) & Dst(1,2) & Dst(1,3) \\ & Dst(2,2) & Dst(2,3) \\ sym & & Dst(3,3) \end{bmatrix} \begin{bmatrix} \chi_{xx} \\ \chi_{yy} \\ \chi_{xy} \end{bmatrix}$$

where

χ	plate curvature
m	bending moments

only input of upper triangle of *Dst* required:

```
Dst(:,1:6) = [  Dst(1,1),  Dst(1,2), Dst(1,3), ...
                Dst(2,2),  Dst(2,3), Dst(3,3)  ]
```

Sst symmetric shear stiffness matrix

$$\begin{bmatrix} q_{xz} \\ q_{yz} \end{bmatrix} = \begin{bmatrix} Sst(1,1) & Sst(1,2) \\ sym & Sst(2,2) \end{bmatrix} \begin{bmatrix} \gamma_{xz} \\ \gamma_{yz} \end{bmatrix}$$

where

γ	shear strain
q	shear forces

only input of upper triangle of *Sst* required:

Sst(:,1:3) = [Sst(1,1), Sst(1,2), Sst(2,2)]

alfast symmetric thermal expansion coefficient matrix

$$\begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} alfaxx & 0 & 0 \\ & alfayy & 0 \\ sym & & alfaxy \end{bmatrix} \begin{bmatrix} \Delta t \\ \Delta t \\ \Delta t \end{bmatrix}$$

where

ε	in-plane strain
γ	shear strain
Δt	temperature

only input of upper triangle of *alfast* required:
alfast(:,1:3) = [alfaxx, alfaxx, alfaxy]

Es matrix of shell material laws, automatically assembled from parameter vectors *Ast, Dst, Sst,* and *alfast*:

Es = [Ast, Dst, Sst, alfast]

Each row of *Es* represents a complete shell material law and will be referenced for each shell by the shell element cards *EC_shell* or by the direct shell assignment vector *ishooke.*

geometry
dicke shell thickness

mass
Roh mass density

Shell element cards:

Shell element cards are specified in the matrix *EC_shell(5, :).* Each column of *EC_shell* represents a different element card which comprises assignments to parameter vectors, e.g. material law.

EC_shell(1, :)	element concept
(2, :)	material law (*Es* index no.)
(3, :)	shell thickness (*dicke* index no.)
(4, :)	mass density (*Roh* index no.)
(5, :)	shell reference input coordinate system (*CS*)

Direct shell assignment vectors:

Shell assignment vectors assign parameter vectors to shell elements or comprise the parameters directly for each element. Shell assignment vectors are always of length *nshel.*

shooke	material law (*Es* index no.)
idicke	shell thickness (*dicke* index no.)
irohdis	mass density (*Roh* index no.)

examples for input in file *.m06 :

isotropic material law (e.g: steel)

```
di      = 0.14;       % shell thickness
Emod    = 2.1E11;   % YOUNG’s modulus
nue     = 0.3;      % POISSON ratio

% --- in-plane stiffness  Ast(:,6)=[Ast11 Ast12 Ast13 Ast22 Ast23 Ast33]
      Ast= Emod*di/(1-nue*nue) * [1 nue  0  1  0  (1-nue)/2 ]  ;

% --- bending stiffness Dst(:,6)= [Dst11 Dst12 Dst13 Dst22 Dst23 Dst33]
      Bst= Emod/(1-nue*nue) *di^3/12;
      Dst= Bst* [1 nue  0  1  0  (1-nue)/2 ] ;

% --- shear stiffness Sst(:,3)= [ Sst11 Sst12 Sst22]
      Sst= di*Emod/2/(1+nue) * [1 0 1] ;

% --- thermal expansion coefficients alfast(:,3)=[alfaxx alfayy alfaxy]
      alfast=[0 0 0] ;


% --- NOTE:    the matrix of shell material laws Es is automatically
%     assembled from parameter vectors Ast, Dst, Sst, and alfast :
%     Es = [ Ast, Dst, Sst, alfast ]
%
%     Each row represents a complete shell material law and will be
%     referenced for each shell by the assignment vector ishooke
```

Fig. 6.2.3.1: Input file *.m06, isotropic material law

orthotropic material law (e.g. CFK)

```
    given design and material parameters:

    di    = [ 0.009 ];      % shell thickness
    nue   = [ 0.3];         % POISSON ratio

    %              CFK
    Exx = [ 2.8115e10 ];
    Exy = [ 0.8631e10 ];
    Eyy = Exx;
    Gxy = [ 0.9741e10 ];
    Gxz = [ 0.1072e10 ];
    Gyz = [ 0.1013e10 ];

% --- in-plane stiffness  Ast(:,6)=[Ast11 Ast12 Ast13 Ast22 Ast23 Ast33]

       Ast= [  di( 1) * [   Exx( 1)    Exy( 1)      0    ...
                                        Eyy( 1)      0    ...
                                               Gxy( 1)    ] ];

% --- bending stiffness Dst(:,6)= [Dst11 Dst12 Dst13 Dst22 Dst23 Dst33]

       Dst= [ di( 1)^3/12 * [  Exx( 1)    Exy( 1)      0    ...
                                           Eyy( 1)      0    ...
                                                  Gxy( 1)    ] ];

% --- shear stiffness Sst(:,3)= [ Sst11 Sst12 Sst22]

       Sst= [           di(1) * [Gxz( 1)      0    ....
                                          Gyz( 1)   ] ];

% --- thermal expansion coefficients alfast(:,3)=[alfaxx alfayy alfaxy]
          alfast=[0 0 0] ;


% --- NOTE:   the matrix of shell material laws Es is automatically
%   assembled from parameter vectors Ast, Dst, Sst, and alfast :
%   Es = [ Ast, Dst, Sst, alfast ]
%
%           Each row represents a complete shell material law and will be
%           referenced for each shell by the assignment vector ishooke
```

Fig. 6.2.3.2: Input file *.m06, orthotropic material law

7. Analysis results, listing, MATFEM data basis

7.1 Analysis results

The type of analysis results is determined by the type of analysis performed (parameter *wahl*, specified in *.m01). The amount of listing in the output file **.aus* can be controlled by the parameter *drucke.* The user can also control the run time display by setting *sw_display*. Both parameters are specified in **.m01*.
All mesh information and the analysis results are available from the MATFEM Data Basis (MDB). Refer to chapter 7.2 and 7.3 for more information about the output data file and the MDB.

Analysis features and results available:

dynamic analysis (*wahl* = 2)
- real eigensolutions (natural frequencies, eigenvectors) of undamped structure
- complex eigensolutions (damped eigenfrequencies , complex eigenvectors) of damped structure
- rigid body mass matrix w.r. to origin of global coordinate system
- acceleration frequency domain response due to multi- DOF force input and proportionally or non-proportionally damping
- acceleration, velocity, displacement time domain response due to multi- DOF force input and proportionally or non-proportionally damping
- single mode indicator function (SIF)
- multivariate indicator function (MIF), if more than one excitation force is applied
- substructure coupling using CRAIG-BAMPTON model
- static model order reduction (GUYAN reduction)

static analysis (*wahl* = 3)
- nodal displacements and nodal forces due to
    - concentrated loads at nodal points
    - uniformly distributed loads
    - temperature loads (for shell elements only)
    - prescribed displacement
- static model order reduction (GUYAN reduction)
- stresses at nodal points or GAUS points (shell elements)
- stress resultants at nodal points for beam elements

calculation of cross section parameters (*wahl* = 6)

- cross section parameters

test eigenvector expansion ('blow up') (*wahl* = 7)

- eigenvectors of the overall model derived from a selected subset (e.g. measurement DOF's) of the overall DOF's

non-linear dynamic analysis (*wahl* = 8)

- all results of *wahl* = 2 are available
- non-linear acceleration frequency domain response due to multi- DOF force input and proportionally or non-proportionally damping

*7.2 MATFEM listing file *.aus*

For each analysis run a listing file **.aus* is created which comprises detailed information about

- program flow
- input parameters
- analysis results
- error handling

The amount of output in the listing file **.aus* can be controlled to some extend by the printing switch *drucke* specified in **.m01*. However, the switch *drucke* does not affect the contents of the data files of the MDB.

drucke= 0 *minimum output, e.g.*
displacement
nodal forces
eigenfrequencies, eigenvectors (mode shapes)
...

drucke=1 *extended output, same as drucke= 0 plus e.g.*
coincidence table of element connectivity
boundary conditions
distributed loads
master/slave DOF
grounded spring elements
parameters for dynamic response analysis
element forces of beam elements at start and end node
element forces of shell elements at GAUSS integration points
...

drucke=2 *extended output, same as drucke= 0 and drucke= 1 plus e.g.*
shell element stresses at GAUSS integration points
...

7.3 MATFEM Data Basis (MDB)

The MATFEM Data Basis (MDB) comprises the following MATLAB *.mat files:

general

*.bcs	boundary conditions
*.bqd	beam principal axis parameters
*.guv	Universal Files of mesh contour (ASCII file !)
*.koi	geometry element coincidences: nodes, DOF, shell type, element cards coincidence, ...
*.kos	interface DOF coincidence for Craig/Bampton substructure analysis
*.mtx	system matrices: stiffness, mass
*.pld	basic plot parameters: nodal points, connectivity, ueberschrift, ...
*.prp	properties of shell and beam elements
*.zuo	coincidence table of DOF after introduction of boundary conditions
*.frc	excitation forces
*.ups	prescribed DOF, initial conditions

static analysis

*.sgp	beam and shell stress resultants, shell element stresses at GAUß integration points
*.sig	shell element stresses at nodal points averaged from adjacent elements
*.ugf	nodal displacements and nodal forces
*.mxx	stress resultants intermediate results of 4 node shell elements to be used for the recalculation of shell element stresses (internal use only)

dynamic analysis

*.mod	modal parameters: eigensolution, modal mass ...
*.acc	time domain acceleration response
*.vel	time domain velocity response
*.dis	time domain displacement response
*.rp1	frequency domain mode indicator functions
*.rp2	real part of frequency response
*.rp3	imaginary part of frequency response
*.ctr	control parameters for dynamic response: type of damping, frequency axis parameter,...
*.rst	data of residual structure for CRAIG/BAMPTON substructure coupling

non-linear dynamic analysis

*.phd	damping matrix transformed to physical coordinates
*.nlp	parameters of local non-linear elements
*.rp4	real part of non-linear frequency response (calculated in ‘up’ direction)
*.rp5	imaginary part of non-linear frequency response (calculated in ‘up’ direction)
*.rp6	real part of non-linear frequency response (calculated in ‘down’ direction)
*.rp7	imag. part of non-linear frequency response (calculated in ‘down’ direction)
*.rp8	real part of total non-linear frequency response (calculated in all directions)
*.rp9	imag. part of total non-linear frequency response (calculated in all directions)
*.dyc	data for dynamic condensation for non-linear response calculation

postprocessor

*.adr	graphic handles

Detailed information about the contents of the files is given in **Appendix B.**

In each file of the MDB the actual MATFEM runtime identifier *MF_RUNID* is stored. This identifier is a global string variable which is set within each analysis run and consists of the MATFEM version identifier, the common file name, actual date and time, e.g.

```
MF_RUNID = '99.01.b018__demo01__26-Apr-0__11:8:16.21'
```

The MATFEM runtime identifier is also listed in the **.aus* file. This identifier helps to assign analysis results of different analysis runs.

Retrieving data from the MATFEM Data Basis

There are two different ways to load data from the MATFEM data basis

a) using the MATLAB command *load*,

```
e.g.:  load  *.mtx  -mat,
```

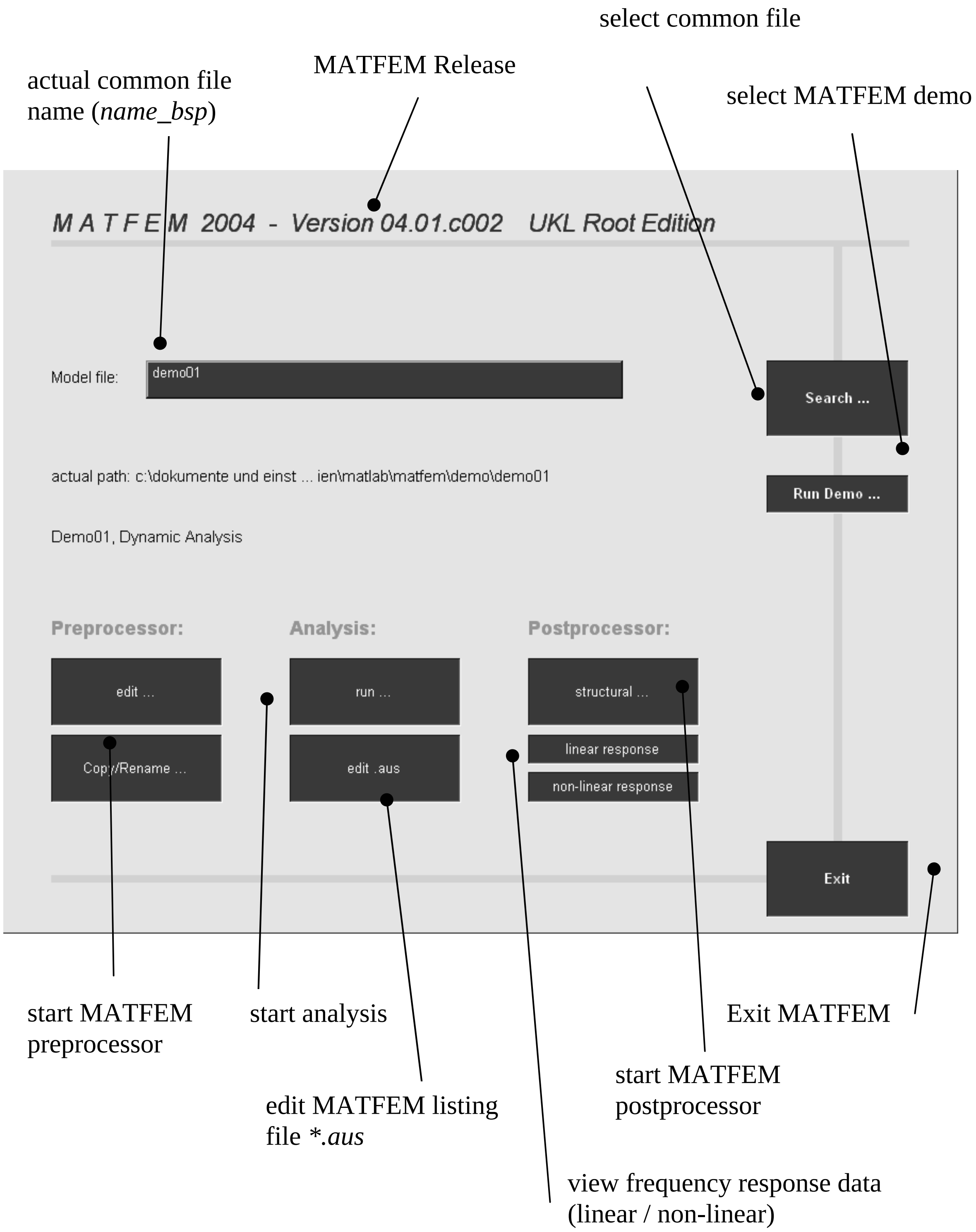
b) using the MATFEM MDB data retrieve functions,

```
e.g.: [ K, Ke, Kekoin, Kes, Keskoin, Fei, Ffpg, koin, va, vb, EKaa, mkel, ...
        M, Me, Mekoin, Mes, Meskoin, EMaa, Mstarr, MF_RUNID_sav  ]= ...
        mf_r_mtx( name_bsp, FID_AUS, warn, MF_RUNID_ref)
```

There is a data retrieve function for each file of the MDB. Press *help mf_r_** , e.g. *help mf_r_mtx*, for detailed information about the syntax.

8. MATFEM GUI

MATFEM main window



M A T F E M 2004 - Version 04.01.c002 UKL Root Edition

Model file: demo01

actual path: c:\dokumente und einst ... ien\matlab\matfem\demo\demo01

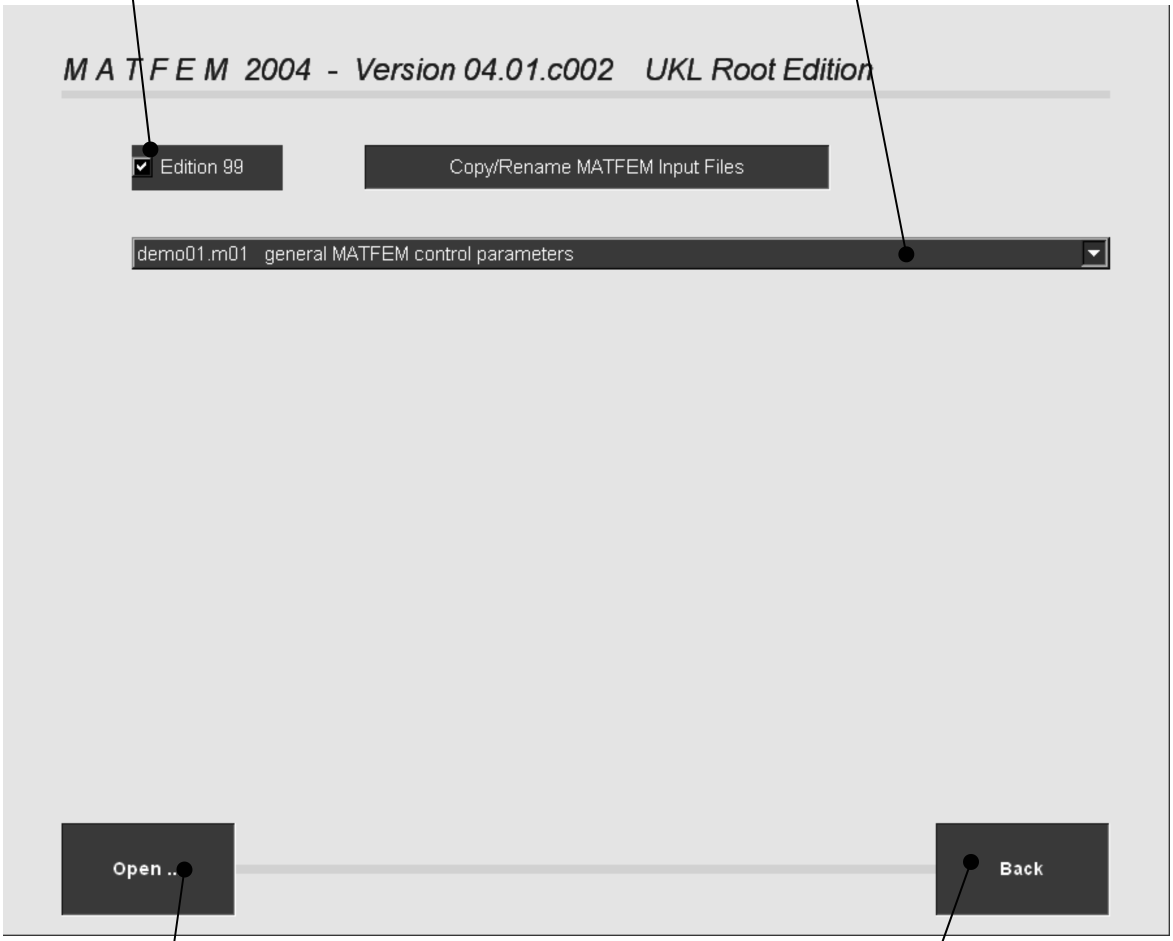
Demo01, Dynamic Analysis

Fig. 8.1: MATFEM Main window

MATFEM Preprocessor

select input file structure:
edition2K: 20 input file (default)
or previous MATFEM 16 input file structure

select input file (pull down menu)


M A T F E M 2004 - Version 04.01.c002 UKL Root Edition
☑ Edition 99
Copy/Rename MATFEM Input Files
demo01.m01 general MATFEM control parameters
Open ..
Back

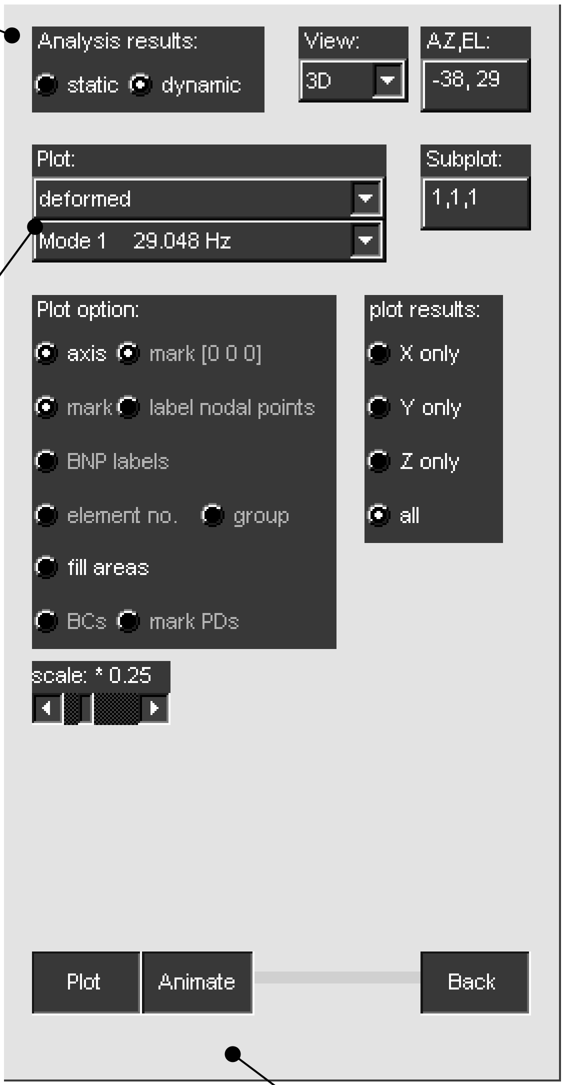
open selected input file

Exit MATFEM preprocessor

Fig. 8.2: MATFEM Preprocessor

MATFEM Postprocessor

select analysis result type to plot



Analysis results: ○ static ● dynamic

View: 3D

AZ,EL: -38, 29

Plot: deformed

Mode 1 29.048 Hz

Subplot: 1,1,1

Plot option:

● axis ● mark [0 0 0]

● mark ○ label nodal points

○ BNP labels

○ element no. ○ group

○ fill areas

○ BCs ○ mark PDs

plot results:

○ X only

○ Y only

○ Z only

● all

scale: * 0.25

1

plot scaling

plot according actual plot settings

Exit postprocessor

animated plot

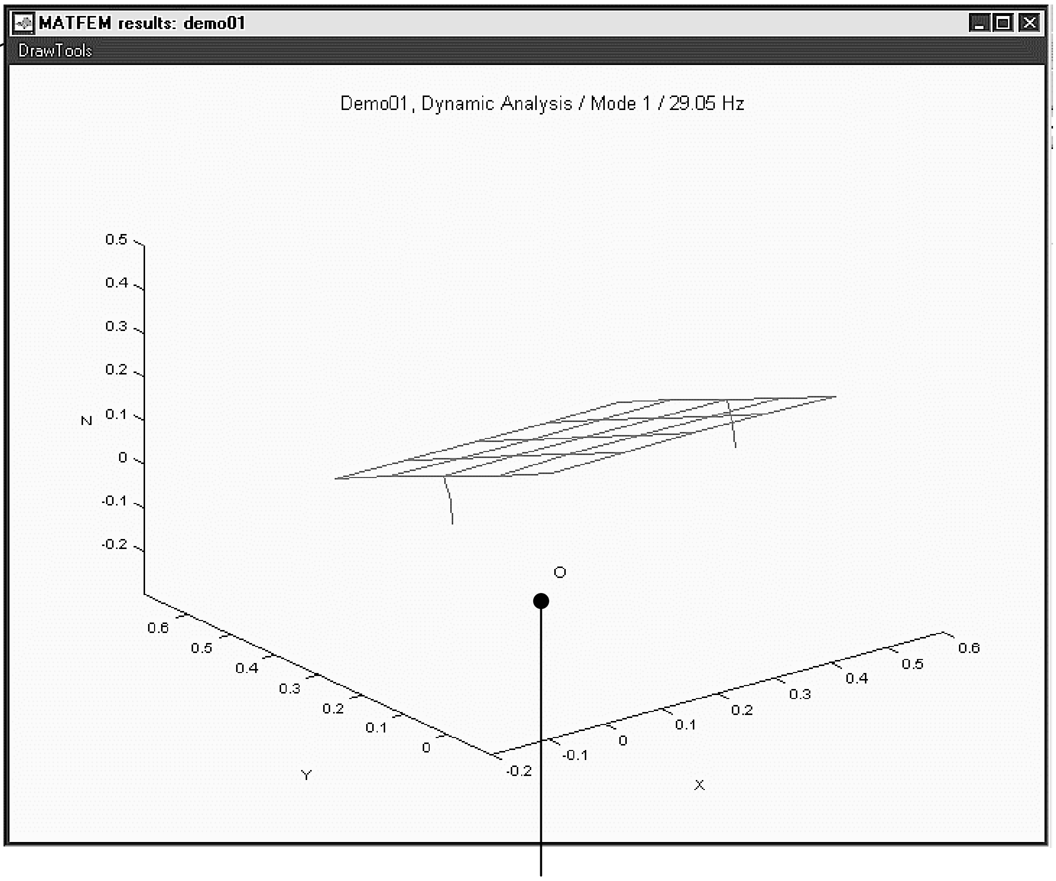
1)

```
undeformed
cross section
arrow plot of Ux, Uy, Uz
deformed/undeformed
deformed
multi-view, deformed
contour
beam stress resultants
arrow plot of Uxx,Uyy,Uzz
arrow plot of Fx, Fy, Fz
arrow plot of Fxx,Fyy,Fzz
shell stress resultants at Gauss IP
shell stresses at Gauss IP
averaged stress resultants at NP
averaged stresses at Gauss NP
```

Fig. 8.3: MATFEM Postprocessor

MATFEM Results Plot (deformed)

1



origin of **global** coordinate system

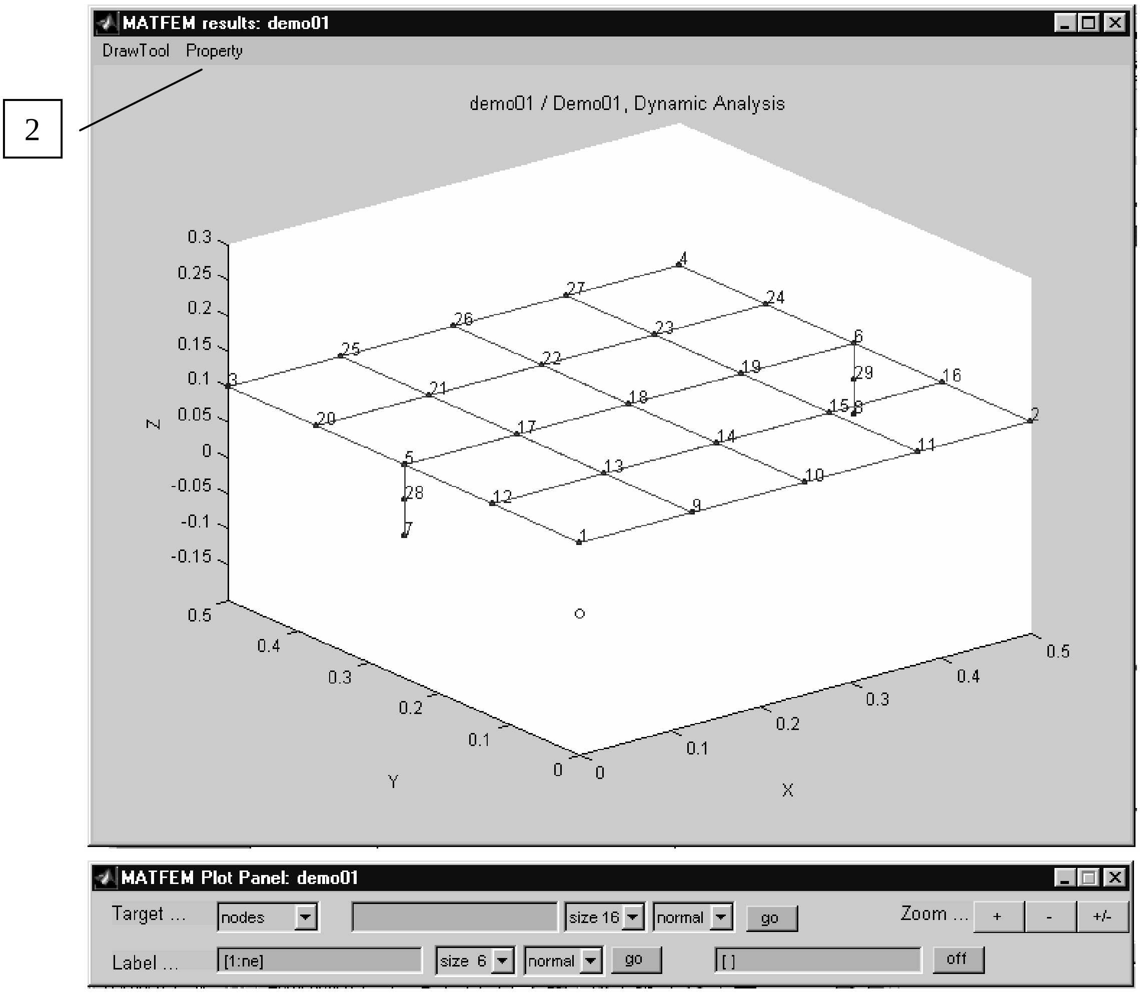
1) drawtools

```
Rotate
Zoom

Axis
  square (3D)
  auto
  on/off

Restore

Menubar

b/w plot to printer
b/w plot to paste
b/w plot bmp file

Keep figure

Print to Printer
Print to PS
Print to HPGL
Print to Paste

toggle xscale
toggle yscale
toggle zscale
```

MATLAB figure menubar

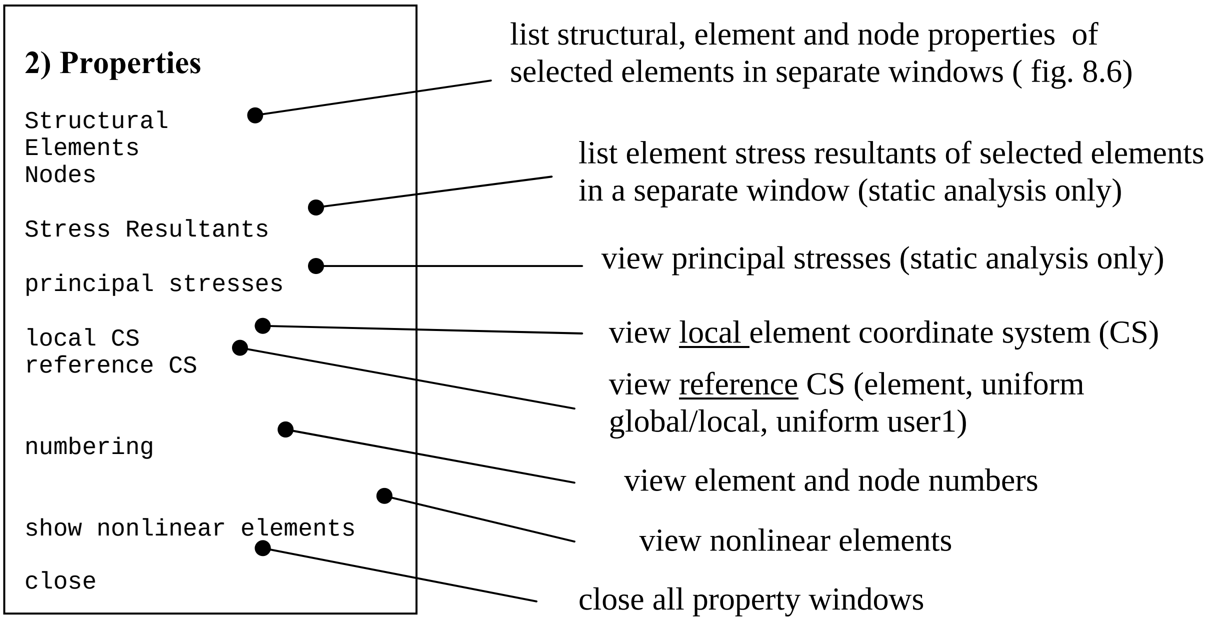
Fig. 8.4: MATFEM results plot (deformed)

MATFEM Results Plot (undeformed), MATFEM Plot Panel



Target ... nodes size 16 normal go

Label ... [1:ne] size 6 normal go [ ] off



2) Properties

Structural
Elements
Nodes — list structural, element and node properties of selected elements in separate windows (fig. 8.6)

Stress Resultants — list element stress resultants of selected elements in a separate window (static analysis only)

principal stresses — view principal stresses (static analysis only)

local CS — view local element coordinate system (CS)

reference CS — view reference CS (element, uniform global/local, uniform user1)

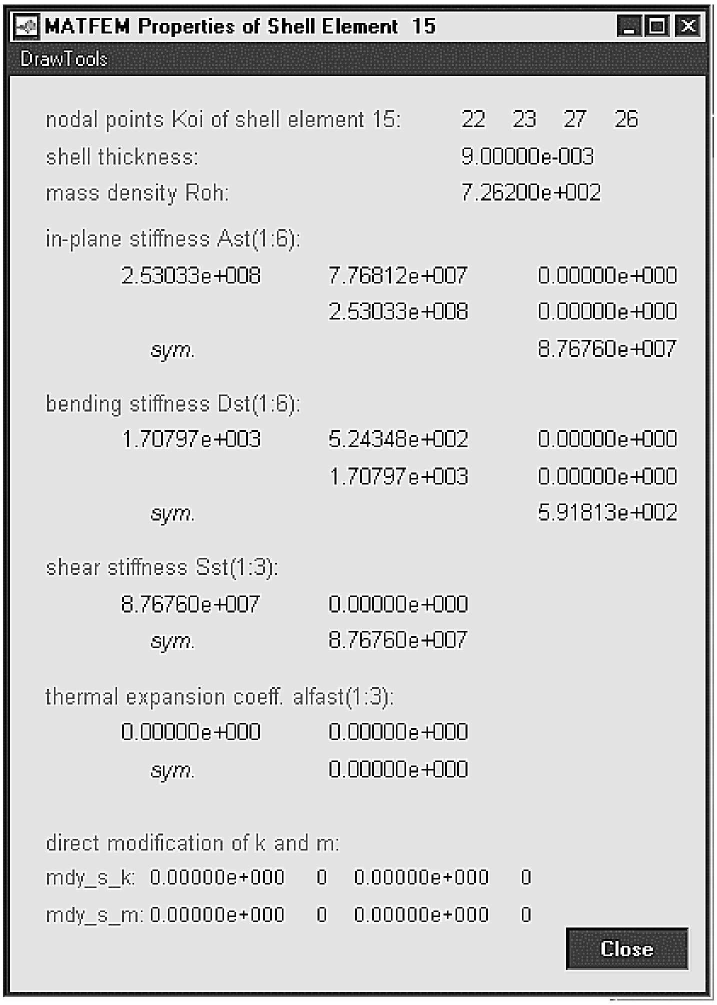
numbering — view element and node numbers

show nonlinear elements — view nonlinear elements

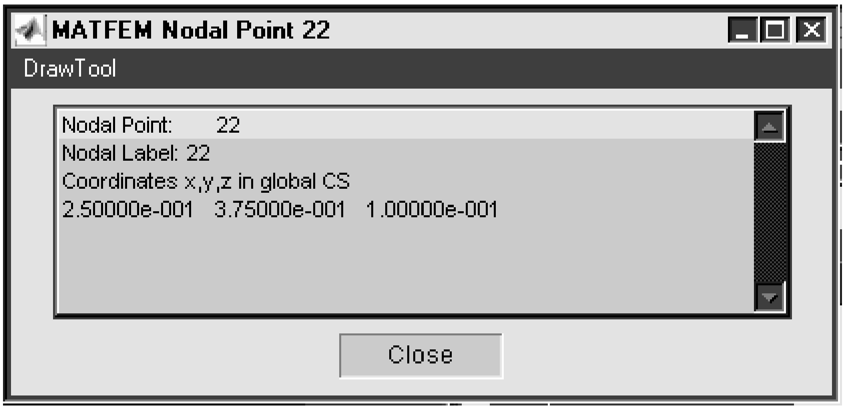
close — close all property windows

Fig. 8.5: MATFEM results plot (undeformed), MATFEM Plot Panel

MATFEM Element Property Window

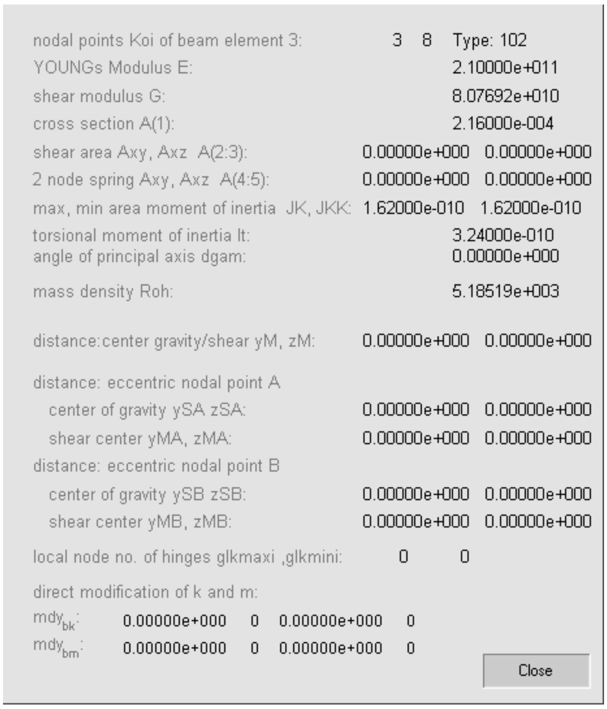


```
nodal points Koi of shell element 15:     22   23   27   26
shell thickness:                          9.00000e-003
mass density Roh:                         7.26200e+002

in-plane stiffness Ast(1:6):
        2.53033e+008    7.76812e+007    0.00000e+000
                        2.53033e+008    0.00000e+000
          sym.                          8.76760e+007

bending stiffness Dst(1:6):
        1.70797e+003    5.24348e+002    0.00000e+000
                        1.70797e+003    0.00000e+000
          sym.                          5.91813e+002

shear stiffness Sst(1:3):
        8.76760e+007    0.00000e+000
          sym.          8.76760e+007

thermal expansion coeff. alfast(1:3):
        0.00000e+000    0.00000e+000
          sym.          0.00000e+000

direct modification of k and m:
mdy_s_k:  0.00000e+000   0   0.00000e+000   0
mdy_s_m: 0.00000e+000   0   0.00000e+000   0
```



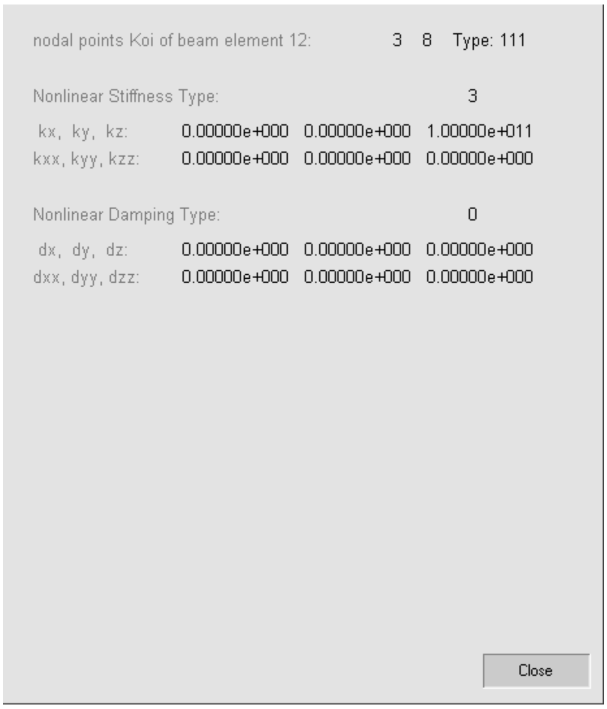
```
Nodal Point:      22
Nodal Label: 22
Coordinates x,y,z in global CS
2.50000e-001   3.75000e-001   1.00000e-001
```

Fig. 8.6: MATFEM property window: shell element, nodal point



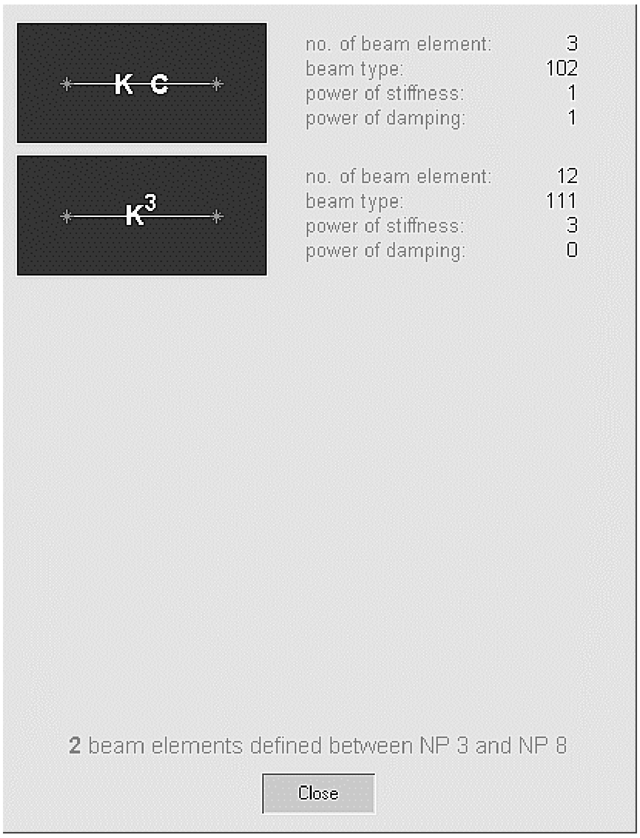
```
nodal points Koi of beam element 3:              3   8   Type: 102
YOUNGs Modulus E:                                        2.10000e+011
shear modulus G:                                         8.07692e+010
cross section A(1):                                      2.16000e-004
shear area Axy, Axz  A(2:3):              0.00000e+000   0.00000e+000
2 node spring Axy, Axz  A(4:5):           0.00000e+000   0.00000e+000
max, min area moment of inertia  JK, JKK: 1.62000e-010   1.62000e-010
torsional moment of inertia It:                          3.24000e-010
angle of principal axis dgam:                            0.00000e+000
mass density Roh:                                        5.18519e+003

distance:center gravity/shear yM, zM:     0.00000e+000   0.00000e+000

distance: eccentric nodal point A
   center of gravity ySA zSA:             0.00000e+000   0.00000e+000
   shear center yMA, zMA:                 0.00000e+000   0.00000e+000
distance: eccentric nodal point B
   center of gravity ySB zSB:             0.00000e+000   0.00000e+000
   shear center yMB, zMB:                 0.00000e+000   0.00000e+000

local node no. of hinges glkmaxi ,glkmini:    0      0

direct modification of k and m:
mdy_bk:    0.00000e+000   0   0.00000e+000   0
mdy_bm:    0.00000e+000   0   0.00000e+000   0
```

linear element



```
nodal points Koi of beam element 12:        3   8   Type: 111

Nonlinear Stiffness Type:                            3
 kx,  ky,  kz:      0.00000e+000  0.00000e+000  1.00000e+011
kxx, kyy, kzz:      0.00000e+000  0.00000e+000  0.00000e+000

Nonlinear Damping Type:                              0
 dx,  dy,  dz:      0.00000e+000  0.00000e+000  0.00000e+000
dxx, dyy, dzz:      0.00000e+000  0.00000e+000  0.00000e+000
```

non-linear element



```
no. of beam element:      3
beam type:              102
power of stiffness:       1
power of damping:         1

no. of beam element:     12
beam type:              111
power of stiffness:       3
power of damping:         0
```

2 beam elements defined between NP 3 and NP 8

Fig. 8.7: MATFEM property windows: beam elements

9. Solutions

9.1 Dynamic linear response analysis in the frequency domain

Using the dynamic response analysis feature in MATFEM it is possible to calculate the linear acceleration frequency response $\ddot{\mathbf{U}}$

$$\ddot{\mathbf{U}}(\omega) = -\,\omega^2\mathbf{U}(\omega) \qquad (9.1.1)$$

of selective DOF’s due to a harmonic excitation

$$\mathbf{F}(t) = \mathbf{F}(j\omega)e^{j\omega t} \qquad (9.1.2)$$

with one or several nodal forces (excitation forces), acc. the basic linear equation of motion

$$(-\omega^2\mathbf{M} + j\omega\mathbf{D} + \mathbf{K})\,\mathbf{U}(j\omega) = \ \mathbf{F}(j\omega) \qquad (9.1.3)$$

where

$\mathbf{M},\mathbf{D},\mathbf{K}$	physical system matrices
ω	excitation frequency
$j=\sqrt{-1}$	imaginary unit

The selected DOF's are referred as MDOF's (measurement degrees of freedom). The amplitudes of the excitation forces may be constant and real valued for all the frequency points (spectral lines) included in the analysis or the user may specify general non-constant complex excitation forces. Within the MATFEM analysis run the results of the dynamic analysis are presented in a plot of the real and imaginary parts of the acceleration frequency response. All MDOF’s are overlaid onto the same graph. The user can view the frequency response in more detail by using the *response* postprocessor, fig. 8.1.
In addition, the Single Mode Indicator Function (SIF, Phasenresonanzkriterium) is calculated from the response.

$$\mathrm{SIF}(i) = \frac{\sum_j \ddot{U}^{re}_{j,i}}{\sum_j \left(\ddot{U}^{re}_{j,i} + \ddot{U}^{im}_{j,i}\right)} \qquad (9.1.4)$$

where

$\ddot{U}^{re}_{j,i},\quad \ddot{U}^{im}_{j,i}$	real and imaginary part of acceleration frequency response for the j-th MDOF at excitation frequency ω_i

If more than one excitation forces are applied, the Multivariate Mode Indicator Function (MIF) [BRILLHART 1] is also calculated. Therefore the eigenvalue problem for each eigenfrequency is solved:

$$\left[-\left(H_{re}^{T}H_{re}+H_{im}^{T}H_{im}\right)\lambda_{1}\left(\omega_{0i}\right)+H_{re}^{T}H_{re}\right]\alpha\left(\omega_{0i}\right)=0 \qquad (9.1.5)$$

where

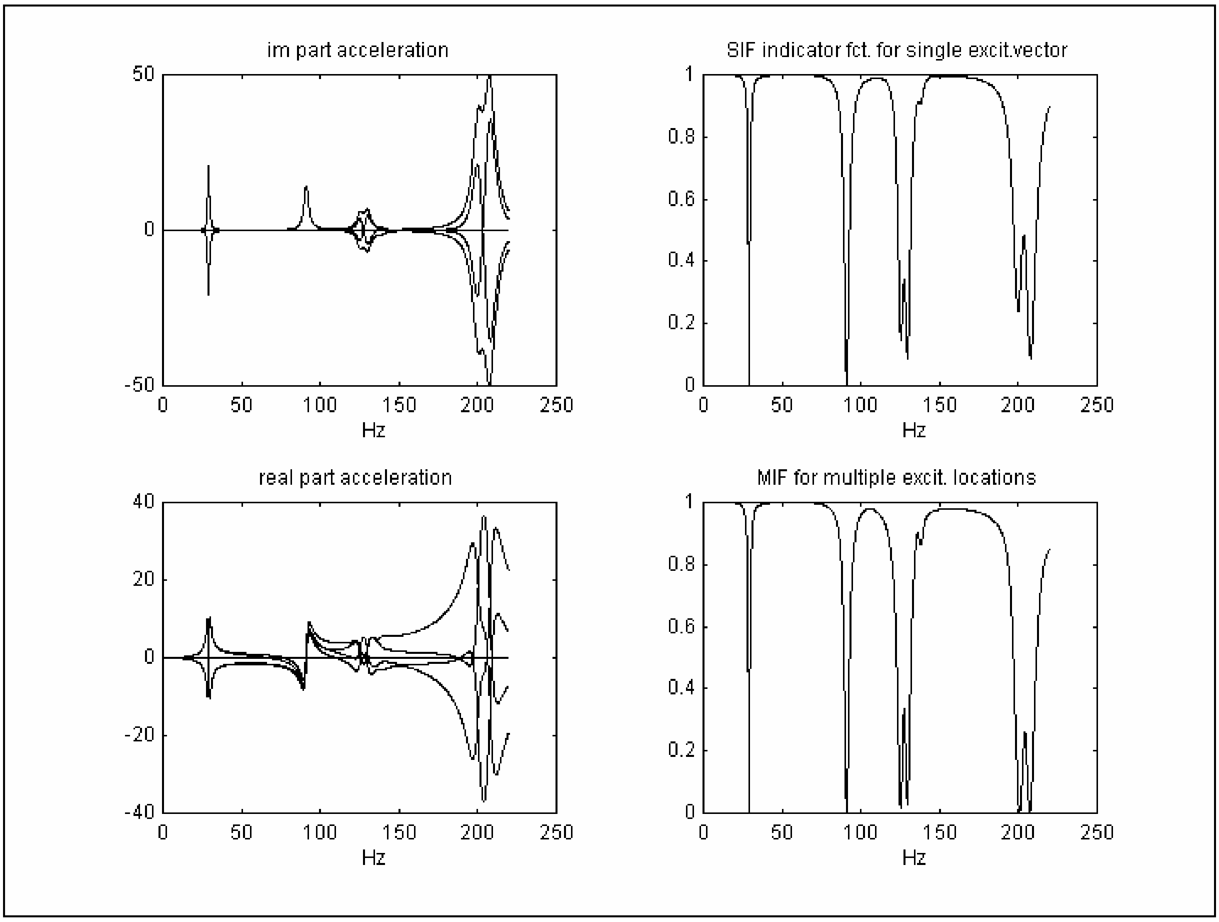
ω_{0i}	eigenfrequency (i = 1...m)
$\lambda_1\left(\omega_{0i}\right)$	smallest eigenvalue = multivariate mode indicator function value at $\omega = \omega_{0i}$
$\mathbf{H} \in R^{n,ne}$	FRF matrix (re = real-, im = imaginary part)
$\alpha(\omega_{0i}) \in R^{ne,1}$	eigenvector = appropriate force vector

This multivariate mode indicator function $\lambda_1\left(\omega\right)$ shows explicitly whether all desired mode shapes can be excited or not. Furthermore the calculation of the multivariate mode indicator function provides the determination of optimum excitation configurations (amplitudes) w.r. to the applied excitation force locations to excite the respective modes shapes best. This excitation force configurations are referred as the optimum excitation vector *foptr* and is listed in the MATFEM listing file **.aus*. If the user would apply this optimum excitation vector in the response calculation instead of the originally specified excitation vector the overall response at an eigenfrequency ω_{0i} would be dominated by a almost pure response of the respective mode shape and therefore fulfill the phase resonance criterion (SIF(ω_{0i}) = 1).

The response analysis parameters and results are stored in the MDB:

*.ctr	response control parameters
*.frc	excitation forces, DOF coincidence
*.rp1	response/reference MDOF identifier, Single Mode Indicator Function, Multivariate Indicator function
*.rp2	real part of frequency response
*.rp3	imaginary part of frequency response

NOTE:
The <u>**acceleration**</u> frequency response is calculated in MATFEM.



(4 MDOF's, 7 modes, 2 constant real valued excitation force, excitation frequency range 0 ... 220 Hz, 1000 excitation frequency points with eigenfrequencies included)

Fig. 9.1.1: Acceleration frequency response, Single Mode Indicator Function (SIF) and Multivariate Mode Indicator function (MIF) of 4 MDOF's of a given structure (demo01).

The parameters of a dynamic frequency domain response analysis are specified in the input file **.m14*

resp_domain response domain:
= 0 no response calculation (suppress response calculation)
= 1 frequency domain
= 2 time domain (see chapter. 9.4)

Specification of modal parameters used for the response calculation:

name_modalresp name of a file with already existing modal data
if *name_modalresp* = [] the actually calculated modal data are used

nresp total number of modes to be included in the response calculation.
nresp = 0 suppresses response calculation
xeffno mode numbers to be included in the response calculation.
The user must take care that xeffno is less or equal to the number of modes, which will be calculated from the eigenvalue problem.

Specification of the damping matrix:

The damping matrix can be specified in physical or modal coordinates. The damping matrices is assumed to be symmetric. The user has to specify one of the following matrices *C, xsi, xsid, prop_d*

C upper triangle of physical damping matrix C (na,na), where na = number of physical DOF’s (line-by-line input)
xsi upper triangle of non-diagonal modal damping matrix *xsi*(*nresp*,*nresp*)
xsid diagonal of modal damping matrix *xsid* (*nresp*). Input in fractions of critical damping
prop_d alpha and beta –factors as multipliers for the stiffness and mass matrix to generate a proportional damping matrix, *prop_d(1,2)*

Specification of dynamic excitation forces:

nef total number of nodal excitation forces
knr nodal points at which the excitation nodal forces apply, *knr*(1, *nef*)
ril DOF direction of *knr*
F - constant excitation forces w.r. to frequency
real valued force amplitude *F*(1, *nef)*
example: Two excitation forces 1N, 1 N:

```
F= [ 1 1 ];
```

- non-constant excitation forces w.r. to frequency
complex valued force amplitude *F(:, 1+ nef)*. In the first column the excitation frequency axis is specified. In columns 2 .. 1+ *nef* the *nef* complex excitation forces w.r. to the frequency axis of column 1 are specified
example: Two linearly increasing excitation forces: 1N, 1N at 2 Hz and 5N and 3 N at 50 Hz

```
F= [    2  1  1
            50  5  3 ];
```

example: One linearly increasing complex excitation forces 1N at 2 Hz and 5N at 50 Hz

```
    j = sqrt( -1)
F= [        2     0+ j* 1
            50     0+ j* 5 ];
```

Note: The force vector is linearly interpolated to the frequency response axis

Specification of frequency response axis:

stepno	number of excitation frequencies to be generated within lower and upper frequency bounds
flow	lower excitation frequency bound [Hz]
fup	upper excitation frequency bound [Hz]
addf	numbers of eigenfrequencies to be included as excitation frequencies

Specification of DOF's for the response calculation (MDOF):

knresp	node numbers for frequency response calculation
riresp	DOF directions of *knresp*

Mode indicator functions, plotting:

imif	switch for calculation of Multivariate Indicator Function (MIF)
plotresp	switch for plotting the frequency response

9.2 Substructure coupling using CRAIG/BAMPTON method

In dynamic analysis of complex structures with a large number of DOF's the CRAIG/BAMPTON (CB) method is a well known method to drastically reduce the overall number of DOF's. Therefore the overall structure is divided into two or more parts:

- main structure
- residual structure(s)

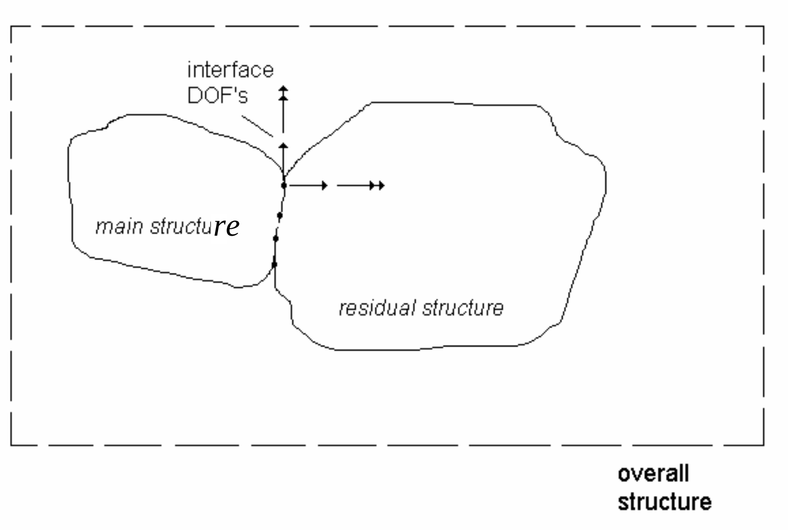


Fig. 9.2.1: Main and one residual structure

The main structure is totally represented in physical coordinates. A residual structure is represented by a number of interface DOF's (physical coordinates) which are shared with the main structure and it is also possible to define some additional physical coordinates which are not coupled to the main structure. In addition a number of modal DOF's are used which are related to the residual structure being <u>fixed</u> at the master DOF's. These modal DOF's represent the dynamic stiffness of the residual structure. The $(n_a, 1)$ DOF's of the residual structure $\mathbf{y}_a$ are transformed to the $(n_m, 1)$ displacements of the master DOF's $\mathbf{y}_m$ and to the $(n_q, 1)$ modal displacements $\mathbf{q}$ related to the residual structure being <u>fixed</u> at the master DOF's.

$$\mathbf{y}_a = \mathbf{T}\,\mathbf{y}_m + \mathbf{\Phi}_a\,\mathbf{q} \tag{9.2-1}$$

where

n_a	number of DOF's of restrained residual structure
n_m	number of master DOF's
n_I	number of interface DOF's ($\le n_m$)
n_p	$= n_m - n_I$ = physical DOF's
$n_q \le n_a + n_m$	no. of retained modes
index I	related to interface DOF's
index a	related to residual structure, <u>fixed</u> at master DOF's
index m	related to master DOF's of residual structure
index p	related to physical DOF's of residual structure
$\mathbf{T} = -\mathbf{K}_{aa}^{-1}\,\mathbf{K}_{am}$	(n_a, n_m) static transformation matrix (Guyan matrix)

$\Phi_a = [\ \varphi_1 \ldots \varphi_{nq}]$ (n_a, n_q) modal matrix of the residual structure <u>fixed</u> at the master DOF's, resulting from the solution of the eigenvalue problem

$$(-\omega_a^2\, \mathbf{M}_{aa} + \mathbf{K}_{aa})\varphi_a = \mathbf{0} \tag{9.2-2}$$

The eigenvalue problem of the unconstrained residual structure with <u>free</u> interfaces and partitioned with respect to the $\mathbf{y}_a$ and $\mathbf{y}_m$ displacements can be expressed by the $\mathbf{y}_m$ and $\mathbf{q}$ displacement coordinates using the transformation acc. eq. (9.2-1) together with the identity $\mathbf{y}_m = \mathbf{y_m}$ which leads to

$$(-\omega_R^2\, \mathbf{M}_R + \mathbf{K}_R)\,\varphi_R = \mathbf{0} \tag{9.2-3}$$

where

index R related to residual structure, <u>free</u> at interface DOF's,

index q related modal DOF's of residual structure, <u>free</u> at interface DOF's,

with the $(n_m + n_q\,,\, n_m + n_q\,)$ Craig/Bampton matrices

$$\mathbf{K}_R = \begin{bmatrix} \mathbf{K}_m & 0 \\ 0 & \gamma_a \end{bmatrix}_R \tag{9.2-4}$$

$$\mathbf{M}_R = \begin{bmatrix} \mathbf{M}_m & \mathbf{M}_{mq} \\ \mathbf{M}_{qm} & \mu_a \end{bmatrix}_R \tag{9.2-5}$$

related to the displacement vector

$$\varphi_R^{\ T} = \left[\ \mathbf{y}_m^{\ T}\ \ \mathbf{q}^T \right] \tag{9.2-6}$$

$\mathbf{K_m}$ and $\mathbf{M}_m$ denote the residual matrices of the structure statically condensed to the master DOF's $\mathbf{y}_m$ by

$$\mathbf{K}_m = \mathbf{T}^T\, \mathbf{K}_{am} + \mathbf{K}_{mm} \tag{9.2-7}$$

$$\mathbf{M}_m = \mathbf{M}_{mm} + \mathbf{T}^T\, \mathbf{M}_{am} + \mathbf{M}_{ma}\, \mathbf{T} + \mathbf{T}^T\, \mathbf{M}_{aa}\, \mathbf{T} \tag{9.2-8}$$

The matrices γ_a and μ_a in eq. 9.2-4,5 represent the modal stiffnesses, the modal masses and the eigenfrequencies of the residual structure <u>fixed</u> at its master DOF's and calculated from eq. (9.2-2)

$$\gamma_a = \ \text{diag}(\ \gamma_i = \mu_i\, \omega_{ai}^2\ ) = \ \Phi_a^{\ T}\, \mathbf{K}_{aa}\, \Phi_a \tag{9.2-9}$$

$$\mu_a = \ \text{diag}(\ \mu_i\,) \ = \ \Phi_a^{\ T}\, \mathbf{M}_{aa}\, \Phi_a \tag{9.2-10}$$

where

ω_{ai} (i= 1,..n_q) retained circular eigenfrequencies of residual structure <u>fixed</u> at master DOF's

The (n_m ,n_q) matrix

$$\mathbf{M}_{mq} = (\mathbf{M}_{ma} + \mathbf{T}^T \mathbf{M}_{aa})\, \mathbf{\Phi_a} \tag{9.2-11}$$

represents the participation matrix which couples the modal and the master DOF’s.

After partitioning the main structure matrices $\mathbf{K}_M$ and $\mathbf{M}_M$ with respect to the (n_s, 1) displacement vector y_s and the (n_I, 1) interface displacement vector $\mathbf{y}_I$ the main structure and the residual structure can be assembled as usual by coupling the submatrices at the interface DOF’s thus yielding the following eigenequation of motion for the coupled system:

$$\left\{ \begin{bmatrix} K_{M,ss} & K_{M,sI} & 0 & 0 \\ & K_{M,II+}K_{R,I} & 0 & 0 \\ & 0 & K_{R,p} & 0 \\ & & & \gamma_R \end{bmatrix} - \omega^2 \begin{bmatrix} M_{M,ss} & M_{M,sI} & 0 & 0 \\ & M_{M,II}+M_{R,I} & 0 & M_{R,Iq} \\ & & M_{R,p} & M_{R,pq} \\ & & & \mu_R \end{bmatrix} \right\} \begin{bmatrix} y_s \\ y_I \\ y_p \\ q_R \end{bmatrix} = 0 \tag{9.2-12}$$

where

index M related to main structure
index R related to residual structure

The matrices γ_R and μ_R in eq. 9.2-12 represent the modal stiffnesses, the modal masses and the eigenfrequencies of the residual structure free at its interface DOF’s

$$\gamma_R = \text{diag}(\gamma_i = \mu_{R,i}\, \omega_{R,i}^2), \tag{9.2-13}$$

$$\mu_R = \text{diag}(\mu_{R,i}). \tag{9.2-14}$$

This equation represents the classical CRAIG/BAMPTON formulation frequently used for substructure coupling analysis. Its accuracy is governed by the number n_q of the modes of the residual structure which are retained in the modal matrix Φ_a in eq. 9.2-1. This formulation shows that the main structure is totally described by its physical quantities in the stiffness and mass matrix whereas the residual structure is described by its modal parameters. In eq. 9.2-12 only one substructure is coupled, but the same technique can be used to couple many (different) substructures to one main structure.

To couple substructures using the Craig/Bampton method two or more (dependent on how many substructures are defined) separate MATFEM analysis runs must be performed:

step 1: analysis of the each residual structure with **free** interfaces

*.m09 ***boundary conditions***
no boundary conditions must be specified.
i.e.: *Kux* = [ ], *Kuy* = [ ] , *Kuz* = [ ] , *Kuxx* = [ ] , *Kuyy* = [ ] , *Kuzz* = [ ]

*.m10 ***static condensation***
specification of node numbers with master DOF’s is required
i.e.: *Hux, Huy, Huz, Huxx, Huyy, Huzz*

numbers of modal DOF's to be saved on file *.rst have to be defined:

e.g.: *qrest = [1 2 4 5 6]*

specification of interface DOF's of substructure to link Craig/Bampton substructure to main structure is required:
i.e.: *Kosx, Kosy, Kosz, Kosxx, Kosyy, Koszz*

NOTE: interface DOF's must be equal or less than master DOF's defined

no file name *name_rst* must be specified:
name_rst = [];

no effective numbers of modal DOF's of the main structure *qeff* must be specified:
qeff = 0

no interface DOF's of main structure to link Craig/Bampton substructures to main structure must be specified:
e.g: *Kox* = [ ], *Koy* = [ ] , *Koz* = [ ] , *Koxx* = [ ] , *Koyy* = [ ] , *Kozz* = [ ]

A *.rst data file is created automatically. This file contains all needed information about the residual structure and will be used in **step 2**.

step 2: analysis of the coupled structure

*.m09 ***boundary conditions***
boundary conditions may be specified as usual.

*.m10 ***static condensation***
no specification of:

- *Hux, Huy, Huz, Huxx, Huyy, Huzz*
- *qrest*
- *Kosx, Kosy, Kosz, Kosxx, Kosyy, Koszz*

The vector of the numbers of modal DOF's of the residual structures *qeff* to be used for the CRAIG/BAMPTON matrices must be specified:
e.g.: *qeff = [1 2 3 NaN 2 3 4 NaN 1 2 5 NaN]* (in case of 3 substructures)
qeff can be empty; then only physical DOF's are coupled and added to main structure. Values of *qeff* must have been defined in *qrest* of the corresponding substructure in **step 1**.

The file name *name_rst* must be specified for each substructure. Each row of *name_rst* contains the name of one substructure. All names need to have the same length. The corresponding *.rst-files were generated in **step 1**. They contain the needed input of the residual structures to assemble the CRAIG/BAMPTON matrices.

specification of interface DOF's of main structure to link Craig/Bampton substructures to main structure:
i.e.: *Kox, Koy, Koz, Koxx, Koyy, Kozz*

An example for a substructure analysis with **one** residual structure using the CRAIG/ BAMPTON method is given in the demonstration directory *demo09*. See also *demo09* in Appendix D for more information. An example with **two** residual structures is given in *demo17*. See also *demo17* in Appendix D for more information.

NOTE:

The residual structure should always be specified using the same global coordinate system as the main structure. However, if the user prefers to use a local coordinate system for the residual structure he must take high care that the local and global coordinate system are specified using the same axis orientation.

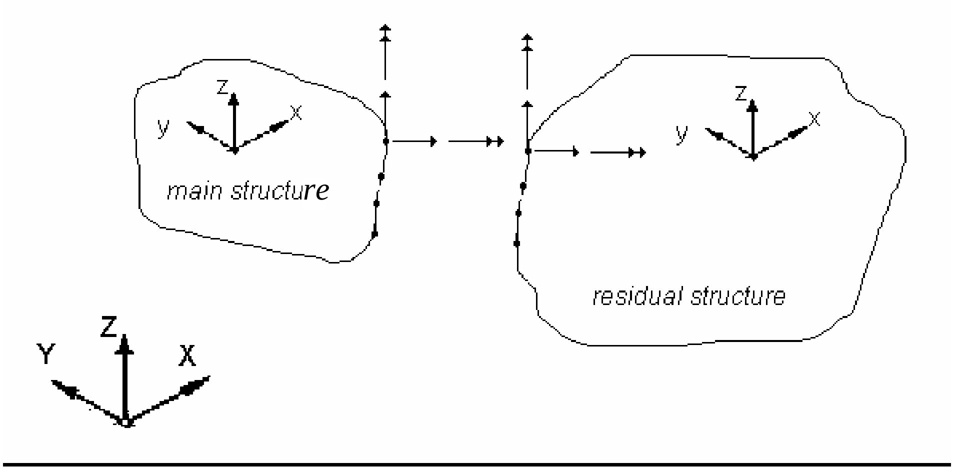


Fig 9.2.2: Axis orientation of **local** coordinate systems for main and residual structure **must** coincide

NOTE:

Using the CRAIG/BAMPTON method, **ONLY** ***eigenfrequencies and eigenvectors*** of the reduced system can be calculated. **NO** ***response*** can be calculated for the reduced system.

9.3 Calculation of cross section parameters

In MATFEM it is possible to calculate the parameters of a given cross section. They are derived from the rigid mass matrix properties in the special case that the plane thickness and the mass density of all elements describing the cross section are set to unity. In MATFEM the rigid mass matrix is calculated from the expression

$$\mathbf{M}_{\text{rigid}} = \mathbf{X}_{\text{r}}^{\text{T}}\mathbf{M}\mathbf{X}_{\text{r}} \tag{9.3.15}$$

where $\mathbf{X}_{\text{r}}$ is the vector of rigid body modes, ref. [LINK 1].

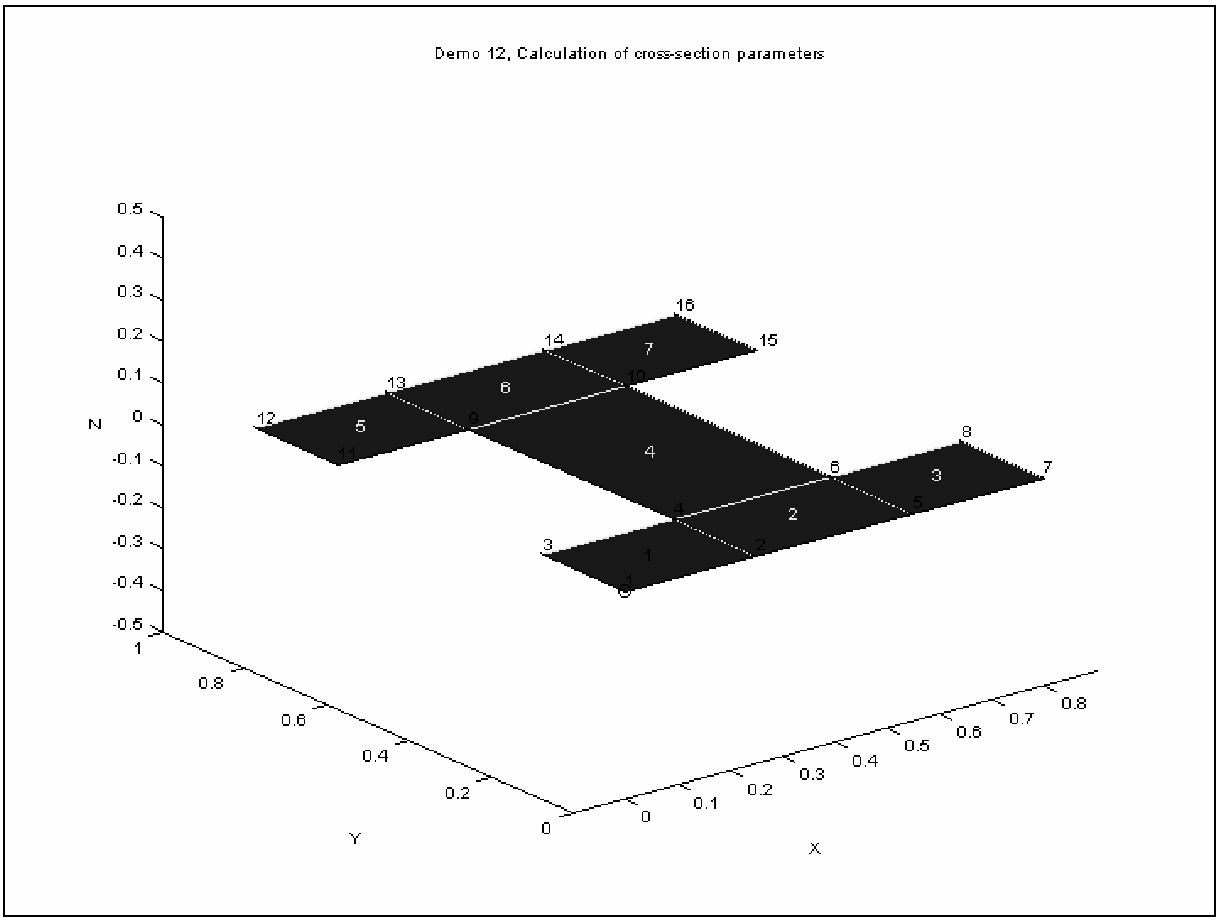


Fig. 9.3.1: Calculation of cross section parameters, (demo12)

The user has to specify the following parameters

*.m01	*wahl* = 6 must be set
*.m04	specifying the cross section by the geometry elements *fla9,flarot,tria,triarot* **only** (no beam elements supported). Input in x,y-coordinates **only** (plane surface)

An example for this type of analysis is given in the demonstration directory *demo12*. See also *demo12* in Appendix D for more information. It is strongly recommended that the user should use the demo12 input files as templates for his application. In this case the user only has to modify the input file *.m04 to represent the actual cross-section.

After the analysis has terminated the user can view the cross section and the corresponding parameters using the postprocessor. The cross section parameters are also listed in the *.aus file:

A	cross section area
xs	x-coordinate of center of gravity
ys	y-coordinate of center of gravity
Ixx	moment of inertia w.r. to x-axis of the origin coordinate system
Iyy	moment of inertia w.r. to y-axis of the origin coordinate system
Ixy	centrifugal moment w.r. to the origin coordinate system
Ixxs	moment of inertia w.r. to x-axis of the center of gravity coordinate system
Iyys	moment of inertia w.r. to y-axis of the center of gravity coordinate system
Ixys	centrifugal moment w.r. to the center of gravity coordinate system
I1	main moment of inertia (max. value)
I2	main moment of inertia (min. value)
phi	main axis angle (in degrees) (measured anti-clockwise w.r. to x-axis)

9.4 Dynamic response analysis in the time domain

The dynamic time domain response of a structure for discrete time steps t_i can be calculated within MATFEM based on the linear equation of motion

$$\mathbf{M}\ddot{\mathbf{U}}(t)+\mathbf{D}\dot{\mathbf{U}}(t)+\mathbf{K}\mathbf{U}(t)=\ \mathbf{F}(t) \tag{9.4.1}$$

where
$\mathbf{M},\mathbf{D},\mathbf{K}$ physical system matrices
$\ddot{\mathbf{U}},\dot{\mathbf{U}},\mathbf{U}$ acceleration, velocity, displacement

There are four classical methods available [Bathe 1], [Brodkorb 1] to calculate the time domain response:

- finite differences
- WILSON/TETA
- NEWMARK
- exact integration

An example for a dynamic response analysis in the time domain is given in the demonstration directory *demo13*. See also *demo13* in Appendix D for more information.

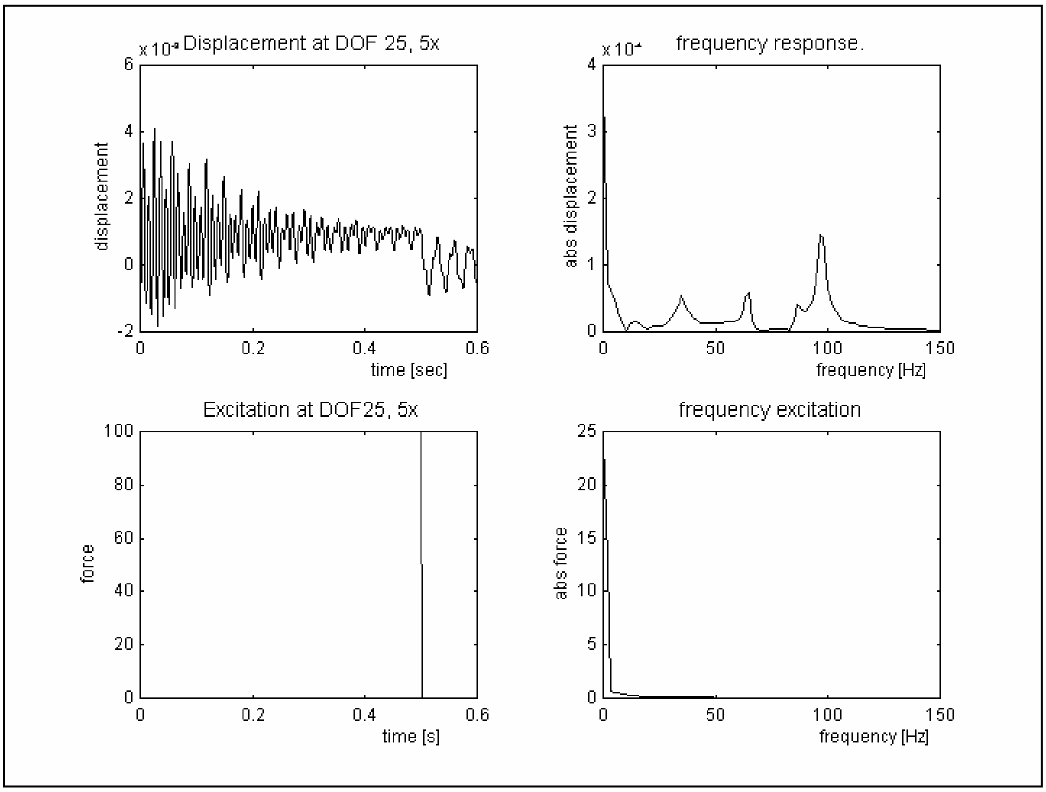


Fig. 9.4.1: Time domain response of an MDOF for a given excitation, amplitude spectrum of the response and the excitation, (demo13)

The parameters of a dynamic time domain response analysis are specified in the input file *.m14

resp_domain response domain:
= 0 no response calculation (suppress response calculation)
= 1 frequency domain (see chapter. 9.1)
= 2 time domain

Specification of modal parameters used for the response calculation:

name_modalresp name of a file with already existing modal data
if *name_modalresp* = [] the actually calculated modal data are used

nresp total number of modes to be included in the response calculation.
nresp = 0 suppresses response calculation
xeffno mode numbers to be included in the response calculation.
The user must take care that *xeffno* is less or equal to the number of modes, which will be calculated from the eigenvalue problem.

Specification of the damping matrix:
The damping matrix can be specified in physical or modal coordinates. The user has to specify one of the following matrices *C, xsi, xsid*

C upper triangle of physical damping matrix *C (na,na),* where *na* = number of physical DOF’s (line-by-line input)
xsi upper triangle of non-diagonal modal damping matrix xsi(nresp,nresp)
xsid diagonal of modal damping matrix *xsid (nresp).* Input in fractions of critical damping

Specification of dynamic excitation forces:

nef total number of nodal excitation forces
knr nodal points at which the excitation nodal forces apply, *knr(1, nef)*
ril DOF direction of *knr*
F excitation forces *F(:, 1+ nef).* In the first column the excitation time axis is specified. In columns 2 .. 1+ *nef* the *nef* excitation forces w.r. to the time axis of column 1 are specified.
example: Two excitation forces: a half cosine and a constant excitation force

```
A1 =  2        excitation amplitudes
A2 = -5
om = 2*pi* 10                     circular frequency
t_f= 0: pi/om/ 100: pi/om
F= [ t_f',  [ A1* cos( om* t_f)]',  A2* ones( length( t_f), 1) ]
```

Specification of DOF's for the response calculation (MDOF):

knresp node numbers for the response calculation
riresp DOF directions of knresp

plotresp switch for plotting the response

Specification of time response calculation method:

tdr_calc	finite differences, WILSON/TETA, NEWMARK, exact integration
wilsonteta_par	parameter of WILSON/TETA method (default 1.4)
newmark_par	parameter of NEWMARK method (default 0.5, 0.25)

Specification of initial conditions:

initcond_pnt	nodal point number with initial conditions different from zero
initcond_dir	DOF direction of *initcond_pnt*
initcond	initial conditions

Specification of time response axis:
The specification of the response time axis is limited to an even spaced time axis.

tdr_time_axis_par	lower time limit, upper time limit, time increment

The time domain response analysis results are stored in the MDB:

*.acc	time domain acceleration response
*.vel	time domain velocity response
*.dis	time domain displacement response

9.5 Calculation of eigensolutions using LANCZOS algorithm

There are four algorithms available within MATFEM to solve the eigenvalue problem

$$(-\lambda \mathbf{M} + \mathbf{K})\mathbf{X} = 0 \tag{9.5.1}$$

where

M	mass matrix
K	stiffness matrix
X	eigenvector
λ	eigenvalue

The user must choose one of the algorithms by specifying the parameter *ieig* in the input file **.m14* . The choice depends on whether the stiffness matrix is singular, i.e. rigid body modes of the structure are present, or regular. The choice also depends on whether all or only a few eigensolutions are required:

ieig = 1 ***M*** regular, ***K*** regular
MATLAB build-in eigensolution solver *eig* is used
all eigensolutions are calculated

ieig = 2 ***M*** regular, ***K*** singular
MATLAB build-in eigensolution solver *eig* is used
all eigensolutions are calculated

ieig = 4 MATFEM LANCZOS algorithm
only *nr* eigensolutions are calculated

ieig = 5 MATLAB build-in LANCZOS algorithm (*eigs*)
only *nr* eigensolutions are calculated

Known Limitations

The MATFEM LANCZOS algorithm (ieig= 4) works quiet well for most applications. However, the results may be erroneous in the very special case where the stiffness matrix is singular and where the stiffness terms of the overall stiffness matrix are totally decoupled from other stiffness terms. Consider a single beam element which local *x,y,z* axis coincides with the global coordinate system *XYZ*. The overall stiffness matrix of the beam is then totally decoupled w.r. to the longitudinal, bending and torsional stiffness. In this special case the results of the eigenvalue problem with free/free boundaries may be erroneous. To overcome this problem

- use *ieig* = 1, *ieig* = 2 or *ieig* = 5 instead

or

- restrain the appropriate DOF's and solve each rigid body mode separately

or

- change the local coordinate system of the structure, so that there is no decoupling of the overall stiffness matrix

9.6 Dynamic non-linear response analysis in the frequency domain using the 'Harmonic Balancing' approach

9. 6.1 Theoretical background of local non-linear 2DOF truss elements

Investigating the dynamic behaviour of real structures, non-linear effects can often be observed. Using classical linear experimental modal analysis procedures in these cases yields erroneous results. Therefore, these non-linear effects must be taken into account by using a suitable model. In this approach, the equation of motion is extended by additional terms describing the influence of non-linear stiffness and non-linear damping. These terms depend on powers of the displacement response. If the non-linearities are assumed to be weak, these non-linear equations can be linearized and transformed to the frequency domain following the procedure of the 'Harmonic Balance' method, [MEYER 1, MEYER 2].

We will focus here on 2-DOF Elements which enclose the special case of a SDOF Element. These local non-linear elements can be assembled into linear finite element models.

The 2-DOF Element consists of two masses connected by linear and non-linear damper(s) and linear and non-linear spring(s). Exemplary, a non-linear 2-DOF Spring is shown in Fig. 9.6.1. If a harmonic excitation is applied at mass m_1 and/or mass m_2, the non-linear spring forces at DOF 1 and DOF 2 can be expressed as a function of the power of the relative displacement between the two masses. This is also valid for non-linear dampers and the corresponding damping forces.

$$F_{nl} = f\left((u_1 - u_2)^a\right) = f\left(\varDelta u^a\right) \tag{9.6.1}$$

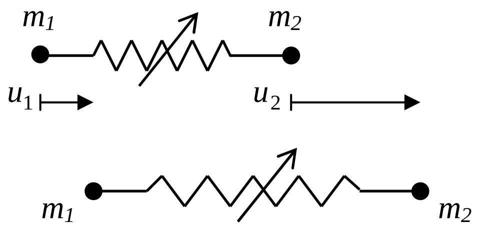


Fig. 9.6.1: 2-DOF Non-linear spring

Modelling approach in the time domain

The linear equation of motion is extended by additional stiffness and damping matrices which depend on powers of the relative displacement between the masses. This yields the non-linear equation of motion in the time domain as

$$\boldsymbol{M}\ddot{\boldsymbol{U}}(t) + \boldsymbol{C}\dot{\boldsymbol{U}}(t) + \boldsymbol{K}\boldsymbol{U}(t) + \sum_a \boldsymbol{C}_a \varDelta\dot{u}(t)^{a-1}\dot{\boldsymbol{U}}(t) + \sum_b \boldsymbol{K}_b \varDelta u(t)^{b-1}\boldsymbol{U}(t) = \boldsymbol{F}(t) \tag{9.6.2a}$$

with

$$\varDelta u(t) = u_1(t) - u_2(t), \tag{9.6.2b}$$

$$\varDelta\dot{u}(t) = \dot{u}_1(t) - \dot{u}_2(t). \tag{9.6.2c}$$

The values of a and b can be rational and non-rational. Two nodes can be connected by more than one local non-linear spring or damper.

Eqn. (9.6.2a) contains the linear element matrices

$$\boldsymbol{M} = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}, \quad \boldsymbol{C} = c \cdot \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad \boldsymbol{K} = k \cdot \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \qquad (9.6.3a,b,c)$$

and the element matrices describing the non-linear properties

$$\boldsymbol{C}_a = c_a \cdot \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad \boldsymbol{K}_b = k_b \cdot \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \qquad (9.6.3d,e)$$

c_a , k_b = damping/stiffness parameter of type a,b.

Transformation to the frequency domain

Eqn (9.6.2a) is linearized and transformed to the frequency domain assuming that for weak non-linearities the displacements $u_1(t)$ and $u_2(t)$ are also harmonic (following the procedure of the 'Harmonic Balance' approach)

$$u_1(t) = u_{1,re} \cdot cos(\omega t) + u_{1,im} \cdot sin(\omega t), \qquad (9.6.4a)$$

$$u_2(t) = u_{2,re} \cdot cos(\omega t) + u_{2,im} \cdot sin(\omega t). \qquad (9.6.4b)$$

Inserting Eqns (9.6.4a,b) into Eqn (9.6.2) yields, after some tedious mathematical operations, the following linearized equation of motion for the 2-DOF Element in the frequency domain

$$\left[\hat{\boldsymbol{Z}}_l + \hat{\boldsymbol{Z}}_{nl}\right] \cdot \begin{bmatrix} \boldsymbol{U}_{re} \\ \boldsymbol{U}_{im} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_{re} \\ \boldsymbol{F}_{im} \end{bmatrix}. \qquad (9.6.5a)$$

Eqn (9.6.5a) contains the linear dynamic stiffness matrix $\hat{\boldsymbol{Z}}_l$

$$\begin{aligned} \hat{\boldsymbol{Z}}_l &= \begin{bmatrix} -\omega^2 \boldsymbol{M} + \boldsymbol{K} & -\omega \boldsymbol{C} \\ \omega \boldsymbol{C} & -\omega^2 \boldsymbol{M} + \boldsymbol{K} \end{bmatrix} \\ &= -\omega^2 \begin{bmatrix} \boldsymbol{M} & 0 \\ 0 & \boldsymbol{M} \end{bmatrix} + \omega \begin{bmatrix} 0 & -\boldsymbol{C} \\ \boldsymbol{C} & 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{K} & 0 \\ 0 & \boldsymbol{K} \end{bmatrix} \qquad (9.6.5b) \\ &= -\omega^2 \hat{\boldsymbol{M}}_l + \omega \hat{\boldsymbol{C}}_l + \hat{\boldsymbol{K}}_l \end{aligned}$$

with the mass and the linear damping and stiffness matrix according to Eqns (9.6.3a-c) and it contains also the non-linear dynamic stiffness matrix $\hat{\boldsymbol{Z}}_{nl}$

$$\hat{\boldsymbol{Z}}_{nl} = \sum_{a,b} A^{a-1,b-1} \cdot \begin{bmatrix} \dfrac{k_b^*}{k} \cdot \boldsymbol{K} & -\omega^a \dfrac{c_a^*}{c} \cdot \boldsymbol{C} \\ \omega^a \dfrac{c_a^*}{c} \cdot \boldsymbol{C} & \dfrac{k_b^*}{k} \cdot \boldsymbol{K} \end{bmatrix}. \qquad (9.6.5c)$$

The displacement vector of the two degrees of freedom is given by $\boldsymbol{U}_{re,im}^{T} = \begin{bmatrix} u_1 & u_2 \end{bmatrix}_{re,im}$.

NOTE:

> The stiffness and damping factors have been changed (indicated by the '*') due to the linearization using the 'Harmonic Balance' approach. For cubic stiffness/damping this yields $c_a^* = \frac{3}{4} c_a, k_a^* = \frac{3}{4} k_a$. In MATFEM, the user has to specify the stiffness/damping factors k$_a$ and c$_a$ directly. The linearization is done automatically by MATFEM. For convenience, the '*'- symbol will be neglected in the following text.

According to Eqn (9.6.5b), Eqn (9.6.5c) can be divided into damping- and stiffness parts

$$\hat{\boldsymbol{Z}}_{nl} = \sum_a A^{a-1} \cdot \omega^a \cdot \frac{c_a}{c} \cdot \begin{bmatrix} 0 & -\boldsymbol{C} \\ \boldsymbol{C} & 0 \end{bmatrix} + \sum_b A^{b-1} \cdot \frac{k_b}{k} \cdot \begin{bmatrix} \boldsymbol{K} & 0 \\ 0 & \boldsymbol{K} \end{bmatrix} = \hat{\boldsymbol{C}}_{nl} + \hat{\boldsymbol{K}}_{nl} \ . \tag{9.6.5d}$$

The matrices in Eqns (9.6.5c,d) depend on the non-linearity parameters k_b and c_a and on the squared maximum displacement amplitude between the two degrees of freedom

$$A^{a-1,b-1} = \left(\sqrt{|u_1|^2 + |u_2|^2 - 2 \cdot \left(u_{1,re} \cdot u_{2,re} + u_{1,im} \cdot u_{2,im} \right)} \right)^{a-1,b-1} \tag{9.6.5e}$$

where $|u_1|^2 = u_{1,re}^2 + u_{1,im}^2$ and $|u_2|^2 = u_{2,re}^2 + u_{2,im}^2$ denote the square of the absolute response amplitudes.

Eqn (9.6.5e) yields $A = 0$ if $u_1 = u_2 \neq 0$, that means the non-linear forces depend on the relative displacement and not on the absolute displacement of u_1 and u_2.

The special case of a SDOF Element is achieved when setting the displacement $u_1 = 0$. That leads to a linear and non-linear dynamic stiffness matrix as shown in Eqns (9.6.5b,d) where the mass, damping and stiffness matrices only contain one element. The squared maximum displacement amplitude according to Eqn (9.6.5e) reduces to

$$A^{a-1,b-1} = \left(\sqrt{|u_2|^2 = u_{2,re}^2 + u_{2,im}^2} \right)^{a-1,b-1} \ . \tag{9.6.6}$$

The derived non-linear elements (SDOF or 2-DOF) may be assembled into linear FE models allowing to model complex structures with local non-linearities like those introduced at joints. The response of the non-linear system has to be calculated iteratively in the frequency domain. Using such a model to predict the measured response requires to assume numerical values for the linear and the non-linear model parameters.

9.6.2 Some explanations using local non-linear elements

The response of a system with local non-linearities has to be calculated iteratively. This is done using the MATLAB function ‘fsolve’ of the optimization toolbox (version 1.5 or 2.0).

Due to stiffness non-linearity, problems of convergence may occur. A big problem is the so called ‘jump phenomenon’ which can be seen if the influence of the local stiffness non-linearity exceeds a certain limit (which can not be specified). A typical shape of the response that has to be expected in such cases is shown in Fig. 9.6.2, where the influence of a stiffening stiffness non-linearity is shown qualitatively.

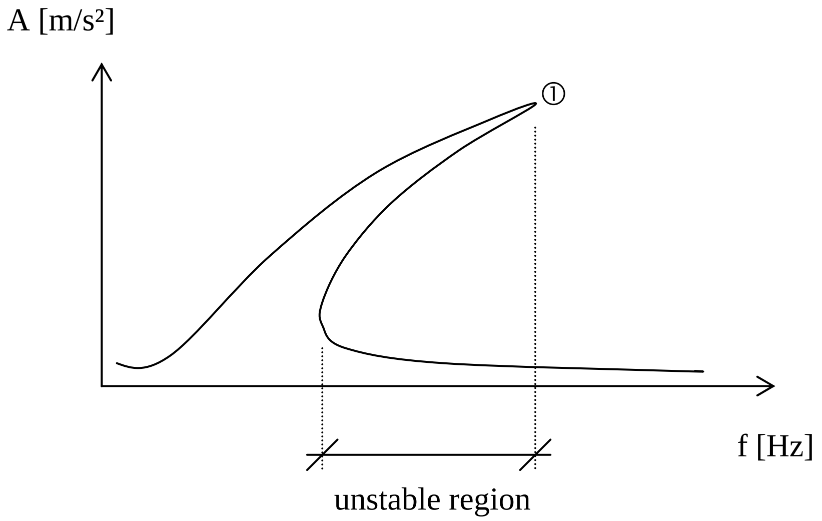


Fig. 9.6.2: Typical response shape of a system with stiffening stiffness non-linearity

A problem calculating the response of a system with such a non-linear behaviour is the so called ‘unstable region’ where three different response amplitudes are in general possible for the same excitation frequency. Only one of these three response amplitudes can be calculated using MATFEM. It depends on the starting values, what response amplitude is found. In general it can be stated, that it is possible to find either the upper branch of the response curve or the lower branch, but it will never be possible to find the middle branch.
To overcome some of the iteration problems, two major strategies are followed. The first is called the response calculation in ‘up’-direction, starting with small excitation frequencies and going up to higher excitation frequencies. Doing this, the results of the iterative calculation of the response are used to extrapolate the starting values for the response calculation at higher excitation frequencies. This is done using linear, quadratic and cubic extrapolation. In the ideal case this will lead to a response as indicated by the solid line in Fig. 9.6.3. It can be seen, that in the ideal case the total upper branch of the response is calculated and the jump occurs at point ①.

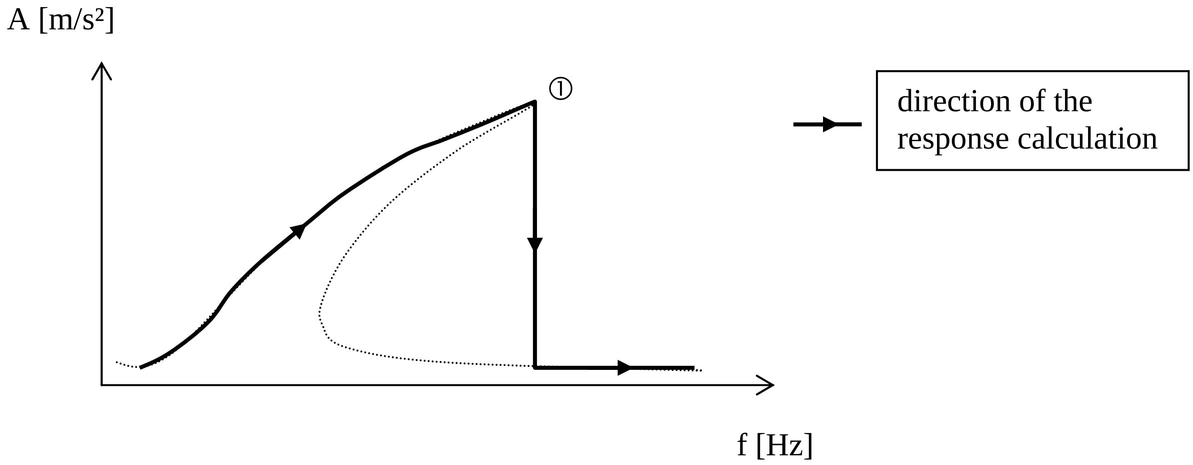


Fig. 9.6.3: Typical response shape of a system with stiffening stiffness non-linearity (calculated in 'up'-direction, ideal case)

The second strategy is called the response calculation in 'down'-direction, starting with high excitation frequencies and going down to smaller excitation frequencies. Doing this, the results of the iterative calculation of the response are used to extrapolate the starting values for the response calculation at smaller excitation frequencies. This is done using linear, quadratic and cubic extrapolation. In the ideal case this will lead to a response as indicated by the solid line in Fig. 9.6.4. It can be seen, that in the ideal case the total lower branch of the response is calculated.

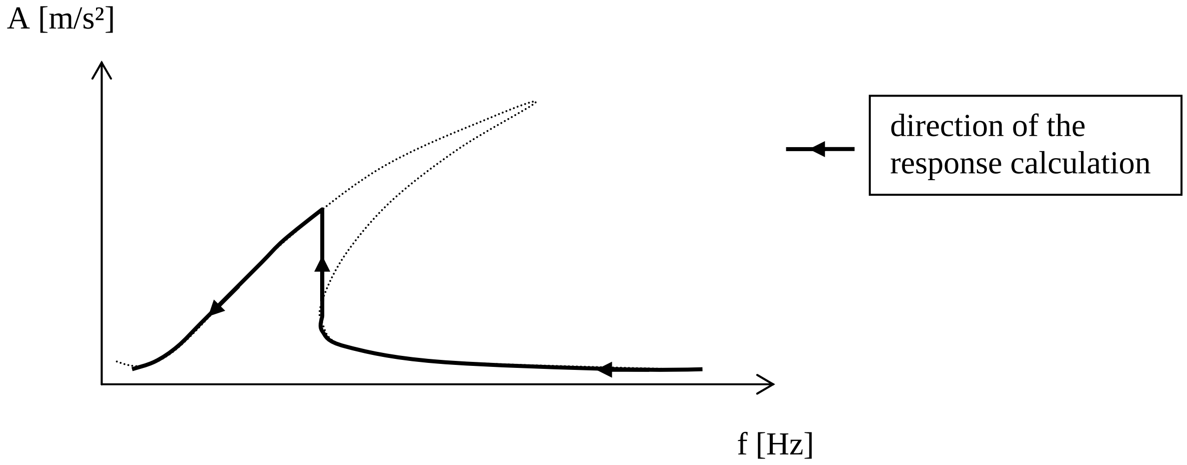


Fig. 9.6.4: Typical response shape of a system with stiffening stiffness non-linearity (calculated in 'down'-direction, ideal case)

A big problem is, that these ideal cases are almost never met. Especially during the response calculation in 'up' direction, problems of convergence often do occur. This leads to a jump to the lower branch, although the end of the upper branch (point ①) is not reached. This leads to a possible response curve which is shown in Fig. 9.6.5. Once the response jumps to the lower branch, there is almost no possibility to get back to the upper branch.

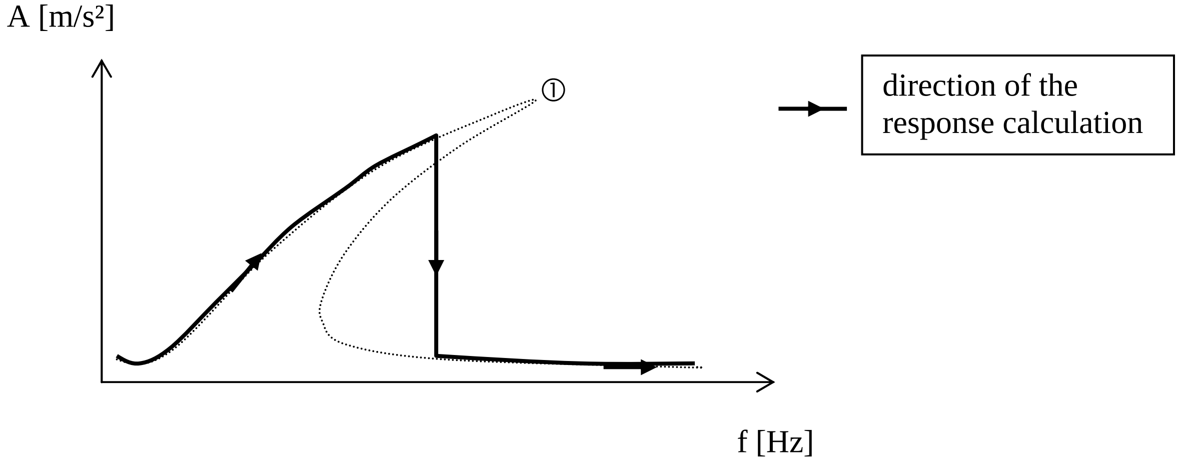


Fig. 9.6.5: Typical response shape of a system with stiffening stiffness non-linearity (calculated in 'up'-direction, general case)

The location of the jump depends on the parameters which are set during the call of 'fsolve'. Sometimes even for the same parameters of iteration the jump does occur at different frequency points. This behaviour should always be kept in mind when dealing with local non-linearites.

Another problem is the number of equations that have to be minimized using 'fsolve'. Due to the mathematical formulation of the non-linear equation of motion, the imaginary and the real parts of the response at all degrees of freedom (DOF's) are needed. For the demo16 example with 5 DOF's this leads to a total amount of 5*2=10 values to be minimized using 'fsolve'. The higher the number of equations, the more often iteration problems (no convergence) are caused. Some experience with a system having 27 DOF's (= 54 values to be minimized) indicates that for a system of that size these kind of problems do occur and cause a sever lack of reproductivity of the location of the jump phenomenon.

NOTE:

The non-linear dynamic response calculation in MATFEM should (by now) only be used for small order systems. In the future, the Craig-Bampton sub-structure coupling shall be used to minimize the number of DOF's to improve the robustness and the speed of the non-linear response calculation.

9.6.3 Specification of local non-linearities: 2DOF truss elements

The specification of the geometry of local non-linear 2DOF truss elements is done in *.m04. It is recommended to use *lines* to define the geometry of these elements. The meshing is also done in *.m04 using a partitioning vector. Here, the user must use a partition of [0 1 NaN], so that the nodes of the truss element correspond with basic nodal points (BNP).

example:

```
linien = [ 1  6 0; 2 7 0; 3  8 0; 4 9 0; 5 10 0; 6 7 0; 7  8 0; 8 9 0; ...
           9 10 0; 6 8 0; 8 10 0; 3 8 0; 6  7 0; 7 8 0];

xsil   = [ 0:1 0:1 0:1 0:1 0:1 0:1 0:1 0:1 0:1 0:1 0:1 0:1 0:1 0:1];

% coincidence line to beam elementcard EC_beam
ec_l_koi = [ 1 1 2 1 1 3 3 3 3 4 4 5 6 7];
```

Fig. 9.6.6: Specification of geometry of a system with 10 BNP, 11 linear *linien* parts consisting of one element and 3 non-linear *linien* parts consisting of one element (demo16). 7 elementcards are defined, No. 1-4 for linear elements and No. 5-7 for non-linear elements.

The properties of the linear and non-linear elements are specified in *.m07. They are assigned in EC_beam. For the non-linear 2DOF truss elements only the beam_type (111), the kind of local non-linear stiffness and local non-linear damping and the corresponding stiffness and damping factors have to be defined. Each elementcard for a beam_type 111-116 can contain up to three (115) or six (all others) local non-linear stiffness elements (max. one for each DOF). Each elementcard for a beam_type 111-112, 116 can contain up to six local non-linear damping elements (one for each DOF). The stiffness and damping factors have to be specified directly in EC_beam (all input w.r. to the local coordinate system!!). All local stiffness elements of one elementcard are of the same type of non-linearity (the same holds for the local damping). The type of local non-linear stiffness and local non-linear damping can be different using the same elementcard.

example:

```
% -- linear 2 node, 12 DOF --    nonlinear 2node, 2 DOF
      EC_beam =  [      102  102  102  102         111    111    111         %  1
                          1    1    1    1           3      0      2         %  2
                          1    2    3    4           0      0  1.1781*1e7    %  3
                          1    1    1    1           0      0      0         %  4
                          1    2    3    4    1.333*1e11    0      0         %  5
                          1    2    3    4           0      0      0         %  6
                          1    2    3    4           0      0      0         %  7
                          1    2    3    4           0      0      0         %  8
                          0    0    0    0           0      2      3         %  9
                          0    0    0    0           0    11.781  1.333*5   % 10
                          0    0    0    0           0      0      0         % 11
                          0    0    0    0           0      0      0         % 12
                          0    0    0    0           0      0      0         % 13
                          0    0    0    0           0      0      0         % 14
                          0    0    0    0           0      0      0         % 15
                          0    0    0    0           0      0      0         % 16
                          0    0    0    0           0      0      0         % 17
                          0    0    0    0           0      0      0         % 18
                          0    0    0    0           0      0      0         % 19
                          0    0    0    0           0      0      0         % 20
                          0    0    0    0           0      0      0         % 21
                 ];
```

Fig. 9.6.7: Specification of properties of linear and non-linear beam elements. Card 5-7 contain non-linear element properties (beam_type 111). Card 5 defines a local non-linear cubic stiffness of value 1.333*1e11 in local z-direction, card 6 defines a local non-linear quadratic damper of value 11.781 in local x-direction and card 7 defines a local non-linear quadratic stiffness of value 1.1781*1e7 in local x-direction and a local non-linear cubic damper of value 1.333*5 in local x-direction (demo16).

NOTE:

The values of the type of non-linearity can also be non-rational (e.g. 3.1 or 1.74)!

9.6.4 Dynamic non-linear frequency response calculation

The parameters of a non-linear dynamic frequency response analysis are specified in the input file *.m14. The input parameters are the same as shown in chapter 9.1 for the linear dynamic response calculation. One additional parameter has to be defined to specify the direction of the non-linear response calculation:

step_dir iteration step direction
0 – ‘up’
1 – ‘down’
2 – ‘up’ + ‘down’

Start analysis by pushing the ’**run ...**’ – button. For a view of the linear response results click on ‘**linear response**’, for a view of the non-linear response results click on ‘**non-linear response’**.

NOTE:

The response of the underlying linear system is always calculated first during a MATFEM non-linear response calculation. Therefore the non-linear elements are neglected. It has to be taken great care, that the underlying linear system consist of only one structure which is not separated into decoupled substructures when the non-linear elements are neglected during the linear response calculation.

NOTE:

If the iteration procedure has difficulties to find a stable solution for each excitation frequency a first step to improve the results is to increase the number of frequency points in the selected frequency range. This will yield better starting values for the iterative non-linear response calculation and therefore possibly lead to better results.

All input and output parameters are specified in *.aus – file. Please use 'edit.aus' button to take a look at the specified data.

The linear response analysis parameters and results are stored in the MDB as shown in chapter 9.1. The parameters and results of the non-linear response calculation are stored in up to six files as listed below:

*.phd	damping matrix transformed to physical coordinates
*.nlp	parameters of local non-linear elements
*.rp4	real part of non-linear frequency response, 'up' - direction
*.rp5	imaginary part of non-linear frequency response, 'up' - direction
*.rp6	real part of non-linear frequency response, 'down' - direction
*.rp7	imaginary part of non-linear frequency response, 'down' - direction

NOTE:

The non-linear **acceleration** frequency response is calculated in MATFEM.

9.7 Dynamic Condensation for non-linear response calculation (in the frequency domain)

9. 7.1 Theoretical background of dynamic condensation

For the dynamic condensation, master (index u) and slave (index s) DOF's have to be defined. The system matrices are divided according to these DOF's as shown in the equation of motion given in Eqn (9.7.1):

$$\left\{-\omega^2\begin{bmatrix}\mathbf{M}_{uu} & \mathbf{M}_{us}\\ \mathbf{M}_{su} & \mathbf{M}_{ss}\end{bmatrix}+\omega\begin{bmatrix}\mathbf{C}_{uu} & \mathbf{C}_{us}\\ \mathbf{C}_{su} & \mathbf{C}_{ss}\end{bmatrix}+\begin{bmatrix}\mathbf{K}_{uu} & \mathbf{K}_{us}\\ \mathbf{K}_{su} & \mathbf{K}_{ss}\end{bmatrix}\right\}\cdot\begin{bmatrix}\mathbf{U}_u\\ \mathbf{U}_s\end{bmatrix}=\begin{bmatrix}\mathbf{F}_u\\ \mathbf{0}\end{bmatrix} \tag{9.7.1}$$

with $\mathbf{K}_{us}=\mathbf{K}_{su}^T,\ \mathbf{C}_{us}=\mathbf{C}_{su}^T,\ \mathbf{M}_{us}=\mathbf{M}_{su}^T$.

From the lower row of Eqn (9.7.1) the following mathematical expression for the connection between the displacements at the slave DOF's and the master DOF's can be derived:

$$\mathbf{U}_s=-\left(\mathbf{K}_{ss}+\omega\mathbf{C}_{ss}-\omega^2\mathbf{M}_{ss}\right)^{-1}\cdot\left(\mathbf{K}_{su}+\omega\mathbf{C}_{su}-\omega^2\mathbf{M}_{su}\right)\cdot\mathbf{U}_u=\mathbf{T}^*\left(\omega\right)\cdot\mathbf{U}_u. \tag{9.7.2}$$

This yields the transformation matrix $\mathbf{T}^*(\omega)$ for the dynamic condensation:

$$\mathbf{T}^*\left(\omega\right)=-\left(\mathbf{K}_{ss}+\omega\mathbf{C}_{ss}-\omega^2\mathbf{M}_{ss}\right)^{-1}\cdot\left(\mathbf{K}_{su}+\omega\mathbf{C}_{su}-\omega^2\mathbf{M}_{su}\right). \tag{9.7.3}$$

The general transformation matrix $\mathbf{T}(\omega)$is then given by:

$$\mathbf{T}\left(\omega\right)=\begin{bmatrix}\mathbf{I}\\ \mathbf{T}^*\left(\omega\right)\end{bmatrix}. \tag{9.7.4}$$

The condensed equation of motion can now be calculated using $\mathbf{T}(\omega)$. This yields

$$\left(-\omega^2\mathbf{M}_{cc}+i\omega\mathbf{C}_{cc}+\mathbf{K}_{cc}\right)\cdot\mathbf{U}_u=\mathbf{F}_u \tag{9.7.5}$$

with

$$\mathbf{M}_{cc}\left(\omega\right)=\mathrm{Re}\left[\mathbf{T}^T\left(\omega\right)\cdot\mathbf{M}\cdot\mathbf{T}\left(\omega\right)\right], \tag{9.7.6}$$

$$\mathbf{C}_{cc}\left(\omega\right)=\mathrm{Re}\left[\mathbf{T}^T\left(\omega\right)\cdot\mathbf{C}\cdot\mathbf{T}\left(\omega\right)\right], \tag{9.7.7}$$

$$\mathbf{K}_{cc}\left(\omega\right)=\mathrm{Re}\left[\mathbf{T}^T\left(\omega\right)\cdot\mathbf{K}\cdot\mathbf{T}\left(\omega\right)\right]. \tag{9.7.8}$$

The condensed system matrices **M**, **C** and **K** in general have complex values (due to the complex transformation matrix in Eqn (9.7.4)). The complex parts are so small that they can be neglected (as indicated in Eqns (9.7.6-8)).

The condensed force vector is given by $\mathbf{F}_c=\mathbf{T}^T\cdot\mathbf{F}$. (9.7.9)

9. 7.2 Dynamic condensation in MATFEM

Dynamic condensation can only be used for wahl == 8 (non-linear response calculation in the frequency domain). To calculate a linear response using dynamic condensation, simply set all non-linearity factors in *.m07 (EC_beam) to zero.

All input for dynamic condensation is defined in *.m14. There, the master DOF’s have to be defined.

example:

Demo18, ECL

```
% --- DOF numbers of master DOF's for dynamic condensation (only for wahl == 8)

      % --- master node numbers of main structure

            %- in X -direction   Dcx(1,:)
               Dcx =  [];

            %- in Y -direction   Dcy(1,:)
               Dcy =  [];

            %- in Z -direction   Dcz(1,:)
               Dcz =  [2:5 7 8];

            %- in XX -direction  Dcxx(1,:)
               Dcxx = [];

               %- in YY -direction  Dcyy(1,:)
               Dcyy = [];

            %- in ZZ -direction  Dczz(1,:)
               Dczz = [];

% --- dynamic condensation  dyn_f(1,:) -----------------------------------
      %     all or some discrete excitation frequencies are used to calculate
      %     the dynamic condensation matrix
      %     special cases:  dyn_f = [0] --> static condensation
      %                     dyn_f = [ ] --> using each excitation frequency for condensation

            dyn_f = [];

% --- dynamic condensation  dyn_wc(1,1) ----------------------------------
      %     calculation of transformation matrix including damping
      %     (exact, dyn_wc = 1, default) or without damping (not exact, dyn_wc = 0)
```

Fig. 9.7.1: Input of master DOF’s in demo18.m14

The initial system has 27 DOF’s, 13 of them are in translational z-direction, 14 of them are in rotational yy-direction. These 27 DOF’s are condensed to 6 master DOF’s in translational z-direction (see Fig. 9.7.1 and ‘demo18’).

The vector *dyn_f* can be specified. The specified frequencies (in Hz) will be used to calculate the dynamic condensation matrices. The condensation will be exact at the specified frequencies and the calculated response will therefore be exact, too. For all other frequency points, the calculated response will deviate from the exact response. The larger the distance between excitation frequency point and given frequency in *dyn_f*, the more the response will deviate from the exact value. If only one frequency is specified, the dynamic condensation matrix is calculated once and is the same for all excitation frequencies. If no frequency is specified, each excitation frequency will be used to calculate a new condensation matrix for each frequency point. The special case of a static condensation is given by setting *dyn_f* to [0].

The value *dyn_wc* can be defined to specify, if the damping shall be taken into account when calculating the transformation matrix T(ω) (default)

NOTE:

Take care that no forces are applied at slave DOF’s (or an error message will be shown).

Some information about the numbers of master and slave DOF’s of the unconstrained and the constrained system are saved to MATFEM data basis in ***.dyc**.

9.8 Iterative Calculation of non-linear acceleration response

The iterative non-linear response calculation in MATFEM can be done using either physical coordinates or modal coordinates. This can be selected in the *.m14 input template (see below)

```
% --- kind of non-linear response calculation  kind_it(1,1)
      %         = 0 physical iterative response calculation (default)
      %         = 1 modal    iterative response calculation

            kind_it = [--- input ---];
```

Fig. 9.8.1: Selection of non-linear iterative response calculation

The general procedure during the iterative response calculation is explained below. The only difference is the definition of the force vectors for physical/modal iteration. Whereas for the physical response calculation the system matrices and the corresponding external force vectors are used directly, the system matrices and the external force vectors are condensed into modal space using the eigenvectors of the underlying linear system. Therfore the displacement vectors are replaced by

$$\mathbf{U}_{re,im} = \mathbf{\Phi}^T \cdot \mathbf{q}_{re,im} \tag{9.8.1}$$

The non-linear acceleration response is calculated iteratively. Therefore, the error between external excitation forces and the vector of internal forces (resulting from inertia, damping or stiffness) is minimized. This is done using the Least-Squares approach. Shown below is the response calculation using the physical system matrices and the corresponding external forces directly.

The vector of external excitation forces is given by

$$\mathbf{F}_{ex} = \begin{bmatrix} \mathbf{F}_{ex,real} \\ \mathbf{F}_{ex,imag} \end{bmatrix}, \tag{9.8.2}$$

where the imaginary part (corresponding to the sine part) of excitation usually equals zero. But that is no necessity of the presented iteration algorithm.

The Vector of internal forces is given by the left part of the equation of motion, if this is written in the 2n*2n –size:

$$\mathbf{F}_{in} = \begin{bmatrix} -\omega^2\mathbf{M} + \mathbf{K} + \mathbf{K}(U) & -\omega\mathbf{C} - \mathbf{C}(U) \\ +\omega\mathbf{C} + \mathbf{C}(U) & -\omega^2\mathbf{M} + \mathbf{K} + \mathbf{K}(U) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{U}_{real} \\ \mathbf{U}_{imag} \end{bmatrix} \tag{9.8.3}$$

where **K**(U) and **C**(U) are non-linear stiffness and damping matrices which depend on powers of the relative displacement between two DOF's. These local non-linear elements lead to non-linear force-parts, which have to be taken into account. That is the reason, why the response calculation has to be done iteratively.

For a given excitation frequency ω, the arbitrary selection of initial values for the displacement response parts $\mathbf{U}_{\text{real}}$ and $\mathbf{U}_{\text{imag}}$, will yield force vectors $\mathbf{F}_{\text{ex}}$ and $\mathbf{F}_{\text{in}}$ which will not be the same. (only in the special case where the sought-after displacement response has been used as initial values by some coincidence). The residual is defined by

$$\Delta\mathbf{F} = \mathbf{F}_{ex} - \mathbf{F}_{in} \tag{9.8.4}$$

which will, in general, yield a value unequal to zero. Main objective of the iterative calculation of the response is the minimization of the residual given by Eqn (9.8.4). That is done by a Least-Squares approach. Therefore, the objective function given by

$$\mathbf{J} = \Delta\mathbf{F}^T \cdot \Delta\mathbf{F} \;\rightarrow\; \min \tag{9.8.5}$$

is minimized. The model vector $\mathbf{F}_{\text{in}}$ is therefore given by a Taylor-series, which is truncated after the first element.

$$\mathbf{F}_{in}(U) = \mathbf{F}_{ana} + \mathbf{G} \cdot \Delta\mathbf{U} \tag{9.8.6a}$$

where

$$\mathbf{F}_{ana} = \mathbf{F}\Big|_{U=U_{ana}} \tag{9.8.6b}$$

$$\mathbf{G} = \frac{\partial\mathbf{F}}{\partial\mathbf{U}}\bigg|_{U=U_{ana}} . \tag{9.8.6c}$$

The residual vector of Eqn (9.8.4) is now given by

$$\Delta\mathbf{F} = \mathbf{F}_{ex} - \mathbf{F}_{ana} - \mathbf{G} \cdot \Delta\mathbf{U} = \mathbf{r}_a - \mathbf{G} \cdot \Delta\mathbf{U} \tag{9.8.7a}$$

where

$$\mathbf{r}_a = \mathbf{F}_{ex} - \mathbf{F}_{ana} . \tag{9.8.7b}$$

From the request that $\partial\mathbf{J}/\partial\Delta\mathbf{U} = 0$ the minimum of the objective function can be found. Then, the changes of the response values are after one iteration step given by

$$\Delta\mathbf{U} = \left(\mathbf{G}^T\mathbf{G}\right)^{-1}\mathbf{G}^T \cdot \mathbf{r}_a . \tag{9.8.8a}$$

The number of parameters (response values) equals the number of available equations. Therefore, the sensitivity matrix becomes quadratic. The pseudo-inverse can be replaced by the normal inverse of the sensitivity matrix **G**. That yields

$$\Delta\mathbf{U} = \mathbf{G}^{-1} \cdot \mathbf{r}_a . \tag{9.8.8b}$$

or

$$\begin{bmatrix} \Delta\mathbf{U}_{real} \\ \Delta\mathbf{U}_{imag} \end{bmatrix} = \mathbf{G}^{-1} \cdot \begin{bmatrix} \mathbf{r}_{a,real} \\ \mathbf{r}_{a,imag} \end{bmatrix}. \tag{9.8.8c}$$

The sensitivity matrix is calculated by finite differences. Therefore, for each displacement DOF ΔU a vector of the sensitivity matrix **G** is calculated.

$$\mathbf{G}_j = \frac{\mathbf{F}_{in}(\mathbf{U}, U_j + 0.001 \cdot U_j) - \mathbf{F}_{in}(\mathbf{U})}{0.001 \cdot U_j} \tag{9.8.9}$$

After assembling the complete sensitivity matrix

$$\mathbf{G}_{(n \cdot n)} = {}_n\begin{bmatrix} \mathbf{G}_1 & \cdots & \mathbf{G}_j & \cdots & \mathbf{G}_n \end{bmatrix}^n \tag{9.8.10}$$

the changes of the response values can be found by using Eqn (9.8.8). The response values after the i-th iteration step are given by

$$\mathbf{U}_{ana,i+1} = \mathbf{U}_{ana,i} + \Delta\mathbf{U}_i \, . \tag{9.8.11}$$

These will be used as initial values for the next iteration step. The procedure will continue until the norm of the residual vector is under a certain threshold. The criterion for the end of the iteration is given by

$$\varepsilon \le norm\left(\Delta\mathbf{F}\right). \tag{9.8.12}$$

10. References

[Link 1] Link, M.
Finite Elemente in der Statik und Dynamik, 3. Auflage
Stuttgart/Leipzig/Wiesbaden, 2002, ISBN 3-519-22953-6

[Gruber 1] Gruber, J.
Stabilitäts- und Spannungsberechnung nach Theorie II. Ordnung von räumlichen Rahmenstrukturen
Diplomarbeit, Universität GH Kassel, 1993

[Matlab 1] MATLAB
High Performance Numeric Computation and Visualization Software
Reference Guide
The MathWorks, Inc
24 Prime Park Way, Natick, Mass

[Craig 1] Craig R.R. , Bampton M.C.C.
Coupling of Substructures for Dynamic Analysis
AIAA Journal, Vol. 6, No. 7, 1968

[Brillhart 1] Brillhart R., Hunt, D. L.
Computation of Total Response Mode Shapes Using Tuned Frequency Response Functions
IMAC 4, Los Angeles 1986

[Kreyszig 1] Kreyszig E.
Advanced Engineering Mathematics, 5th Edition
John Wiley & Sons, New York 1983

[Gröger 1] Gröger, G.
Entwicklung und Erprobung eines dreieckigen ebenen Schalenelementes mit Schubverformung
Diplomarbeit, Universität GH Kassel, 2000

[Jeych&Kirk 1] Jeychandrabose C. Kirkhope J.
An Alternative Explicit Formulation for the DKT Plate-Bending Element
Int. J. Num. Methods in Engineering. Vol. 21, 1289-1293 (1985)

[Long&Xu 1] Long Yu, Xu Yin
Generalized conforming quadrilateral membrane element with vertex rigid rotational freedom
Computers & Structures, Vol. 5.2, No. 4, pp 749-755, (1994)

[Bathe 1] Bathe, K. J.
Finite Element Procedures
Prentice Hall, 1996
ISBN 0-13-301458-4

[Brodkorb 1] Brodkorb, J.
Zeitbereichsverfahren zur Berechnung der dynamischen Antwort
Projektarbeit, FB 14 , Fachgebiet Leichtbau und Strukturmechanik
Universität Gh Kassel, Kassel, 1998

[Zou 1] Zou Z., Link M.
A quasi conforming plate element for the analysis of plates and shells
Internal Report
Department of civil engineering
Laboratory of Lightweight Structures and Structural Mechanics
University of Kassel, Kassel, 1995

[Meyer 1] Meyer, S.; Weiland, M.; Link, M.
Identifikation von lokalen Steifigkeits- und Dämpfungsnichtlinearitäten
VDI-Schwingungstagung 2000, VDI Berichte 1550, Kassel, 2000

[Meyer 2] Meyer, S.
Modellbildung und Identifikation von lokalen nichtlinearen Steifigkeits- und Dämpfungseigenschaften in komplexen strukturdynamischen Finite Elemente Modellen, Dissertation
Universität Kassel, Kassel, 2003