FACULTY OF ENGINEERING
DEPARTMENT OF CIVIL ENGINEERING
STRUCTURAL MECHANICS SECTION
KASTEELPARK ARENBERG 40
B-3001 LEUVEN

Stabil

MATLAB toolbox for Structural Mechanics

USER'S GUIDE
STABIL VERSION 3.1
APRIL 2020
REPORT BWM-2020-05

Contact information

Web	`http://bwk.kuleuven.be/bwm/stabil`
Email	`stabil@kuleuven.be`
Phone	+32 16 32 16 82
Address	Structural Mechanics Section Department of Civil Engineering, KU Leuven Kasteelpark Arenberg 40, B-3001 Leuven, Belgium

Copyright

Acknowledgements

Stabil has been developed in the frame of OOI Project 2006/20 "An interactive and adaptive application for the static and dynamic analysis of structures", funded by the KU Leuven Educational Policy Unit. The financial support is gratefully acknowledged.

Contents

Chapter 1

Getting started

1.1 About Stabil

Stabil is a MATLAB toolbox for structural mechanics, based on the finite element method. Classical finite element beam and truss elements, as well as shell elements and elements for 2D and 3D elasticity have been implemented, and the toolbox can be used to solve a variety of static and dynamic structural problems. In addition, Stabil contains a number of postprocessing functions tailored to structural analysis, including plots of member forces and displacements of a structure. The user can interact with Stabil at a low level of abstraction or a high level of abstraction, which is further detailed in this manual. Due to this multi-level approach, the toolbox is suitable for educational purposes and for use in a research environment: the high level functions allow for an easy and efficient implementation of many common problems, while the low level functions facilitate customization and the implementation of novel finite element techniques in a research context.

Stabil is written in standard MATLAB language, so that the source code is available to the user. No additional MATLAB toolboxes are required and the toolbox can be used in a Windows, Linux, or MacOS environment.

1.2 Obtaining and installing Stabil

Stabil can be downloaded from the internet at `https://bwk.kuleuven.be/bwm/stabil`. It is distributed as a ZIP archive, which should be extracted to a directory on the hard disk, e.g. `C:\Users\/%username/%\Documents\matlab\stabil`. After extraction of the ZIP archive, this directory should contain a number of `m`-files.

The Stabil directory must be subsequently added to the MATLAB path to make the toolbox functions available in MATLAB:
- In MATLAB, click on ‘Set Path...’ in the ‘Home’ tab.
- Click on ‘Add Folder’ and select the Stabil directory.
- Save the path and close the dialog window.

1.3 Terms of use

Stabil is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. The toolbox can be used and modified for educational and research purposes. However, it cannot be used for consulting, nor commercialized in any form. Regardless of any provision of the GNU General Public License, Stabil may not be used for commercial purposes without explicit written permission from the authors.

Furthermore, if you use Stabil for research, please include a reference to Stabil in scientific publications of your work (articles, reports, books, etc.).

Stabil is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with Stabil. If not, see `https://www.gnu.org/licenses/`.

1.4 Scope and structure of this document

The present document is the user's guide to Stabil version 4.0. It is a combination of a reference guide, providing an overview of all functions of Stabil, and a tutorial, presenting a set of examples to illustrate the use of Stabil. This document is not meant as a textbook: a theoretical background is only considered where this is necessary to define the functionality of Stabil in a clear way. The reader is referred to various works on Matrix analysis of Structures [5], Finite Elements [1, 3, 4, 6–9], or Dynamics of Structures [2] for a broad theoretical background.

This document is composed of the following chapters:

Chapter 1. Getting started (p. 1)
The aim of this chapter is to get the user started with Stabil and the accompanying user's guide. The installation procedure of Stabil is explained, the terms of use of Stabil are clarified, and the scope and structure of the user's guide are discussed.

Chapter 2. Static analysis of structures (p. 3)
In this chapter, the use of stabil for the static analysis of structures is explained through a number of examples.

Chapter 3. Dynamic analysis of structures (p. 25)
In this chapter, the use of stabil for the dynamic analysis of structures is explained through a number of examples.

Chapter 4. Element guide (p. 37)
This chapter presents an overview of the element types available in Stabil. The conventions (local coordinate system, nodal connectivity) for each element type are given and reference is made to the Stabil functions related to the element implementation.

Chapter 5. Functions — By category (p. 55)
This chapter gives an overview of all functions in Stabil, organized by category.

Chapter 6. Functions — Alphabetical list (p. 57)
This chapter consists of an alphabetical list of the functions in Stabil. The syntax, the input and output arguments, and the use of each function are described in detail. The information provided in this chapter is also accessible at the MATLAB prompt through the `help` command.

Chapter 2

Static analysis of structures

2.1 Example 1.1: static analysis of a frame

The basic concepts of the Stabil toolbox are introduced through the example of a simple frame structure, shown in figure 2.1. The frame has a height and width of 4 m and is clamped at node 1 and pinned at node 5. An internal hinge is present at nodes 2 and 3, and a diagonal brace is present between nodes 1 and 4. At node 4 a point load $F = 5\,\mathrm{kN}$ is applied and the vertical beam on the left is loaded by a distributed load $p = 2\,\mathrm{kN/m}$. The beams and columns have a concrete rectangular cross-section with a width of 0.2 m and a height of 0.4 m. The concrete has a Young's modulus of 30 GPa and a Poisson's ratio of 0.2. The diagonal brace has a circular steel cross-section with a diameter of 8 mm with a Young's modulus of 210 GPa and a Poisson coefficient of 0.3.

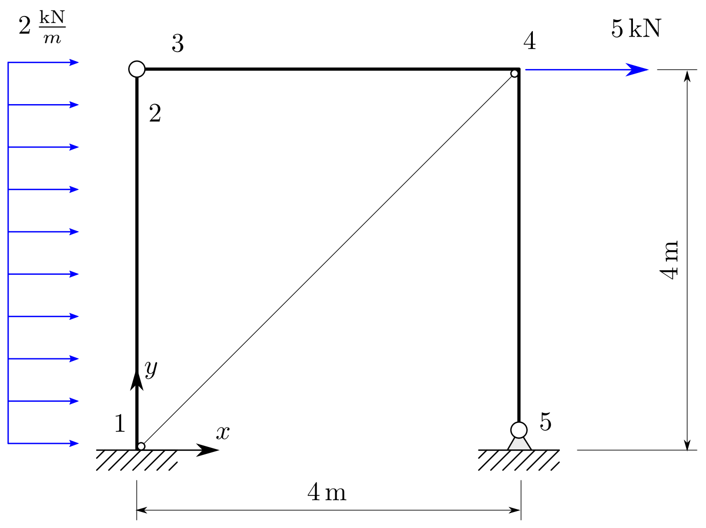


Figure 2.1: Simple frame structure.

The code to compute the deformation and bending moment in the frame structure is listed below:

```
% StaBIL manual
% Example 1.1: static analysis of a frame
% Units: m, kN

% Nodes=[NodID X  Y  Z]
Nodes=  [1     0  0  0;
         2     0  4  0;
         3     0  4  0;
         4     4  4  0;
         5     4  0  0;
         6     1  5  0];          % reference node

% Check the node coordinates as follows:
figure
plotnodes(Nodes);
```

```
% Element types -> {EltTypID EltName}
Types=  {1        'beam';
         2        'truss'};

b=0.10;
h=0.25;
r=0.004;

% Sections=[SecID A      ky   kz   Ixx Iyy Izz       yt   yb   zt   zb]
Sections=  [1     b*h    Inf  Inf  0   0   b*h^3/12  h/2  h/2  b/2  b/2;
            2     pi*r^2 NaN  NaN  NaN NaN NaN       NaN  NaN  NaN  NaN];

% Materials=[MatID     E nu];
Materials=  [1      30e6 0.2;               % concrete
             2     210e6 0.3];              % steel

% Elements=[EltID TypID SecID MatID n1 n2 n3]
Elements=  [1     1     1     1     1  2  6;
            2     1     1     1     3  4  6;
            3     1     1     1     5  4  6;
            4     2     2     2     1  4  NaN];

% Check node and element definitions as follows:
hold('on');
plotelem(Nodes,Elements,Types);
title('Nodes and elements');

% Degrees of freedom
% Assemble a column matrix containing all DOFs at which stiffness is
% present in the model:
DOF=getdof(Elements,Types);

% Remove all DOFs equal to zero from the vector:
%  - 2D analysis: select only UX,UY,ROTZ
%  - clamp node 1
%  - hinge at node 5
seldof=[0.03; 0.04; 0.05; 1.00; 5.01; 5.02];
DOF=removedof(DOF,seldof);

% Assembly of stiffness matrix K
K=asmkm(Nodes,Elements,Types,Sections,Materials,DOF);

% Nodal loads: 5 kN horizontally on node 4.
seldof=[4.01];
PLoad=    [5];

% Assembly of the load vectors:
P=nodalvalues(DOF,seldof,PLoad);

% Distributed loads are specified in the global coordinate system
% DLoads=[EltID n1globalX n1globalY n1globalZ ...]
DLoads=  [1     2 0 0 2 0 0];

P=P+elemloads(DLoads,Nodes,Elements,Types,DOF);

% Constraint equations: Constant=Coef1*DOF1+Coef2*DOF2+ ...
% Constraints=[Constant  Coef1 DOF1  Coef2 DOF2 ...]
Constr=       [0         1     2.01  -1    3.01;
               0         1     2.02  -1    3.02];

% Add constraint equations
[K,P]=addconstr(Constr,DOF,K,P);

% Solve K * U = P
U=K\P;
```

```
% Plot displacements
figure
plotdisp(Nodes,Elements,Types,DOF,U,DLoads,Sections,Materials)

% The displacements can be displayed as follows:
printdisp(Nodes,DOF,U);

% Compute element forces
Forces=elemforces(Nodes,Elements,Types,Sections,Materials,DOF,U,DLoads);

% The element forces can be displayed in a orderly table:
printforc(Elements,Forces);

% Plot element forces
figure
plotforc('norm',Nodes,Elements,Types,Forces,DLoads)
title('Normal forces')

figure
plotforc('sheary',Nodes,Elements,Types,Forces,DLoads)
title('Shear forces')

figure
plotforc('momz',Nodes,Elements,Types,Forces,DLoads)
title('Bending moments')

% Plot stresses
figure
plotstress('snorm',Nodes,Elements,Types,Sections,Forces,DLoads)
title('Normal stresses due to normal forces')

figure
plotstress('smomzt',Nodes,Elements,Types,Sections,Forces,DLoads)
title('Normal stresses due to bending moments around z: top')

figure
plotstress('smomzb',Nodes,Elements,Types,Sections,Forces,DLoads)
title('Normal stresses due to bending moments around z: bottom')

figure
plotstress('smax',Nodes,Elements,Types,Sections,Forces,DLoads)
title('Maximal normal stresses')

figure
plotstress('smin',Nodes,Elements,Types,Sections,Forces,DLoads)
title('Minimal normal stresses')
```

After the definition of model parameters, the analysis starts by defining the nodes of the model, where the nodes are defined in the (x,y)-plane. The nodes are represented by a matrix that contain node numbers as a first column, followed by the x, y and z coordinates of each node. Next, the element types are defined by a cell array, `Types`, containing type numbers followed by a string (either `beam` or `truss`). The `Sections` matrix defines the section properties (cross section, moments of inertia,...). In a similar way, the `Materials` matrix defines material properties (the Young's modulus and Poisson coefficient). The model definition is then completed by providing an element connectivity table `Elements`, which refers to the previously defined element, material, section and node numbers. In order to uniquely define a local coordinate system for the beam elements, use is made of a reference node (node 6). The local x-axis is directed from the first node of the element to the last node, the local y-axis is perpendicular to the local x-axis with its origin corresponding to the first node of the element, and pointing in the direction of the reference node. The same convention applies in a three-dimensional setting, as indicated in the `beam` element reference sheet on page 38.

Next, the degrees of freedom (DOF's) are specified. The DOF's are defined using a `node.index` approach, where the digits to the left of the decimal point refer to the node number and the digits to the right of the decimal point refer to degree of freedom. By convention, the degrees of freedom `01`, `02`, and `03` correspond to translations u_x, u_y, and u_z in the global coordinate directions whereas the degrees of freedom `04`, `05`, and `06`

Node	U_x [m]	U_y [m]	U_z [m]	φ_x [rad]	φ_y [rad]	φ_z [rad]
1	0.0000e+0	0.0000e+0	0.0000e+0	0.0000e+0	0.0000e+0	0.0000e+0
2	6.6633e-3	3.2297e-6	0.0000e+0	0.0000e+0	0.0000e+0	-1.8161e-3
3	6.6633e-3	3.2297e-6	0.0000e+0	0.0000e+0	0.0000e+0	4.0356e-4
4	6.6538e-3	-3.6160e-5	0.0000e+0	0.0000e+0	0.0000e+0	-8.3665e-4
5	0.0000e+0	0.0000e+0	0.0000e+0	0.0000e+0	0.0000e+0	- 2.0769e-3
6	0.0000e+0	0.0000e+0	0.0000e+0	0.0000e+0	0.0000e+0	0.0000e+0
Element Node	**N [kN]**	**V_y [kN]**	**V_z [kN]**	**T [kNm]**	**M_y [kNm]**	**M_z [kNm]**
1 i	6.0557e-1	6.2201e+0	-0.0000e+0	-0.0000e+0	-0.0000e+0	-8.8804e+00
1 j	6.0557e-1	-1.7799e+0	0.0000e+0	0.0000e+0	0.0000e+0	-4.4409e-16
2 i	-1.7799e+0	6.0557e-1	-0.0000e+0	-0.0000e+0	-0.0000e+0	0.0000e+00
2 j	-1.7799e+0	6.0557e-1	0.0000e+0	0.0000e+0	0.0000e+0	2.4223e+00
3 i	-6.7799e+0	-6.0557e-1	-0.0000e+0	-0.0000e+0	-0.0000e+0	6.6613e-16
3 j	-6.7799e+0	-6.0557e-1	0.0000e+0	0.0000e+0	0.0000e+0	-2.4223e+00
4 i	8.7318e+0	-0.0000e+0	-0.0000e+0	-0.0000e+0	-0.0000e+0	0.0000e+00
4 j	8.7318e+0	0.0000e+0	0.0000e+0	0.0000e+0	0.0000e+0	-0.0000e+00

Table 2.1: Nodal displacements, reaction forces, and member forces for example 1.1.

correspond to rotations φ_x, φ_y, φ_z around the global coordinate axes. For example, DOF `3.02` represents the translation u_y of node 3. This approach of defining the degrees of freedom for the problem at hand is very instructive, since it enforces reasoning on the kinematics of the structure and how boundary conditions are accounted for. This manual input of the vector of degrees of freedom is seen as a low-level functionality of the toolbox. Alternatively, the high-level functions (`getdof`, `selectdof`, `removedof`) could be used to generate and process this DOF vector, allowing for more complex structural models to be analyzed in an efficient way.

The finite element stiffness matrix is next assembled using the `asmkm` function. This function takes the `Nodes`, `Elements`, `Types`, `Sections`, and `Materials` variables that define the finite element model and assembles the sparse global stiffness matrix `K` corresponding to the `dof` vector. The `asmkm` function is a high-level function that loops over the various elements of the mesh and calls low-level functions that generate element stiffness matrices.

In Stabil, both nodal loads as well as distributed loads on elements can be considered, and dedicated functions are available to either generate nodal loads on DOFs (the `nodalvalues` function) or distributed loads on elements (the `elemloads` function). The load vector `P` is available in the Matlab environment, and has the same size as the `dof`-vector.

In order to account for the internal hinge between nodes 2 and 3, constraint equations are added to the system of equations to couple the horizontal and vertical displacements of both nodes. This is achieved through the function `addconstr` that modifies the stiffness matrix `K` to account for linear constraint equations that relate various degrees of freedom in the system.

The finite element system of equations ($\mathbf{Ku} = \mathbf{f}$) is solved using the Matlab $\backslash$ (left divide, `mldivide`) command. Since the global stiffness matrix `K` is a sparse matrix, a sparse solver is used by Matlab by default. The resulting displacement vector `u` has the same size as the `dof` vector and the load vector `P` and is equally available in the Matlab environment.

In Stabil, a number of postprocessing functions is available to easily plot the deformed structure or the resulting member forces. Figure 2.2 shows the resulting displacement of the structure as obtained with the `plotdisp` command. The plotdisp command takes the model definition (`Nodes`, `Elements`, `Types`), the displacement solution and corresponding `dof` vector, and (optionally) the distributed load with the section and material definitions (`Sections`, and `Materials`). A key feature of the Stabil toolbox is that the deformed shape is plotted in an exact way, accounting for the (cubic) shape functions of the beam element and the deformation of the element due to distributed loads. This allows to demonstrate how distributed loads are reduced to equivalent nodal loads.

Figure 2.3 shows the member forces in the structure as plotted with the `plotforc` command. Like the `plotdisp` command, the `plotforc` command accounts for the effect of distributed loads, as is apparent in the quadratic variation of the bending moment in the left column in figure 2.3.

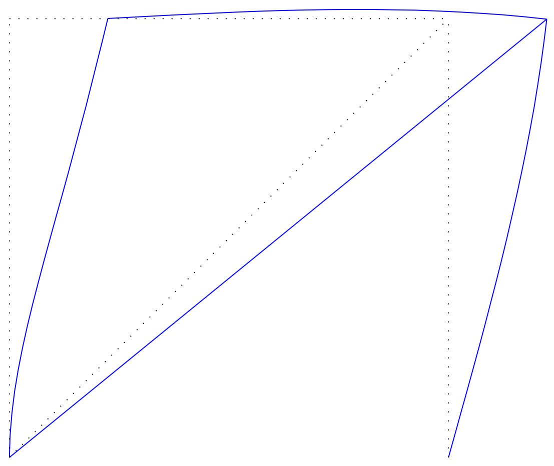
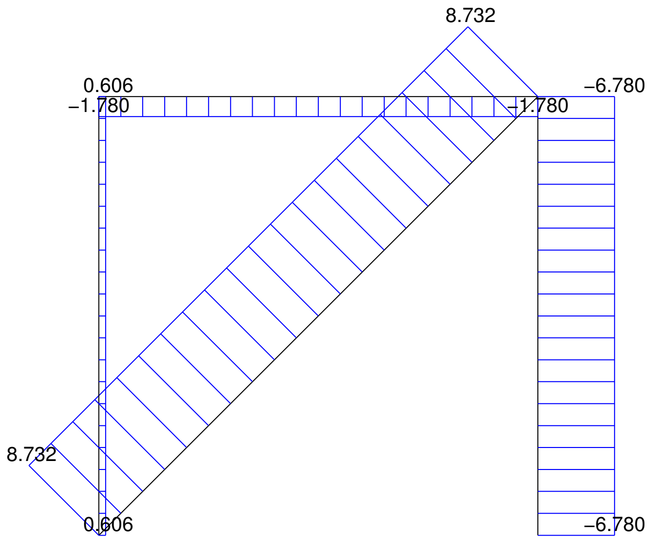
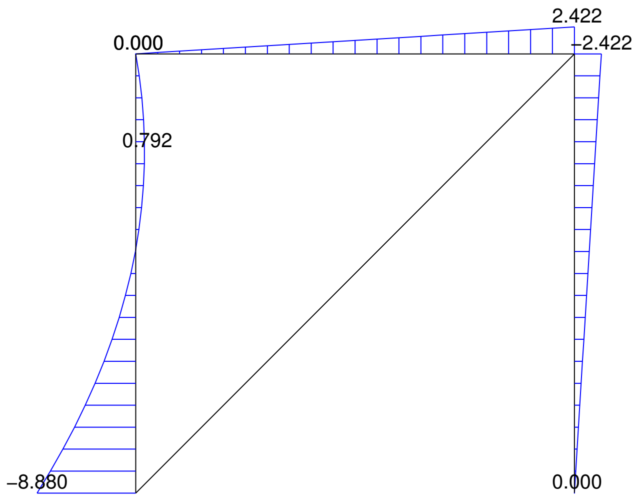


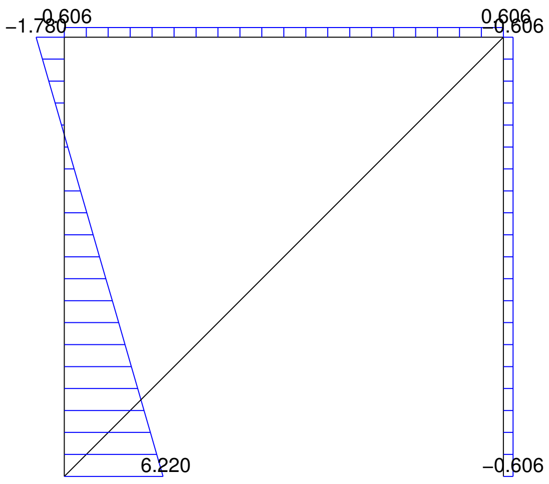
Figure 2.2: Deformed frame structure.



(a)

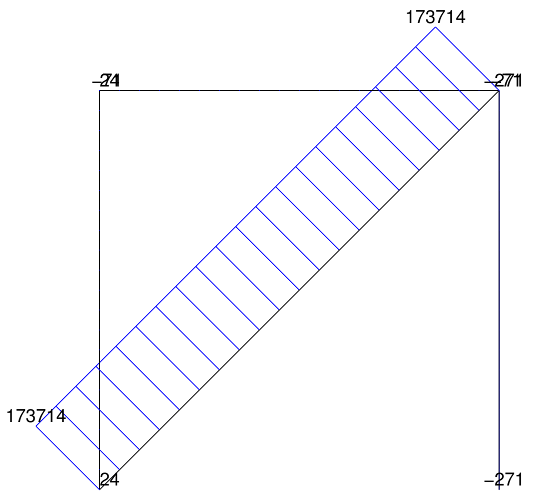


(b)

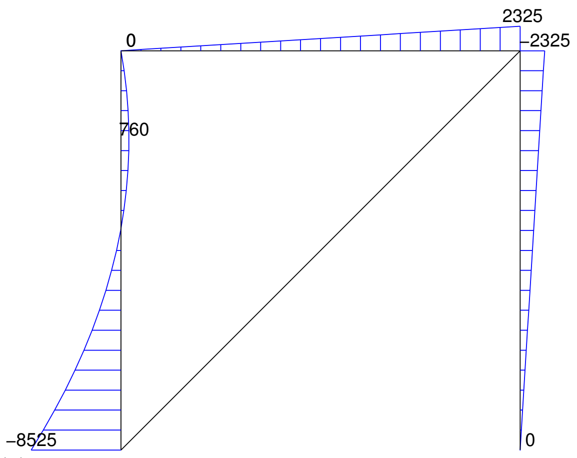


(c)

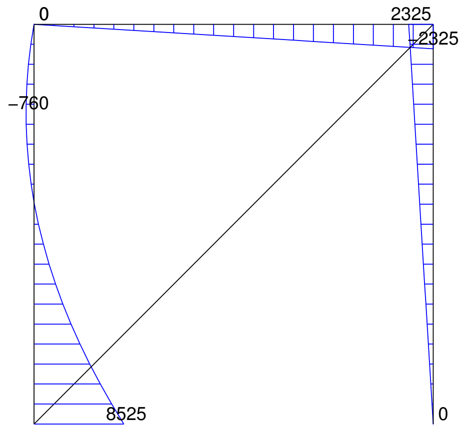
Figure 2.3: (a) Normal forces [kN], (b) bending moments [kNm], and (c) shear forces [kN] in the frame structure.



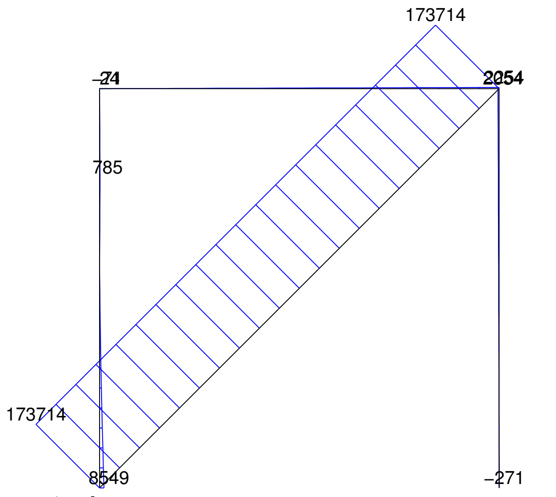
(a) normal forces



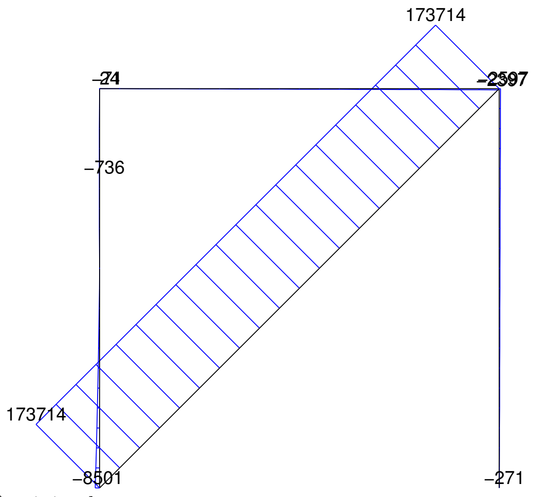
(b) bending moments around z: top



(c) bending moments around z: bottom



(c) maximal



(d) minimal

Figure 2.4: Normal stresses.

2.2 Example 1.2: static analysis of a 3D frame

In order to demonstrate the three-dimensional capabilities of Stabil, the following example considers a 3D frame structure (figure 2.5). The frame consists of rectangular concrete beams (Young's modulus $E = 35 \times 10^9\,\mathrm{N/m^2}$ and Poisson coefficient $\nu = 0.2$) with a height of 0.5 m and a width of 0.2 m and concrete columns of 0.3 m by 0.2 m.

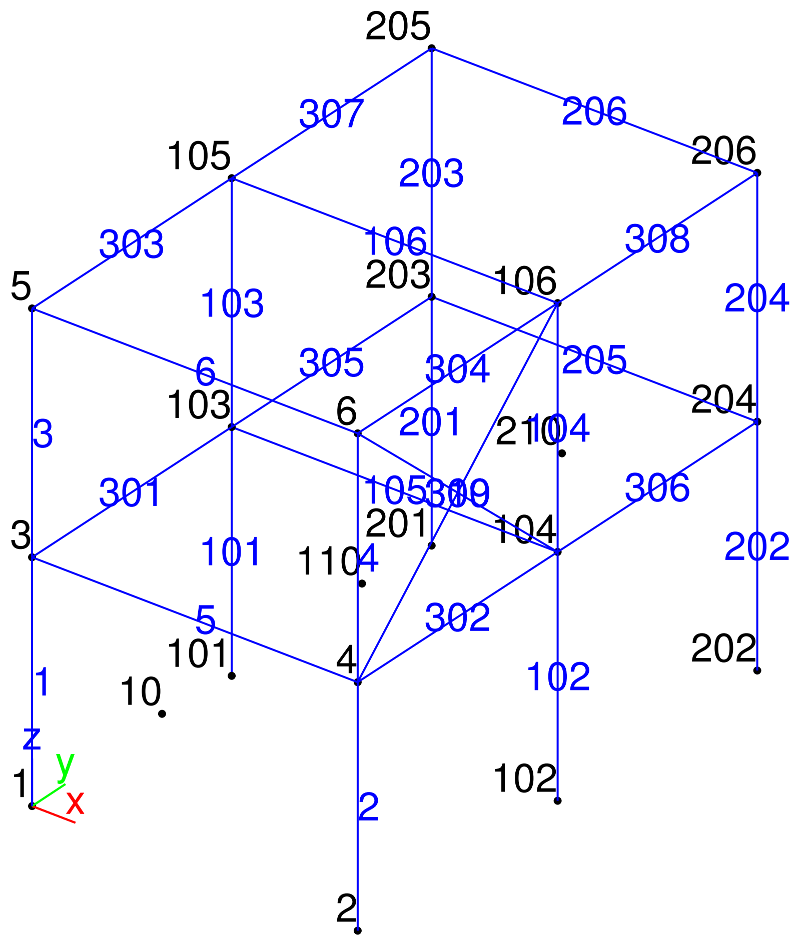


Figure 2.5: Example 1.2: nodes, elements, loads and boundary conditions.

The following input file is structured in a similar way as example 1.1. It starts by defining nodes, element types, sections, and materials. Subsequently, the elements are defined by the use of a reference node. The `reprow` command is employed to replicate rows of the `Nodes` and `Elements` matrix. The `getdof` and `seldof` commands are then used to generate and process the DOF vector. After assembling the stiffness matrix and assembling the loads, the system is solved, the forces and reaction forces are computed and results are finally plotted, resulting in figures 2.6 to 2.9 for the different load cases and load combinations.

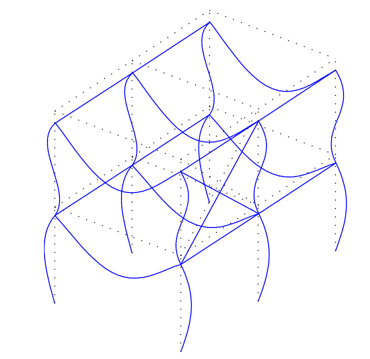
```
% StaBIL manual
% Example 1.2: static analysis of a 3D frame
% Units: m, kN

% Types={EltTypID EltName}
Types=  {1        'beam';
         2        'truss'};

% Sections
bCol=0.2;  % Column section width
hCol=0.3;  % Column section height
bBeam=0.2; % Beam section width
hBeam=0.5; % Beam section height
Atruss=0.001;
```

```
% Sections=[SecID A ky kz Ixx Iyy Izz yt yb zt zb]
Sections=  [1  hCol*bCol   Inf Inf 0.196*bCol^3*hCol   bCol^3*hCol/12 ...
               hCol^3*bCol/12  hCol/2  hCol/2  bCol/2  bCol/2;   % Columns
            2  hBeam*bBeam Inf Inf 0.249*bBeam^3*hBeam bBeam^3*hBeam/12 ...
               hBeam^3*bBeam/12 hBeam/2 hBeam/2 bBeam/2 bBeam/2; % Beams
            3  Atruss NaN NaN NaN NaN NaN NaN NaN NaN NaN];

% Materials=[MatID E      nu      rho];
Materials=  [1     35e9   0.2     2500;   %concrete
             2     210e9  0.3     7850];  %steel

L=5;
H=3.5;
B=4;

% Nodes=[NodID X  Y  Z]
Nodes=  [1     0  0  0;
         2     L  0  0]
Nodes=reprow(Nodes,1:2,2,[2 0 0 H])
Nodes=[Nodes;
         10    2  0  2]             % reference node
Nodes=reprow(Nodes,1:7,2,[100 0 B 0])

figure
plotnodes(Nodes);

% Elements=[EltID TypID SecID MatID n1 n2 n3]
Elements=[ 1      1     1     1     1  3  10;
           2      1     1     1     2  4  10];
Elements=reprow(Elements,1:2,1,[2 0 0 0 2 2 0])
Elements=[ Elements;
           5      1     2     1     3  4  10;
           6      1     2     1     5  6  10];
Elements=reprow(Elements,1:6,2,[100 0 0 0 100 100 100])
Elements=[ Elements;
          301     2     3     2     3  103 NaN;
          302     2     3     2     4  104 NaN];
Elements=reprow(Elements,19:20,1,[2 0 0 0 2 2 0])
Elements=reprow(Elements,19:22,1,[4 0 0 0 100 100 0])
Elements=[ Elements;
          309     2     3     2      4 106 NaN;
          310     2     3     2      6 104 NaN];

hold('on');
plotelem(Nodes,Elements,Types);
title('Nodes and elements');

% Plot elements in different colors in order to check the section definitions
figure
plotelem(Nodes,Elements(find(Elements(:,3)==1),:),Types,'r');
hold('on');
plotelem(Nodes,Elements(find(Elements(:,3)==2),:),Types,'g');
plotelem(Nodes,Elements(find(Elements(:,3)==3),:),Types,'b');
title('Elements: sections')

% Degrees of freedom
DOF=getdof(Elements,Types);

% Boundary conditions: hinges
seldof=[  1.01;    1.02;    1.03;    2.01;    2.02;    2.03;
        101.01; 101.02; 101.03; 102.01; 102.02; 102.03;
        201.01; 201.02; 201.03; 202.01; 202.02; 202.03;];

DOF=removedof(DOF,seldof);
```

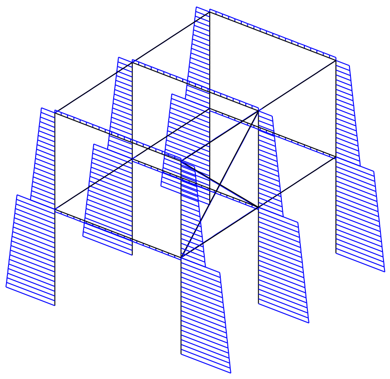
```
% Assembly of stiffness matrix K
K=asmkm(Nodes,Elements,Types,Sections,Materials,DOF);

% Loads

% Own weight
DLoadsOwn=accel([0 0 9.81],Elements,Types,Sections,Materials);

% Wind load

% DLoads=[EltID n1globalX n1globalY n1globalZ ...]
DLoadsWind =[1 0 0    0 0 1500 0;
             2 0 0    0 0 1500 0;
             3 0 1500 0 0 1500 0;
             4 0 1500 0 0 1500 0];

DLoads=multdloads(DLoadsOwn,DLoadsWind);

P=elemloads(DLoads,Nodes,Elements,Types,DOF);

% Solve K * U = P
U=K\P;

figure
plotdisp(Nodes,Elements,Types,DOF,U(:,1),DLoads(:,:,1),Sections,Materials)
title('Displacements: own weight')

figure
plotdisp(Nodes,Elements,Types,DOF,U(:,2),DLoads(:,:,2),Sections,Materials)
title('Displacements: wind')

% Compute forces
[ForcesLCS,ForcesGCS]=elemforces(Nodes,Elements,Types,Sections,Materials,DOF,U,DLoads);

% Compute reaction forces for load case 1
Freac=reaction(Elements,ForcesGCS(:,:,1),[1.03; 2.03; 101.03; 102.03; 201.03; 202.03])

% Plot element forces for load case 1
figure
plotforc('norm',Nodes,Elements,Types,ForcesLCS(:,:,1),DLoads(:,:,1))
title('Normal forces: Own weight')
figure
plotforc('sheary',Nodes,Elements,Types,ForcesLCS(:,:,1),DLoads(:,:,1))
title('Shear forces along y: Own weight')
figure
plotforc('shearz',Nodes,Elements,Types,ForcesLCS(:,:,1),DLoads(:,:,1))
title('Shear forces along z: Own weight')
figure
plotforc('momx',Nodes,Elements,Types,ForcesLCS(:,:,1),DLoads(:,:,1))
title('Torsional moments: Own weight')
figure
plotforc('momy',Nodes,Elements,Types,ForcesLCS(:,:,1),DLoads(:,:,1))
title('Bending moments around y: Own weight')
figure
plotforc('momz',Nodes,Elements,Types,ForcesLCS(:,:,1),DLoads(:,:,1))
title('Bending moments around z: Own weight')

% Plot element forces for load case 2
figure
plotforc('norm',Nodes,Elements,Types,ForcesLCS(:,:,2),DLoads(:,:,2))
title('Normal forces: Wind')
figure
plotforc('sheary',Nodes,Elements,Types,ForcesLCS(:,:,2),DLoads(:,:,2))
title('Shear forces along y: Wind')
figure
plotforc('shearz',Nodes,Elements,Types,ForcesLCS(:,:,2),DLoads(:,:,2))
```

```
title('Shear forces along z: Wind')
figure
plotforc('momx',Nodes,Elements,Types,ForcesLCS(:,:,2),DLoads(:,:,2))
title('Torsional moments: Wind')
figure
plotforc('momy',Nodes,Elements,Types,ForcesLCS(:,:,2),DLoads(:,:,2))
title('Bending moments around y : Wind')
figure
plotforc('momz',Nodes,Elements,Types,ForcesLCS(:,:,2),DLoads(:,:,2))
title('Bending moments around z: Wind')

% Load combinations

% Safety factors
gamma_own=1.35;
gamma_wind=1.5;

% Combination factors
psi_wind=1;

% Load combination (Ultimate Limit State, ULS)
U_ULS=gamma_own*U(:,1)+gamma_wind*psi_wind*U(:,2);
Forces_ULS=gamma_own*ForcesLCS(:,:,1)+gamma_wind*psi_wind*ForcesLCS(:,:,2);
DLoads_ULS(:,1)=DLoads(:,1,1)
DLoads_ULS(:,2:7)=gamma_own*DLoads(:,2:7,1)+gamma_wind*psi_wind*DLoads(:,2:7,2);

figure
plotdisp(Nodes,Elements,Types,DOF,U_ULS,DLoads_ULS,Sections,Materials)

printdisp(Nodes,DOF,U_ULS);

printforc(Elements,Forces_ULS);

% Plot element forces
figure
plotforc('norm',Nodes,Elements,Types,Forces_ULS,DLoads_ULS)
title('Normal forces: ULS')
figure
plotforc('sheary',Nodes,Elements,Types,Forces_ULS,DLoads_ULS)
title('Shear forces along y: ULS')
figure
plotforc('shearz',Nodes,Elements,Types,Forces_ULS,DLoads_ULS)
title('Shear forces along z: ULS')
figure
plotforc('momx',Nodes,Elements,Types,Forces_ULS,DLoads_ULS)
title('Torsional moments: ULS')
figure
plotforc('momy',Nodes,Elements,Types,Forces_ULS,DLoads_ULS)
title('Bending moments around y: ULS')
figure
plotforc('momz',Nodes,Elements,Types,Forces_ULS,DLoads_ULS)
title('Bending moments around z: ULS')

% Plot stresses
figure
plotstress('snorm',Nodes,Elements,Types,Sections,Forces_ULS,DLoads_ULS)
title('Normal stresses due to normal forces')
figure
plotstress('smomyt',Nodes,Elements,Types,Sections,Forces_ULS,DLoads_ULS)
title('Normal stresses due to bending moments around y: top')
figure
plotstress('smomyb',Nodes,Elements,Types,Sections,Forces_ULS,DLoads_ULS)
title('Normal stresses due to bending moments around y: bottom')
figure
plotstress('smomzt',Nodes,Elements,Types,Sections,Forces_ULS,DLoads_ULS)
title('Normal stresses due to bending moments around z: top')
```

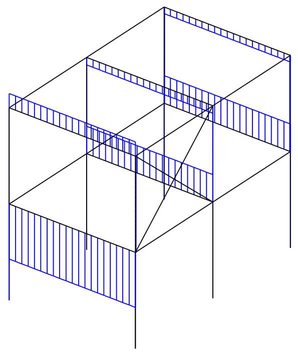
```
figure
plotstress('smomzb',Nodes,Elements,Types,Sections,Forces_ULS,DLoads_ULS)
title('Normal stresses due to bending moments around z: bottom')
figure
plotstress('smax',Nodes,Elements,Types,Sections,Forces_ULS,DLoads_ULS)
title('Maximal normal stresses')
figure
plotstress('smin',Nodes,Elements,Types,Sections,Forces_ULS,DLoads_ULS)
title('Minimal normal stresses')
```



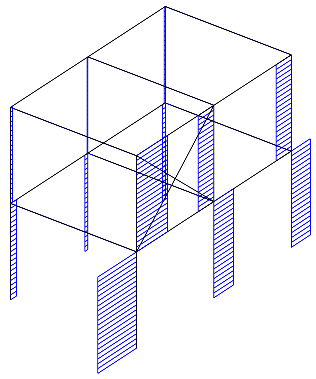
(a) Displacements



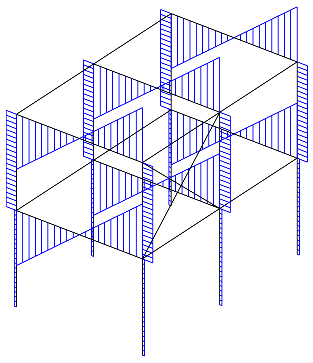
(b) Normal forces



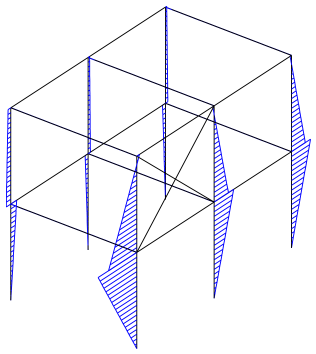
(c) Torsional moments



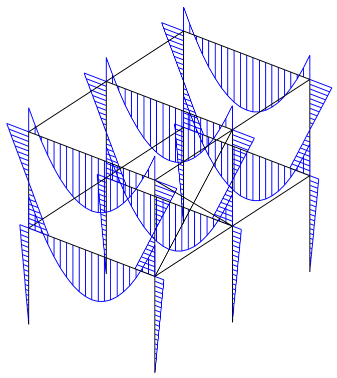
(d) Shear forces along z



(e) Shear forces along y

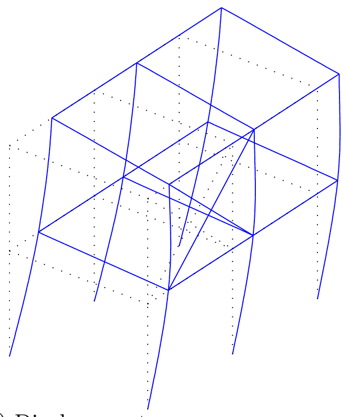


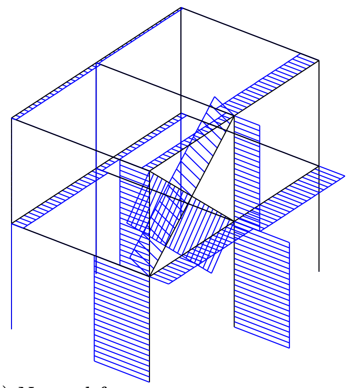
(f) Bending moments around y

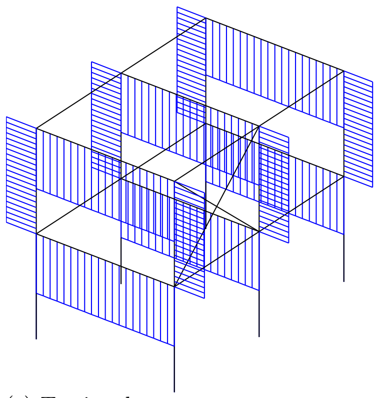


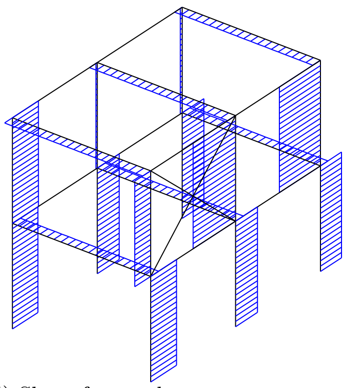
(g) Bending moments around z

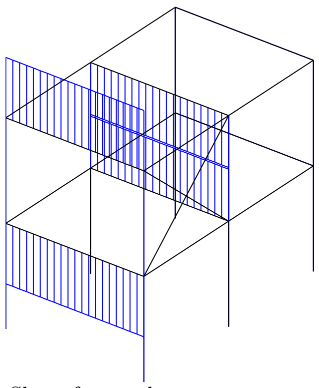
Figure 2.6: Results for load case 1 of example 1.2.

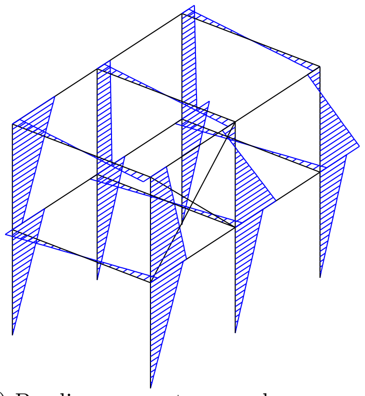

(a) Displacements


(b) Normal forces


(c) Torsional moments


(d) Shear forces along z


(e) Shear forces along y


(f) Bending moments around y

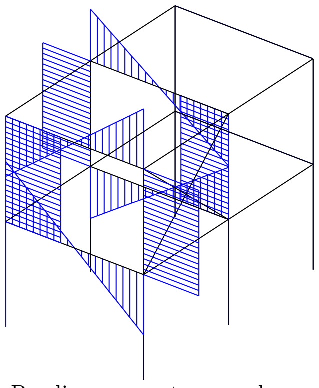
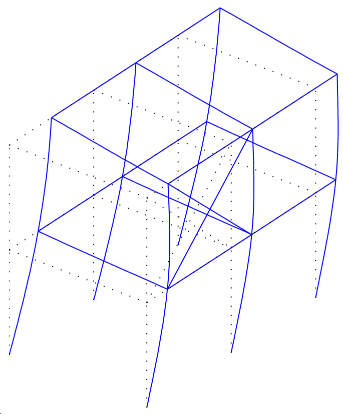
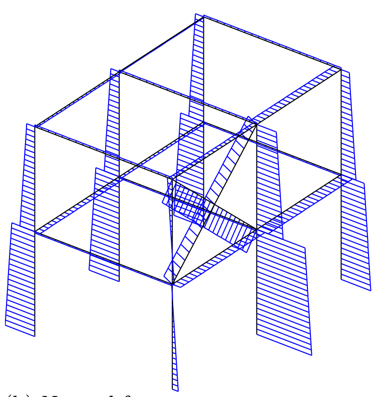
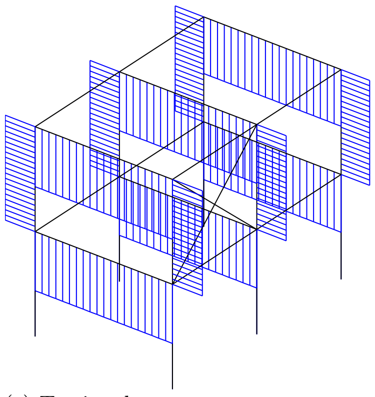
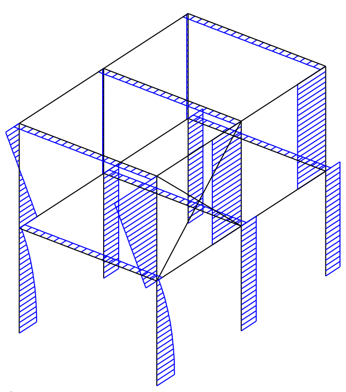

(g) Bending moments around z

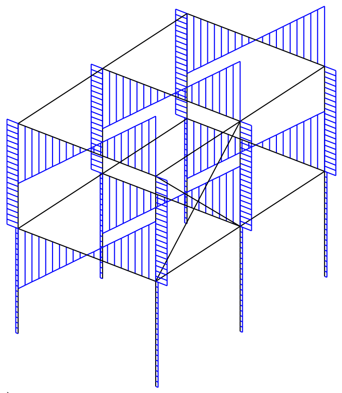
Figure 2.7: Results for load case 2 of example 1.2.

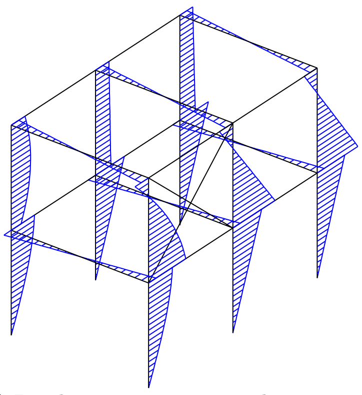

(a) Displacements


(b) Normal forces


(c) Torsional moments


(d) Shear forces along z


(e) Shear forces along y


(f) Bending moments around y

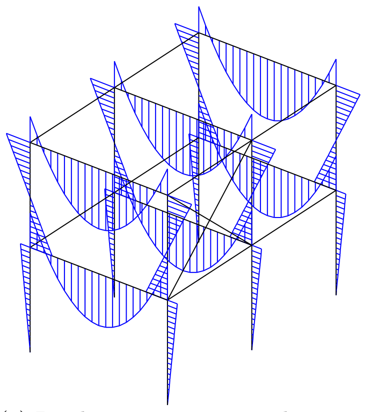
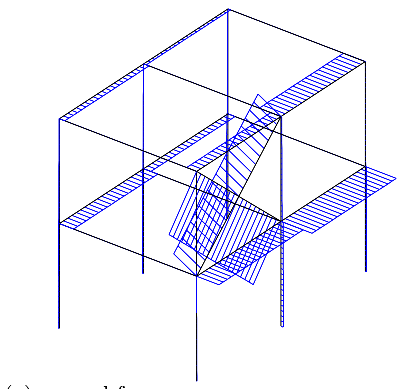
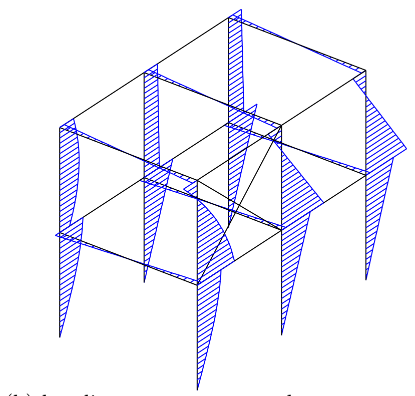
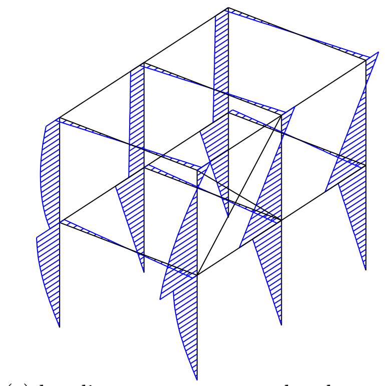
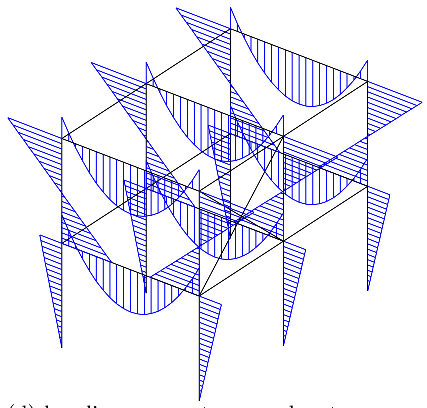

(g) Bending moments around z

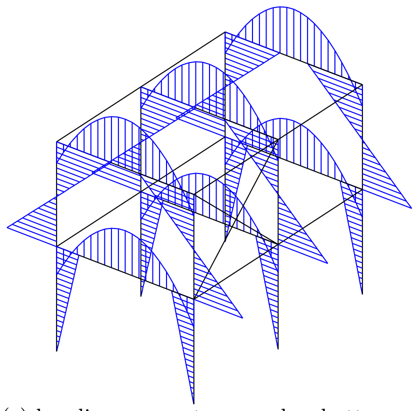
Figure 2.8: Results for load combination ULS of example 1.2.

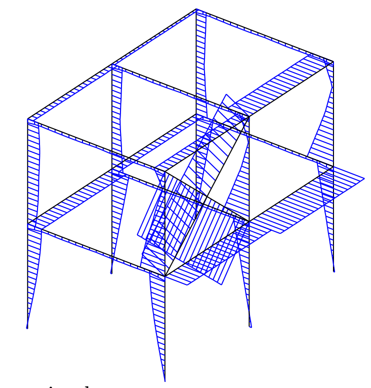

(a) normal forces


(b) bending moments around y: top


(c) bending moments around y: bottom


(d) bending moments around z: top


(e) bending moments around z: bottom


(f) maximal

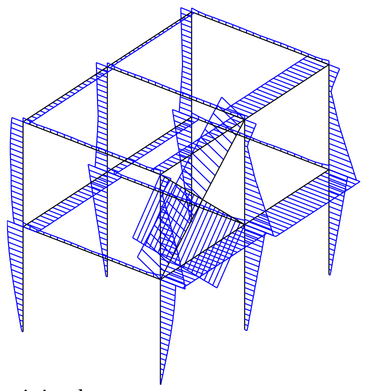

(g) minimal

Figure 2.9: Normal stresses for load combination ULS of example 1.2.

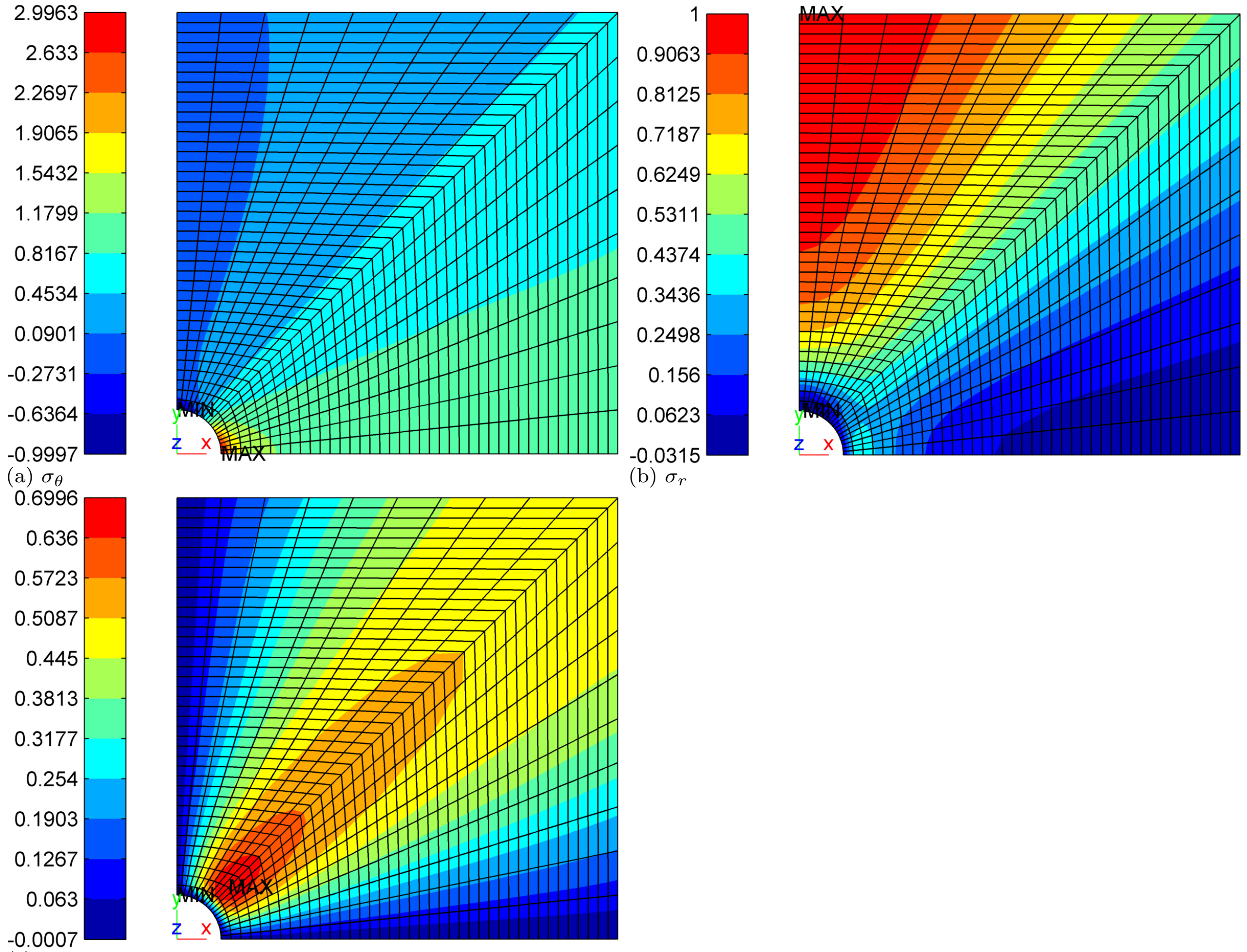
2.3 Example 1.3: static analysis of a plate with a circular hole

This example demonstrates the use of Stabil for the solution of 2D plane strain continuum problems. A thin disk with a circular hole is subjected to uniaxial tension. The stress concentration around the hole is examined:

```
% StaBIL manual
% Example 1.3: static analysis of a plate with a circular hole

% Stress concentration around a circular hole in a disk.
r = 1;                          % Radius of hole [m]
L = 20;                         % Width of plate [m]
p = 1;                          % Uniform pressure [N/m^2]
n = 40;                         % Mesh parameter (nElem = m*n)
m = 20;                         % Mesh parameter
Types = {1 'shell8'};           % {EltTypID EltName}
Sections = [1 1];               % [SecID t]
Materials = [1 200000 0];       % [MatID E nu]



% define lines
Line1 = [r 0 0;L/2 0 0];
Line2 = [L/2 0 0;L/2 L/2 0;0 L/2 0];
Line3 = [0 L/2 0;0 r 0];
Line4 = [r*sin((0:5).'*pi/10) r*cos((0:5).'*pi/10) zeros(6,1)];

[Nodes,Elements,Edge1,Edge2,Edge3,Edge4] = makemesh(Line1,Line2,Line3,Line4,...
               m,n,Types(1,:),Sections(1,1),Materials(1,1),'L2method','linear');

% Check mesh:
figure;
plotnodes(Nodes,'numbering','off');
hold('on')
plotelem(Nodes,Elements,Types,'numbering','off');
title('Nodes and elements');

% Select all dof:
DOF = getdof(Elements,Types);
% Apply boundary conditions:
% - Line1 and Line3: symmetry condition
seldof = [0.03;0.04;0.05;0.06;Edge1+0.02;Edge3+0.01];
DOF = removedof(DOF,seldof);

% Assemble K:
K = asmkm(Nodes,Elements,Types,Sections,Materials,DOF);

% Apply load to upper edge (equivalent nodal forces):
C = selectdof(DOF,Edge2(1:(end-1)/2+1)+0.02);
P = C.'*[1/6; repmat([2/3; 1/3],((length(Edge2)-1)/2-2)/2,1);2/3; 1/6]*p*L/m;
U = K\P;

% Calculate stress in cylindrical global coordinate system
[SeGCS]=elemstress(Nodes,Elements,Types,Sections,Materials,DOF,U,'gcs','cyl');

% Get nodal stress from element solution
[SnGCS] = nodalstress(Nodes,Elements,Types,SeGCS);

% plot results
figure;
plotstresscontourf('sx',Nodes,Elements,Types,SnGCS)
title('\sigma_{r}')
figure;
plotstresscontourf('sy',Nodes,Elements,Types,SnGCS)
title('\sigma_{\theta}')
figure;
```

```
plotstresscontourf('sxy',Nodes,Elements,Types,SnGCS)
title('\sigma_{r\theta}')
```

Figure 2.10



(a) σ_θ

(b) σ_r

(c) $\sigma_{r\theta}$

Figure 2.10: Results for the hole-in-disk problem.

2.4 Example 1.4: static analysis of a barrel vault roof

A barrel vault roof subjected to its self weight is analysed. The curved edges are simply supported and the straight edges are free. Due to symmetry only a quarter of the roof is modelled and symmetry boundary conditions are applied.

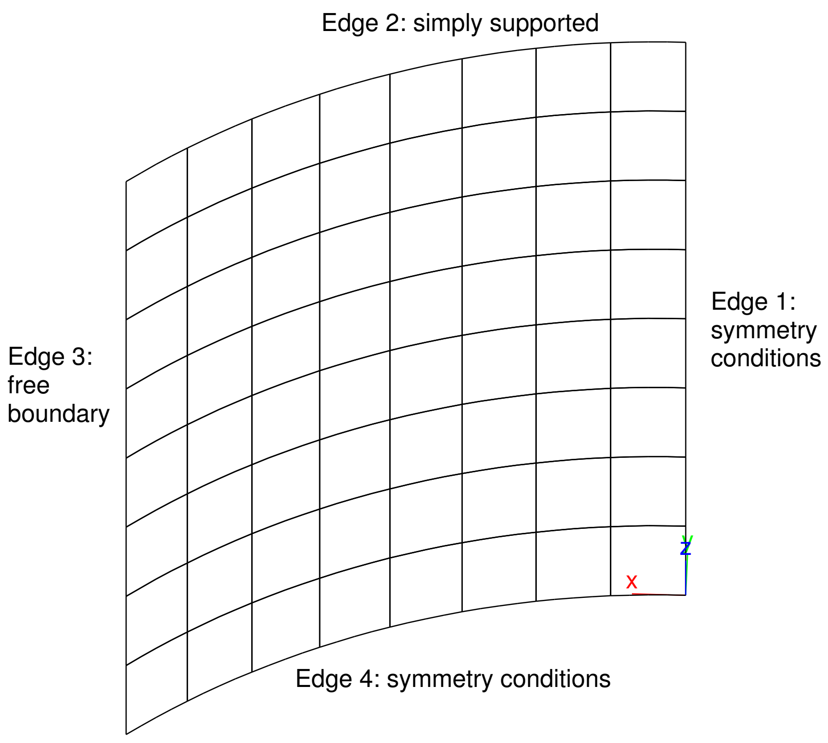
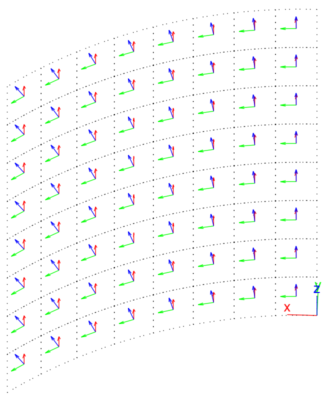


Figure 2.11: 8×8 mesh of a quarter of a cylindrical roof.

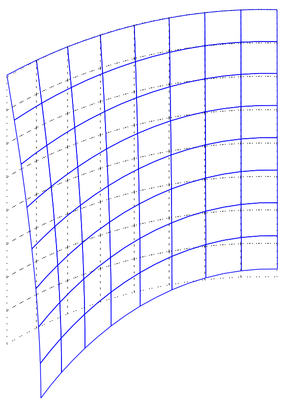
```
% StaBIL manual
% Example 1.4: static analysis of a barrel vault roof

% A BARREL VAULT ROOF SUBJECTED TO ITS SELF WEIGHT
% Reference: COOK, CONCEPTS AND APPL OF F.E.A., 2ND ED., 1981, PP. 284-287.

%% Parameters
R=25;                % Radius of cylindrical roof
L=50;                % Length of cylindrical roof
t=0.25;              % Thickness of roof
theta = 40*pi/180;   % Angle of cylindrical roof
E = 4.32*10^8;       % Youngs modulus
nu = 0;              % Poisson coefficient
rho = 36.7347;       % Density
N = 8;               % Number of elements

%% Mesh

% Lines = [x1 y1 z1;x2 y2 z2;...]
Line1 = [0 0 0;0 0 L/2];
Line2 = [R*sin(theta*(0:0.1:1).') R*cos(theta*(0:0.1:1).')-R L*ones(11,1)/2];
Line3 = [R*sin(theta) R*cos(theta)-R L/2; R*sin(theta) R*cos(theta)-R 0];
Line4 = [R*sin(theta*(1:-0.1:0).') R*cos(theta*(1:-0.1:0).')-R zeros(11,1)];

% Specify element type for mesh
Materials = [1 E nu rho];
Sections = [1 t];
```

```
Types = {1 'shell8'};

% Mesh the area between lines 1,2,3,4 with N * N elements of type shell8,
% section number 1 and material number 1.
[Nodes,Elements,Edge1,Edge2,Edge3,Edge4] = ...
    makemesh(Line1,Line2,Line3,Line4,N,N,Types(1,:),1,1);

% Check mesh:
figure;
plotnodes(Nodes,'numbering','off');
hold('on')
plotelem(Nodes,Elements,Types,'numbering','off');
title('Nodes and elements');

%% Assemble stiffness matrix

% Select all dof:
DOF = getdof(Elements,Types);

% Apply boundary conditions:
% - Line1 and Line4: symmetry condition
% - Line2: simply supported
sdof = [Edge1+0.01;Edge1+0.06;Edge1+0.05;Edge2+0.02;Edge2+0.01;
        Edge2+0.06;Edge4+0.03;Edge4+0.04;Edge4+0.05];
DOF = removedof(DOF,sdof);

% Assemble K:
K = asmkm(Nodes,Elements,Types,Sections,Materials,DOF);

%% Solution

% Apply gravitational acceleration and determine equivalent nodal forces:
DLoads=accel([0 9.8 0],Elements,Types,Sections,Materials);
P=elemloads(DLoads,Nodes,Elements,Types,DOF);


% Solve K * U = P:
U = K\P;

% Plot displacements:
figure;
plotdisp(Nodes,Elements,Types,DOF,U)
title('Displacements')

% Check target displacement:
TP1 = selectdof(DOF,intersect(Edge3,Edge4)+0.02);
Utp1 = TP1*U
ratio_u = -Utp1/0.3016

%% Stress

% Determine element stress in global and local(element) coordinate system:
[SeGCS,SeLCS,vLCS]=elemstress(Nodes,Elements,Types,Sections,Materials,DOF,U);

% print stress:
printstress(Elements,SeGCS)

% plot stress contour:
figure;
plotstresscontour('sx',Nodes,Elements,Types,SeGCS,'location','bot')
title('sx in gcs (element solution)')

% plot filled contours:
figure;
plotstresscontourf('sx',Nodes,Elements,Types,SeGCS,'location','bot')
title('sx in gcs (element solution)')
```

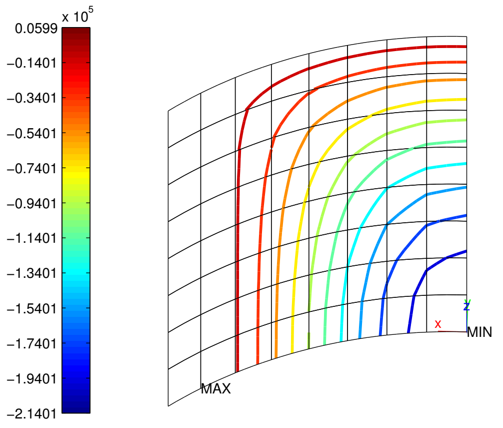
```
figure;
plotstresscontour('sx',Nodes,Elements,Types,SeLCS,'location','bot')
title('sx in lcs')

% plot lcs for shell elements
figure;
plotlcs(Nodes,Elements,Types,vLCS)
title('local coordinate system')

% Calculate nodal solution
% SnGCS:  stress arranged per element
% SnGCS2: stress arranged per node
[SnGCS,SnGCS2]=nodalstress(Nodes,Elements,Types,SeGCS);
[SnLCS,SnLCS2]=nodalstress(Nodes,Elements,Types,SeLCS);
figure;
plotstresscontour('sx',Nodes,Elements,Types,SnGCS,'location','bot');
title('sx in gcs (nodal solution)')

% Stress ratios:

ratio_sz = SnGCS2(intersect(Edge3,Edge4),16)/358420
ratio_st = SnGCS2(intersect(Edge4,Edge1),14)/(-213400)

%% Shell forces

% Shell forces (element solution):
[FeLCS]=elemshellf(Elements,Sections,SeLCS);
figure;
plotshellfcontour('my',Nodes,Elements,Types,FeLCS)
title('my (element solution)')

% Nodal solution:
[FnLCS,FnLCS2]=nodalshellf(Nodes,Elements,Types,FeLCS);
figure;
plotshellfcontour('my',Nodes,Elements,Types,FnLCS)
title('my (nodal solution)')

%% Principal stress

% Principal stresses:
[Spr,Vpr]=principalstress(Elements,SnGCS);
figure;
plotstresscontour('s1',Nodes,Elements,Types,Spr,'location','bot');
title('s1')
% plot principal stresses:
figure;
plotprincstress(Nodes,Elements,Types,Spr,Vpr)
title('principal stresses')
```



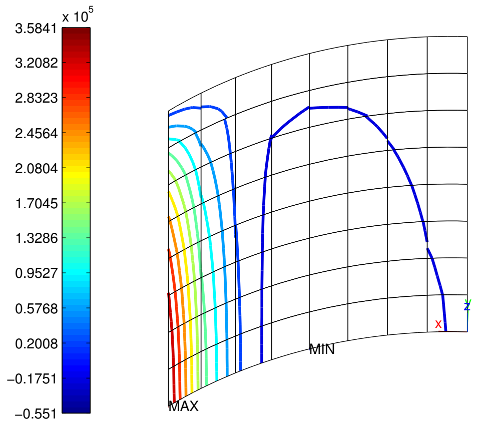
(a) Local coordinate systems



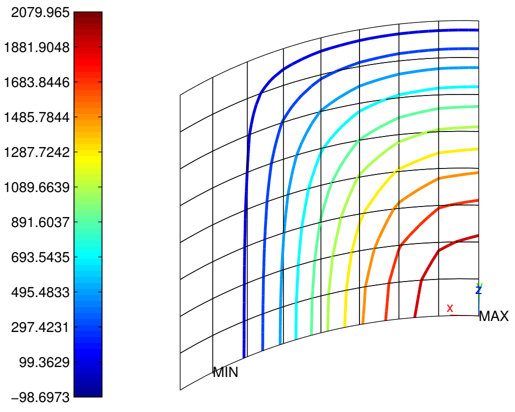
(b) Displacements



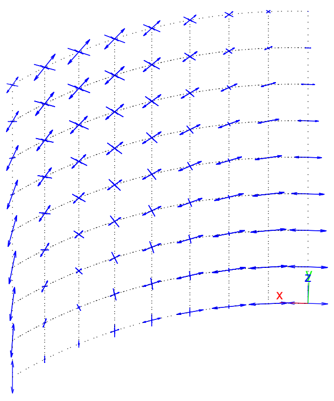
(c) σ_{xx} at bottom of shell in GCS (element solution)



(d) σ_{xx} at bottom of shell in LCS (element solution)



(e) m_y in LCS (nodal solution)



(f) Principal stresses at top of shell

Figure 2.12: Cylindrical roof: results.

Chapter 3

Dynamic analysis of structures

3.1 Example 2.1: dynamic analysis of a frame

The frame that was treated in section 2.1 is reconsidered, computing the eigenmodes and dynamic response of the structure.

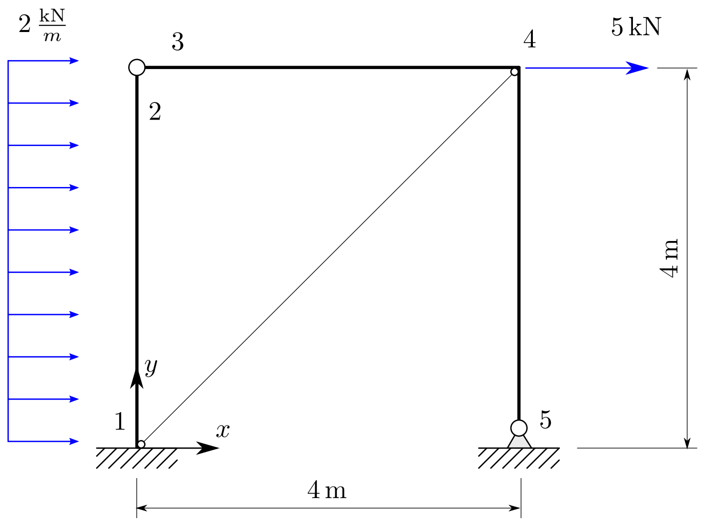


Figure 3.1: Simple frame structure.

3.1.1 Model

```
% StaBIL manual
% Example 2.1: dynamic analysis: model
% Units: m, N

L=4;
H=4;
nElemCable=8;

% Nodes=[NodID X   Y   Z]
Nodes=  [1     0   0   0;
          2     L   0   0];
Nodes=reprow(Nodes,1:2,4,[2 0 H/4 0]);
Nodes=  [Nodes;
          11    0   H   0];
Nodes=reprow(Nodes,11,3,[1 L/4 0 0]);
Nodes=  [Nodes;
          15    1   5   0];          % reference node
Nodes=  [Nodes;
```

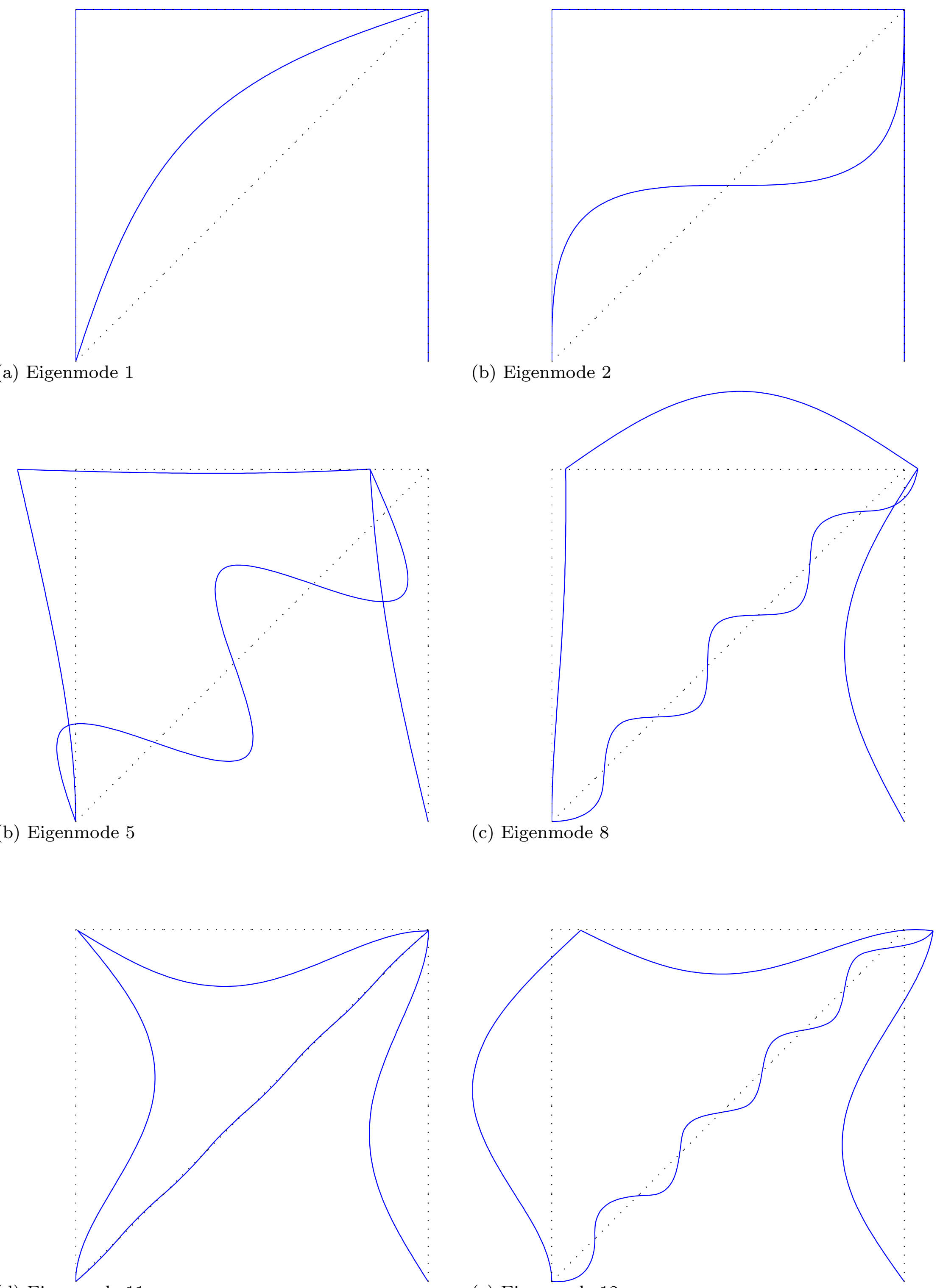
```
         16    0  0  0];
Nodes=reprow(Nodes,16,nElemCable,[1 L/nElemCable H/nElemCable 0]);

% Check the node coordinates as follows:
figure
plotnodes(Nodes);

% Element types -> {EltTypID EltName}
Types=  {1        'beam'};

b=0.10;
h=0.25;
r=0.004;

% Sections=[SecID A      ky   kz   Ixx Iyy Izz]
Sections=  [1     b*h    Inf  Inf  0   0   b*h^3/12;
            2     pi*r^2 Inf  Inf  0   0   pi*r^4/4];

% Materials=[MatID E nu];
Materials=  [1      30e9 0.2 2500;              % concrete
             2     210e9 0.3 7850];             % steel

% Elements=[EltID TypID SecID MatID n1 n2 n3]
Elements=  [1     1     1     1     1  3  15;
                        2     1     1     1     2  4  15];
Elements=reprow(Elements,1:2,3,[2 0 0 0 2 2 0]);
Elements=[ Elements;
           9      1     1     1     11 12 15;
           10     1     1     1     12 13 15;
           11     1     1     1     13 14 15;
           12     1     1     1     14 10 15;
           13     1     2     2     16 17 15];
Elements=reprow(Elements,13,(nElemCable-1),[1 0 0 0 1 1 0]);

% Check node and element definitions as follows:
hold('on');
plotelem(Nodes,Elements,Types);
title('Nodes and elements');

% Degrees of freedom
DOF=getdof(Elements,Types);

% Boundary conditions
seldof=[0.03; 0.04; 0.05; 1.01; 1.02; 1.06; 2.01; 2.02; 16.01; 16.02];

DOF=removedof(DOF,seldof);

% Assembly of stiffness matrix K
[K,M]=asmkm(Nodes,Elements,Types,Sections,Materials,DOF);

% DLoads=[EltID n1globalX n1globalY n1globalZ ...]
DLoads=[1 2000 0 0 2000 0 0;
        3 2000 0 0 2000 0 0;
        5 2000 0 0 2000 0 0;
        7 2000 0 0 2000 0 0];
b=elemloads(DLoads,Nodes,Elements,Types,DOF); % Spatial distribution, nodal (nDOF * 1)

% Constraint equations: Constant=Coef1*DOF1+Coef2*DOF2+ ...
% Constraints=[Constant  Coef1 DOF1  Coef2 DOF2 ...]
Constr=       [0         1     9.01  -1    11.01;
               0         1     9.02  -1    11.02;
               0         1     10.01 -1    (16.01+nElemCable);
               0         1     10.02 -1    (16.02+nElemCable)];

[K,b,M]=addconstr(Constr,DOF,K,b,M);
```

3.1.2 Eigenmodes

```
% StaBIL manual
% Example 2.1: dynamic analysis: eigenvalue problem
% Units: m, N

% Assembly of M and K
tutorialdyna;

% Eigenvalue problem
nMode=12;
[phi,omega]=eigfem(K,M,nMode);

% Display eigenfrequenties
disp('Lowest eigenfrequencies [Hz]');
disp(omega/2/pi);

% Plot eigenmodes
figure;
plotdisp(Nodes,Elements,Types,DOF,phi(:,1),'DispMax','off')
figure;
plotdisp(Nodes,Elements,Types,DOF,phi(:,2),'DispMax','off')
figure;
plotdisp(Nodes,Elements,Types,DOF,phi(:,5),'DispMax','off')
figure;
plotdisp(Nodes,Elements,Types,DOF,phi(:,8),'DispMax','off')
figure;
plotdisp(Nodes,Elements,Types,DOF,phi(:,11),'DispMax','off')
figure;
plotdisp(Nodes,Elements,Types,DOF,phi(:,12),'DispMax','off')

% Animate eigenmodes
figure;
animdisp(Nodes,Elements,Types,DOF,phi(:,1))
title('Eigenmode 1')

figure;
animdisp(Nodes,Elements,Types,DOF,phi(:,2))
title('Eigenmode 2')

figure;
animdisp(Nodes,Elements,Types,DOF,phi(:,5))
title('Eigenmode 5')

figure;
animdisp(Nodes,Elements,Types,DOF,phi(:,8))
title('Eigenmode 8')

figure;
animdisp(Nodes,Elements,Types,DOF,phi(:,11))
title('Eigenmode 11')

figure;
animdisp(Nodes,Elements,Types,DOF,phi(:,12))
title('Eigenmode 12')
```

3.1.3 Modal superposition: time domain: piecewise exact integration

```
% StaBIL manual
% Example 2.1: dynamic analysis: modal superposition: piecewise exact integration
% Units: m, N

% Assembly of M and K
```



(a) Eigenmode 1 (b) Eigenmode 2

(b) Eigenmode 5 (c) Eigenmode 8

(d) Eigenmode 11 (e) Eigenmode 12

Figure 3.2: Results of example 2.1.

```
tutorialdyna;

% Sampling parameters
T=2.5;                              % Time window
dt=0.002;                           % Time step
N=T/dt;                             % Number of samples
```

```
t=[0:N-1]*dt;                    % Time axis

% Eigenvalue analysis
nMode=12;                        % Number of modes to take into account
[phi,omega]=eigfem(K,M,nMode);   % Calculate eigenmodes and eigenfrequencies
xi=0.07;                         % Constant modal damping ratio

% Excitation
bm=phi.'*b;                      % Spatial distribution, modal (nMode * 1)
q=zeros(1,N);                    % Time history (1 * N)
q((t>=0.50) & (t<0.60))=1;       % Time history (1 * N)
pm=bm*q;                         % Modal excitation (nMode * N)

% Modal analysis
x=msupt(omega,xi,t,pm,'zoh');

% Modal displacements -> nodal displacements
u=phi*x;                         % Nodal response (nDOF * N)

% Figures
figure;
plot(t,x);
title('Modal response (piecewise linear exact integration)');
xlabel('Time [s]');
xlim([0 4.1])
ylabel('Displacement [m kg^{0.5}]');
legend([repmat('Mode ',nMode,1) num2str([1:nMode].')]);

figure;
c=selectdof(DOF,[9.01; 13.02; 17.02]);
plot(t,c*u);
title('Nodal response (piecewise linear exact integration)');
xlabel('Time [s]');
xlim([0 4.1])
ylabel('Displacement [m]');
legend('9.01','13.02','17.02');

% Movie
figure;
animdisp(Nodes,Elements,Types,DOF,u);

% Display
disp('Maximum modal response');
disp(max(abs(x),[],2));

disp('Maximum nodal response 9.01 13.02 17.02');
disp(max(abs(c*u),[],2));
```

3.1.4 Modal superposition: transform to frequency domain

```
% StaBIL manual
% Example 2.1: dynamic analysis: direct method: frequency domain
% Units: m, N

% Assembly of M and K
tutorialdyna;

% Sampling parameters
N=2048;                          % Number of samples
dt=0.002;                        % Time step
T=N*dt;                          % Period
F=N/T;                           % Sampling frequency
df=1/T;                          % Frequency resolution
t=[0:N-1]*dt;                    % Time axis
```

```
f=[0:N/2-1]*df;                 % Positive frequencies corresponding to FFT [Hz]
Omega=2*pi*f;                   % Idem [rad/s]

% Eigenvalue analysis
nMode=12;                       % Number of modes to take into account
[phi,omega]=eigfem(K,M,nMode);  % Calculate eigenmodes and eigenfrequencies
xi=0.07;                        % Constant modal damping ratio

% Excitation
bm=phi.'*b;                     % Spatial distribution, modal (nMode * 1)
q=zeros(1,N);                   % Time history (1 * N)
q((t>=0.50) & (t<0.60))=1;      % Time history (1 * N)
Q=fft(q);                       % Frequency content (1 * N)
Q=Q(1:N/2);                     % Frequency content, positive freq (1 * N/2)
Pm=bm*Q;                        % Modal excitation, positive freq (nMode * N/2)

% Modal analysis
[X,H]=msupf(omega,xi,Omega,Pm); % Modal response, positive freq (nMode * N/2)

% F-dom -> t-dom
X=[X, zeros(nMode,1), conj(X(:,end:-1:2))];
x=ifft(X,[],2);                 % Modal response (nMode * N)

% Modal displacements -> nodal displacements
u=phi*x;                        % Nodal response (nDOF * N)

% Figures
figure;
subplot(3,2,1);
plot(t,q,'.-');
xlim([0 4.1])
ylim([0 1.2]);
title('Excitation time history');
xlabel('Time [s]');
ylabel('Force [N/m]');

subplot(3,2,2);
plot(f,abs(Q)/F,'.-');
title('Excitation frequency content');
xlabel('Frequency [Hz]');
ylabel('Force [N/m/Hz]');

subplot(3,2,4);
plot(f,abs(H),'.-');
title('Modal transfer function');
xlabel('Frequency [Hz]');
ylabel('Displacement [m/N]');
legend([repmat('Mode ',nMode,1) num2str([1:nMode].')]);

subplot(3,2,6);
plot(f,abs(X(:,1:N/2))/F,'.-');
title('Modal response');
xlabel('Frequency [Hz]');
ylabel('Displacement [m kg^{0.5}/Hz]');

subplot(3,2,5);
plot(t,x);
title('Modal response (calculation in f-dom)');
xlabel('Time [s]');
xlim([0 4.1])
ylabel('Displacement [m kg^{0.5}]');

figure;
plot(t,x);
title('Modal response (calculation in f-dom)');
xlabel('Time [s]');
```

```
xlim([0 4.1])
ylabel('Displacement [m kg^{0.5}]');
legend([repmat('Mode ',nMode,1) num2str([1:nMode].')]);

figure;
c=selectdof(DOF,[9.01; 13.02; 17.02]);
plot(t,c*u);
title('Nodal response (calculation in f-dom)');
xlabel('Time [s]');
xlim([0 4.1])
ylabel('Displacement [m]');
legend('9.01','13.02','17.02');

% Movie
figure;
animdisp(Nodes,Elements,Types,DOF,u);

% Display
disp('Maximum modal response');
disp(max(abs(x),[],2));

disp('Maximum nodal response 9.01 13.02 17.02');
disp(max(abs(c*u),[],2));
```

3.1.5 Direct time integration

```
% StaBIL manual
% Example 2.1: dynamic analysis: direct time integration: trapezium rule
% Units: m, N

% Assembly of M, K and C
tutorialdyna;
[phi,omega]=eigfem(K,M);             % Calculate eigenmodes and eigenfrequencies
xi=0.07;                             % Damping ratio
nModes=length(K)-size(Constr,1);
C=M.'*phi(:,1:nModes)*diag(2*xi*omega(1:nModes))*phi(:,1:nModes).'*M;
                                     % Modal -> full damping matrix C

% Sampling parameters
T=2.5;                               % Time window
dt=0.002;                            % Time step
N=T/dt;                              % Number of samples
t=[0:N-1]*dt;                        % Time axis

% Excitation
q=zeros(1,N);                        % Time history (1 * N)
q((t>=0.50) & (t<0.60))=1;           % Time history (1 * N)
p=b*q;                               % Nodal excitation (nDOF * N)

% Direct time integration - trapezium rule
alpha=1/4;
delta=1/2;
theta=1;
u=newmark(M,C,K,dt,p,[alpha delta theta]);

% Figures
figure;
c=selectdof(DOF,[9.01; 13.02; 17.02]);
plot(t,c*u);
title(['Nodal response (direct time integration)']);
xlabel('Time [s]');
xlim([0 4.1])
ylabel('Nodal displacements [m]');
legend('9.01','13.02','17.02');
```

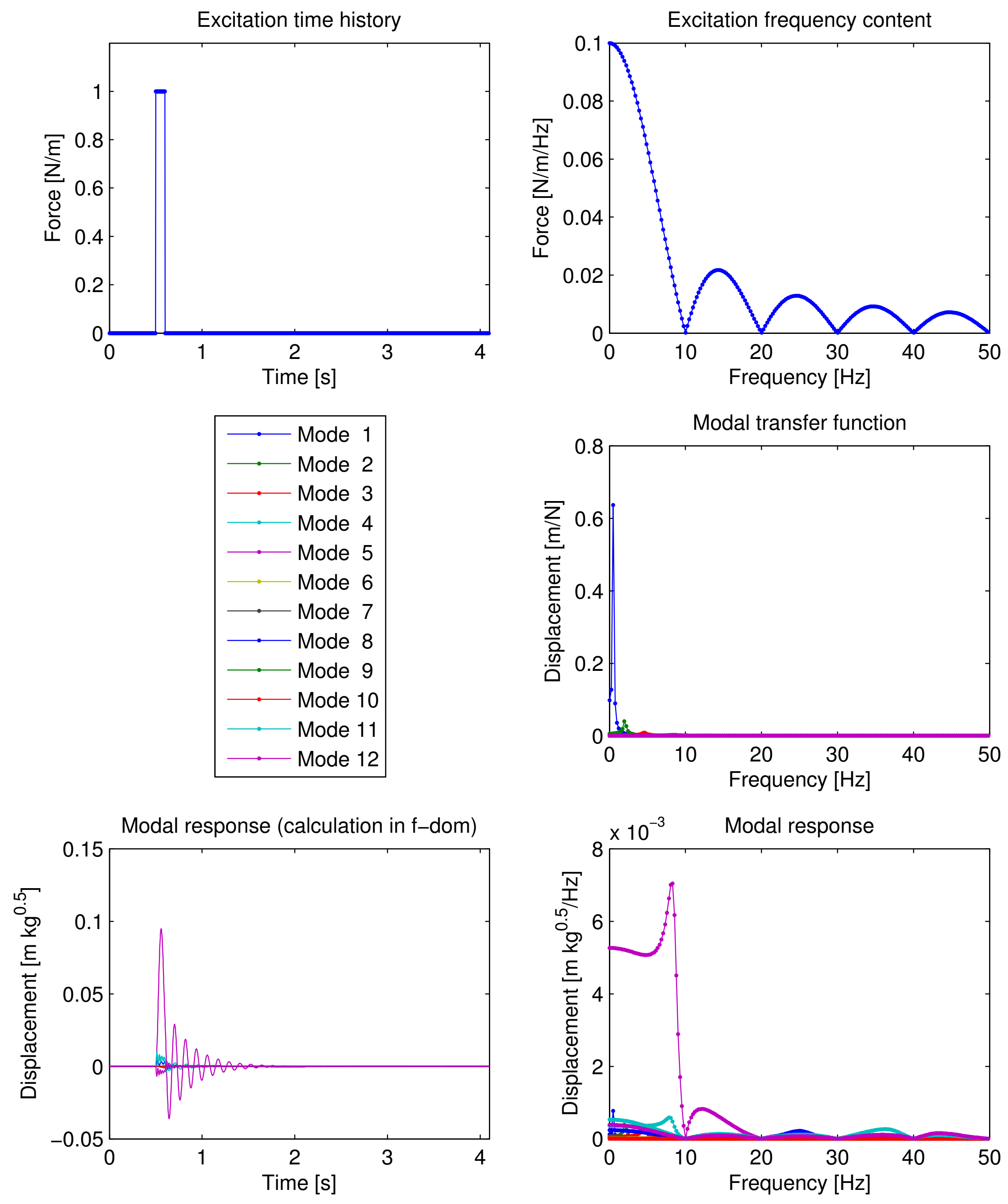
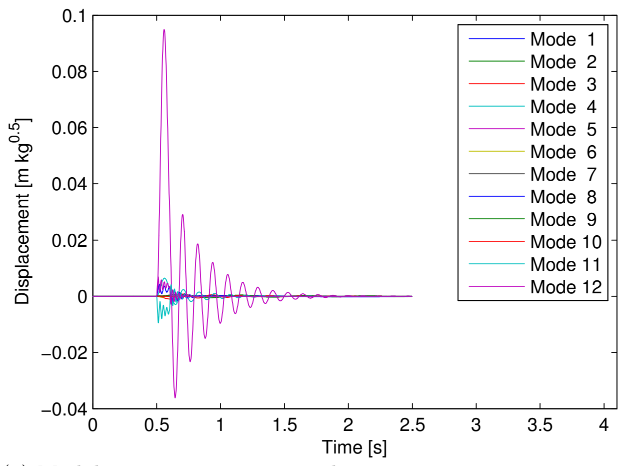


Figure 3.3: Modal superposition in frequency domain.

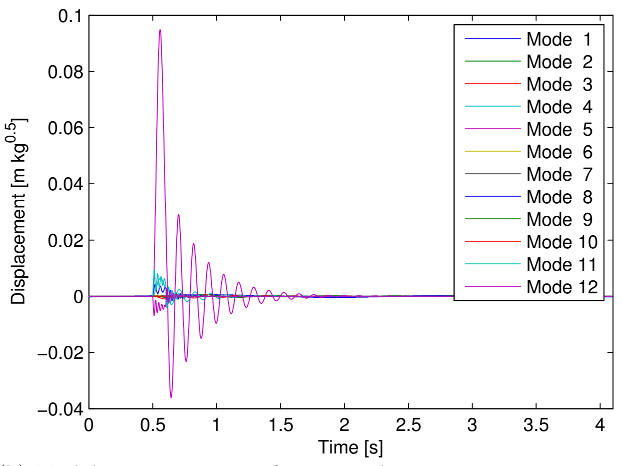
```
% Movie
figure;
animdisp(Nodes,Elements,Types,DOF,u);

% Display
disp('Maximum nodal response 9.01 13.02 17.02');
disp(max(abs(c*u),[],2));
```

3.1.6 Direct solution in the frequency domain

```
% StaBIL manual
% Example 2.1: dynamic analysis: modal superposition: transform to f-dom
```



(a) Modal superposition in time domain



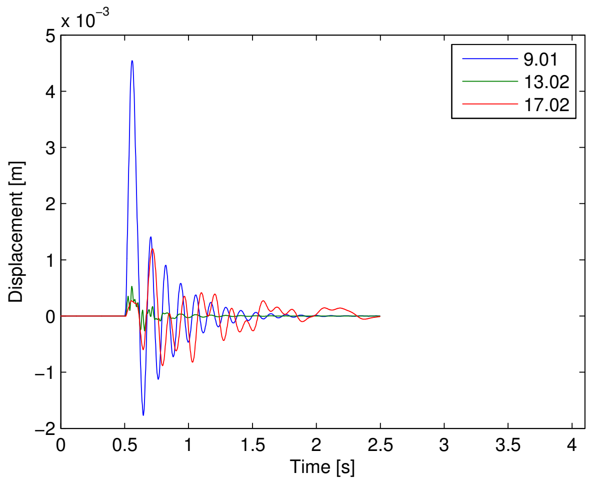
(b) Modal superposition in frequency domain

Figure 3.4: Modal response: comparison.

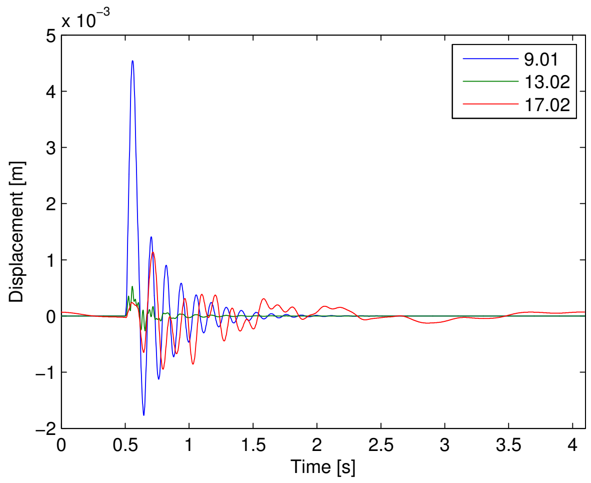
```
% Units: m, N

% Assembly of M, K and C
tutorialdyna;
[phi,omega]=eigfem(K,M);         % Calculate eigenmodes and eigenfrequencies
xi=0.07;                         % Damping ratio
nModes=length(K)-size(Constr,1);
C=M.'*phi(:,1:nModes)*diag(2*xi*omega(1:nModes))*phi(:,1:nModes).'*M;
                                 % Modal -> full damping matrix C

% Sampling parameters
N=2048;                          % Number of samples
dt=0.002;                        % Time step
T=N*dt;                          % Period
F=N/T;                           % Sampling frequency
df=1/T;                          % Frequency resolution
t=[0:N-1]*dt;                    % Time axis
f=[0:N/2-1]*df;                  % Positive frequencies corresponding to FFT [Hz]
Omega=2*pi*f;                    % Idem [rad/s]

% Excitation
q=zeros(1,N);                    % Time history (1 * N)
q((t>=0.50) & (t<0.60))=1;       % Time history (1 * N)
Q=fft(q);                        % Frequency content (1 * N)
Q=Q(1:N/2);                      % Frequency content, positive freq (1 * N/2)
Pd=b*Q;                         % Nodal excitation, positive freq (nDOF * N/2)

% Solution for each frequency
Ud=zeros(size(Pd));
for k=1:N/2
    Kd=-Omega(k)^2*M+Omega(k)*i*C+K;
    Ud(:,k)=Kd\Pd(:,k);
end

% F-dom -> t-dom
Ud=[Ud, zeros(length(K),1), conj(Ud(:,end:-1:2))];
u=ifft(Ud,[],2);                % Nodal response (nDOF * N)

% Figures
figure;
c=selectdof(DOF,[9.01; 13.02; 17.02]);
plot(t,c*u);
title('Nodal response (direct method in f-dom)');
xlabel('Time [s]');
xlim([0 4.1])
```

```
ylabel('Displacement [m]');
legend('9.01','13.02','17.02');

% Movie
figure;
animdisp(Nodes,Elements,Types,DOF,u);

% Display
disp('Maximum nodal response 9.01 13.02 17.02');
disp(max(abs(c*u),[],2));
```

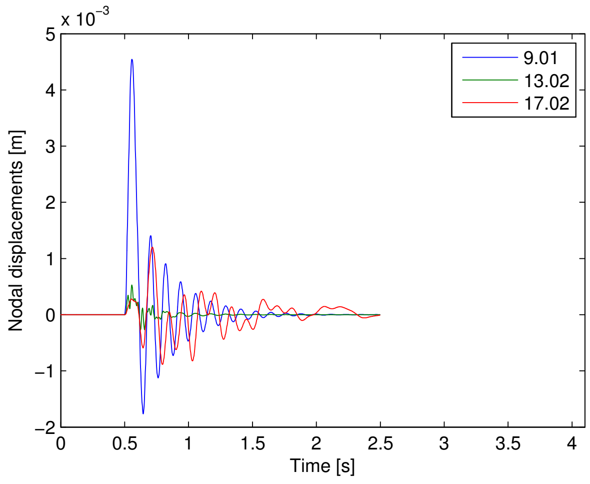
3.1.7 Comparison



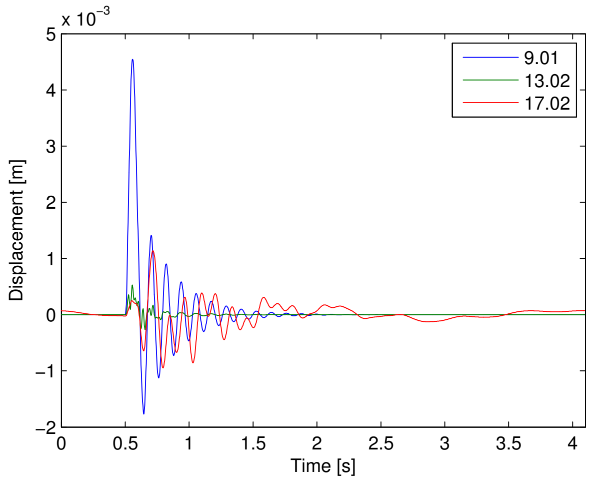
(a) Modal superposition in time domain



(b) Modal superposition in frequency domain



(b) Direct time integration



(c) Direct method in frequency domain

Figure 3.5: Nodal response: comparison.

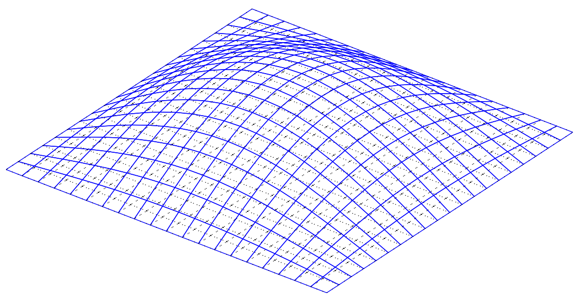
Computational cost in seconds

	Time domain	Freq domain
Modal superposition	0.167	0.147
Direct method	0.213	0.789

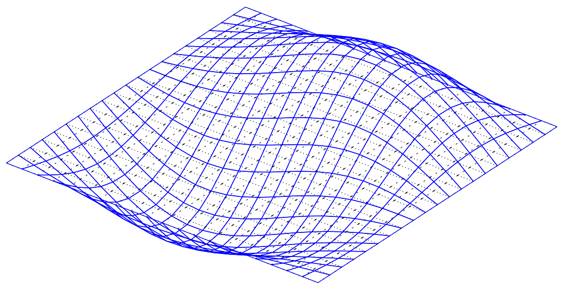
3.2 Example 2.2: dynamic analysis of a plate

In this example the eigenfrequencies of a simply supported rectangular plate are calculated using shell4 and compared with the theoretical solution.

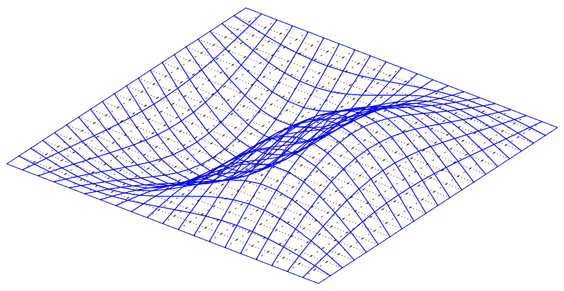
```
% StaBIL manual
% Example 2.2: dynamic analysis of a plate

% dynamic plate problem (dkt element)

% parameters

Lx = 10;                    % length x-direction
Ly = 10;                    % length y-direction
t = 1;                      % thickness plate
E = 200000;                 % E-modulus
nu = 0.3;                   % Poisson-coeff.
rho = 7000;                 % mass density
m = 20;                     % number of elements in x-direction
n = 20;                     % number of elements in y-direction
Types = {1 'shell4'};       % {EltTypID EltName}
Sections = [1 t];           % [SecID t]
Materials = [1 E nu rho];   % [MatID E nu]


% mesh

Line1 = [0 0 0;Lx 0 0];
Line2 = [Lx 0 0;Lx Ly 0];
Line3 = [Lx Ly 0;0 Ly 0];
Line4 = [0 Ly 0;0 0 0];

[Nodes,Elements,Edge1,Edge2,Edge3,Edge4] = makemesh(Line1,Line2,Line3,Line4,n,m,Types(1,:),1,1);
figure;
plotnodes(Nodes);
figure;
plotelem(Nodes,Elements,Types);

% DOFs (simply supported plate)

DOF = getdof(Elements,Types);
sdof = [0.01;0.02;0.06;[Edge1;Edge2;Edge3;Edge4]+0.03;[Edge1;Edge3]+0.05;[Edge2;Edge4]+0.04];
DOF = removedof(DOF,sdof);

% K & M

[K,M] = asmkm(Nodes,Elements,Types,Sections,Materials,DOF);

% eigenmodes

nMode = 10;
[phi,omega] = eigfem(K,M,nMode);
figure;
animdisp(Nodes,Elements,Types,DOF,phi(:,1));

% analytical solution

[mm,nn] = meshgrid((1:m),(1:n));
aomega = sqrt(E*t^2/(12*(1-nu^2)*rho))*((mm*pi/Lx).^2+(nn*pi/Ly).^2);
aomega = reshape(aomega,numel(aomega),1);
aomega = sort(aomega);
ratio = omega./aomega(1:nMode)
```



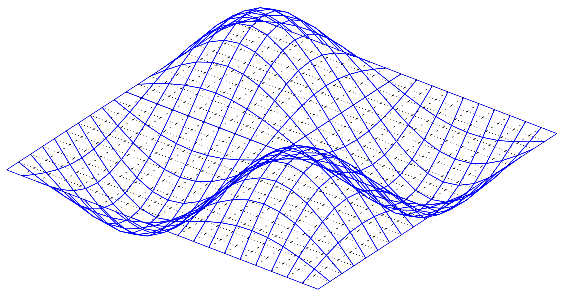
(a) 0.1268 Hz



(b) 0.3170 Hz



(c) 0.3170 Hz



(d) 0.5050 Hz

Figure 3.6: The first four eigenmodes of a thin plate.

f_{FEM}[Hz]	$f_{\text{analytical}}$[Hz]	$f_{\text{FEM}}/f_{\text{analytical}}$
0.1268	0.1270	0.9983
0.3170	0.3176	0.9982
0.3170	0.3176	0.9982
0.5050	0.5082	0.9939
0.6355	0.6352	1.0004
0.6355	0.6352	1.0004
0.8202	0.8258	0.9932
0.8202	0.8258	0.9932
1.0848	1.0799	1.0046
1.0848	1.0799	1.0046

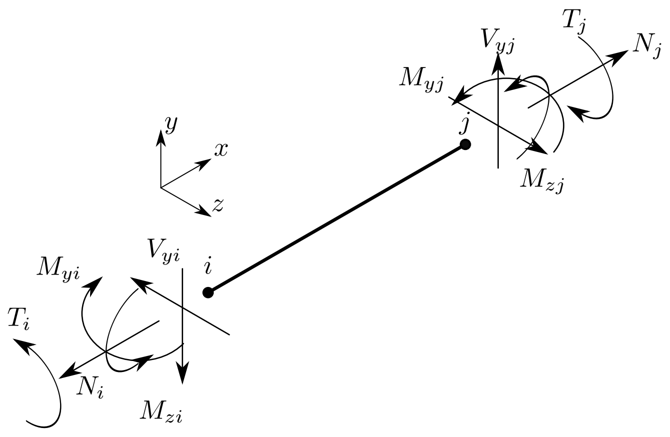
Table 3.1: Eigenfrequencies of a thin plate.

Chapter 4

Element guide

This chapter presents an overview of the element types available in Stabil. The conventions (local coordinate system, nodal connectivity) for each element type are given and reference is made to the Stabil functions related to the element implementation.

beam



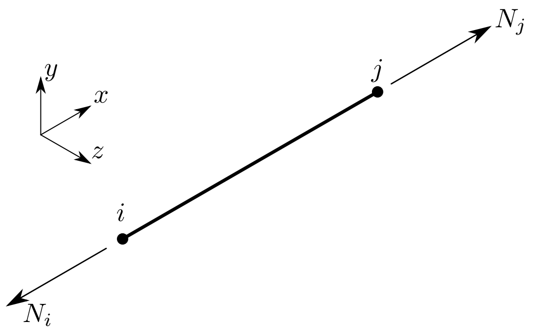
Description

Euler-Bernoulli beam element with a cubic interpolation of the beam deflection.

Functions

`dof_beam`	Element degrees of freedom for a beam element.	p.117
`ke_beam`	Beam element stiffness and mass matrix in global coordinate system.	p.168
`kelcs_beam`	Beam element stiffness and mass matrix in local coordinate system.	p.164
`trans_beam`	Transform coordinate system for a beam element.	p.291
`loads_beam`	Equivalent nodal forces for a beam element in the GCS.	p.192
`loadslcs_beam`	Equivalent nodal forces for a beam element in the LCS.	p.189
`accel_beam`	Compute the distributed loads for a beam due to an acceleration.	p.59
`forces_beam`	Compute the element forces for a beam element.	p.153
`forceslcs_beam`	Compute the element forces for a beam element in the LCS.	p.151
`nelcs_beam`	Shape functions for a beam element.	p.206
`nedloadlcs_beam`	Shape functions for a distributed load on a beam element.	p.205
`coord_beam`	Coordinates of the beam elements for plotting.	p.78
`disp_beam`	Return matrices to compute the displacements of the deformed beams.	p.99
`dispgcs2lcs_beam`	Transform the element displacements to the LCS for a beam.	p.97
`fdiagrgcs_beam`	Return matrices to plot the forces in a beam element.	p.144
`fdiagrlcs_beam`	Force diagram for a beam element in LCS.	p.148
`sdiagrgcs_beam`	Return matrices to plot the stresses in a beam element.	p.253
`sdiagrlcs_beam`	Stress diagram for a beam element in LCS.	p.256

truss



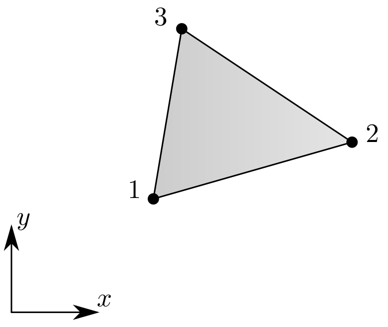
Description

Linear truss element.

Functions

`dof_truss`	Element degrees of freedom for a truss element.	p.134
`ke_truss`	Truss element stiffness and mass matrix in global coordinate system.	p.186
`kelcs_truss`	Truss element stiffness and mass matrix in local coordinate system.	p.167
`trans_truss`	Transform coordinate system for a truss element.	p.297
`loads_truss`	Equivalent nodal forces for a truss element in the GCS.	p.199
`accel_truss`	Compute the distributed loads for a truss due to an acceleration.	p.66
`forces_truss`	Compute the element forces for a truss element.	p.155
`forceslcs_truss`	Compute the element forces for a truss element in the LCS.	p.152
`coord_truss`	Coordinates of the truss elements for plotting.	p.96
`disp_truss`	Return matrices to compute the displacements of the deformed trusses.	p.115
`dispgcs2lcs_truss`	Transform the element displacements to the LCS for a truss.	p.98
`fdiagrgcs_truss`	Return matrices to plot the forces in a truss element.	p.146
`sdiagrgcs_truss`	Return matrices to plot the stresses in a truss element.	p.255

plane3



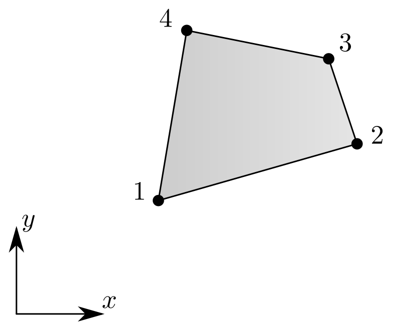
Description

The `plane3` element is a 3-node linear isoparametric 2D triangular plane element, commonly referred to as the "Constant Strain Triangle" (CST).

Functions

`dof_plane3`	Element degrees of freedom for a plane3 element.	p.121
`ke_plane3`	Element stiffness and mass matrix in global coordinate system.	p.173
`coord_plane3`	Coordinates of the plane3 element for plotting.	p.83
`disp_plane3`	Return matrices to compute the displacements of the deformed element.	p.103
`se_plane3`	Compute the element stresses for a plane3 element in the GCS.	p.264
`selcs_plane3`	Compute the element stresses for a plane3 element in the LCS.	p.257
`patch_plane3`	Patch information of the plane3 elements for plotting.	p.213

plane4



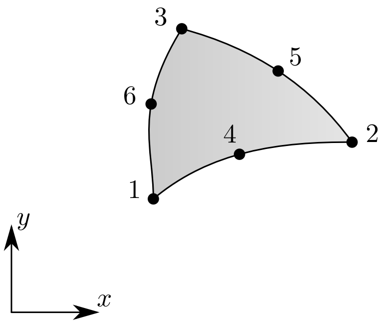
Description

The `plane4` element is a 4-node linear isoparametric 2D quadrilateral plane element.

Functions

`dof_plane4`	Element degrees of freedom for a plane4 element.	p.122
`ke_plane4`	Element stiffness and mass matrix in global coordinate system.	p.174
`coord_plane4`	Coordinates of the plane3 element for plotting.	p.84
`disp_plane4`	Return matrices to compute the displacements of the deformed element.	p.104
`se_plane4`	Compute the element stresses for a plane4 element in the GCS.	p.265
`selcs_plane4`	Compute the element stresses for a plane4 element in the LCS.	p.258
`patch_plane4`	Patch information of the plane4 elements for plotting.	p.214

plane6



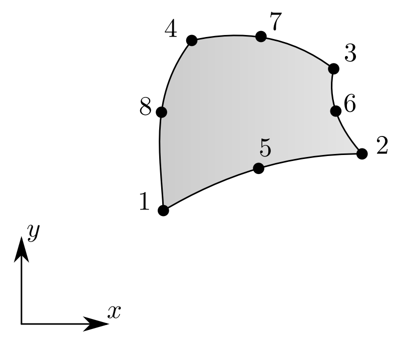
Description

The `plane6` element is a 6-node quadratic isoparametric 2D triangular plane element.

Functions

`dof_plane6`	Element degrees of freedom for a plane6 element.	p.123
`ke_plane6`	Element stiffness and mass matrix in global coordinate system.	p.175
`coord_plane6`	Coordinates of the plane3 element for plotting.	p.85
`disp_plane6`	Return matrices to compute the displacements of the deformed element.	p.105
`se_plane6`	Compute the element stresses for a plane6 element in the GCS.	p.266
`selcs_plane6`	Compute the element stresses for a plane6 element in the LCS.	p.259
`patch_plane6`	Patch information of the plane6 elements for plotting.	p.215

plane8



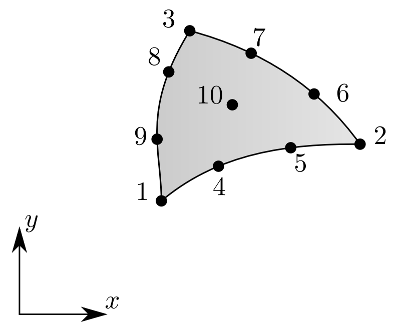
Description

The `plane8` element is a 8-node quadratic isoparametric 2D rectangular plane element.

Functions

`dof_plane8`	Element degrees of freedom for a plane8 element.	p.124
`ke_plane8`	Element stiffness and mass matrix in global coordinate system.	p.176
`coord_plane8`	Coordinates of the plane3 element for plotting.	p.86
`disp_plane8`	Return matrices to compute the displacements of the deformed element.	p.106
`se_plane8`	Compute the element stresses for a plane8 element in the GCS.	p.267
`selcs_plane8`	Compute the element stresses for a plane8 element in the LCS.	p.260
`patch_plane8`	Patch information of the plane8 elements for plotting.	p.216

plane10



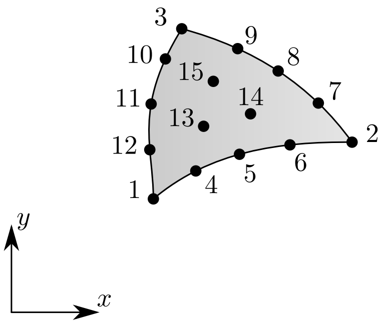
Description

The `plane10` element is a 10-node cubic isoparametric 2D triangular plane element.

Functions

`dof_plane10`	Element degrees of freedom for a plane10 element.	p.119
`ke_plane10`	Element stiffness and mass matrix in global coordinate system.	p.171
`coord_plane10`	Coordinates of the plane3 element for plotting.	p.81
`disp_plane10`	Return matrices to compute the displacements of the deformed element.	p.101

plane15



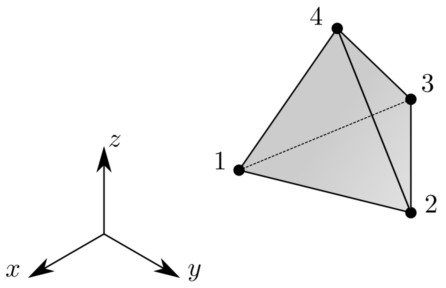
Description

The `plane15` element is a 15-node quartic isoparametric 2D triangular plane element.

Functions

`dof_plane15`	Element degrees of freedom for a plane8 element.	p.120
`ke_plane15`	Element stiffness and mass matrix in global coordinate system.	p.172
`coord_plane15`	Coordinates of the plane3 element for plotting.	p.82
`disp_plane15`	Return matrices to compute the displacements of the deformed element.	p.102

solid4



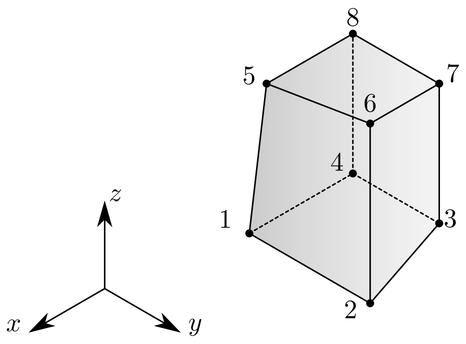
Description

The `solid4` element is a 4-node linear isoparametric 3D solid tetrahedral element.

Functions

`dof_solid4`	Element degrees of freedom for a solid4 element.	p.132
`ke_solid4`	Element stiffness and mass matrix in global coordinate system.	p.184
`coord_solid4`	Coordinates of the plane3 element for plotting.	p.94
`patch_solid4`	Patch information of the solid4 elements for plotting.	p.222

solid8



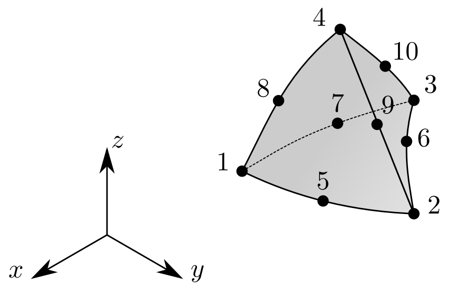
Description

The `solid8` element is a 8-node linear isoparametric 3D solid element.

Functions

`dof_solid8`	Element degrees of freedom for a solid4 element.	p.133
`ke_solid8`	Element stiffness and mass matrix in global coordinate system.	p.185
`coord_solid8`	Coordinates of the plane3 element for plotting.	p.95
`patch_solid8`	Patch information of the solid8 elements for plotting.	p.223

solid10



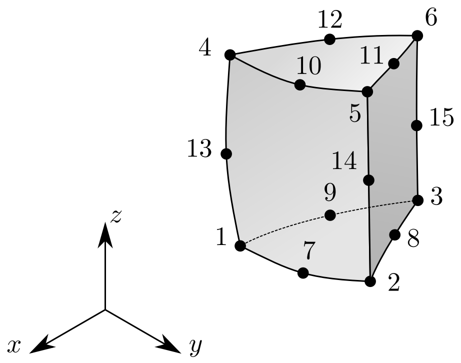
Description

The `solid10` element is a 10-node quadratic isoparametric 3D solid tetrahedral element.

Functions

`dof_solid10`	Element degrees of freedom for a solid10 element.	p.129
`ke_solid10`	Element stiffness and mass matrix in global coordinate system.	p.181
`coord_solid10`	Coordinates of the plane3 element for plotting.	p.91
`disp_solid10`	Return matrices to compute the displacements of the deformed element.	p.111
`patch_solid10`	Patch information of the solid10 elements for plotting.	p.220

solid15

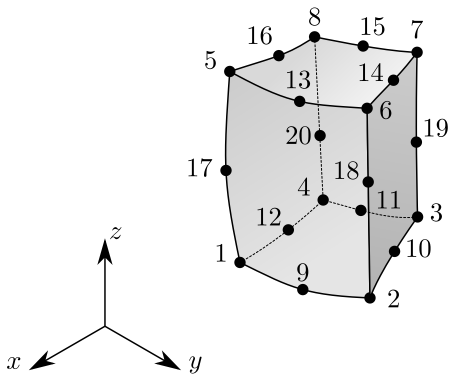


Description

The `solid15` element is a 15-node quadratic isoparametric 3D solid tetrahedral element.

`dof_solid15`	Element degrees of freedom for a solid15 element.	p.130
`ke_solid15`	Element stiffness and mass matrix in global coordinate system.	p.182
`coord_solid15`	Coordinates of the plane3 element for plotting.	p.92
`disp_solid15`	Return matrices to compute the displacements of the deformed element.	p.112

solid20



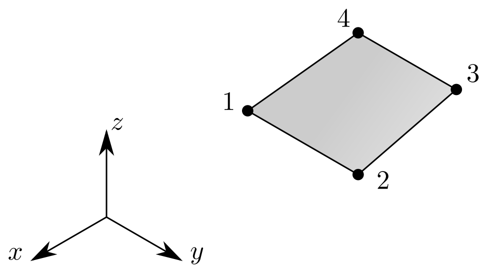
Description

The `solid20` element is a 20-node quadratic isoparametric 3D solid tetrahedral element.

Functions

`dof_solid20`	Element degrees of freedom for a solid20 element.	p.131
`ke_solid20`	Element stiffness and mass matrix in global coordinate system.	p.183
`coord_solid20`	Coordinates of the plane3 element for plotting.	p.93
`disp_solid20`	Return matrices to compute the displacements of the deformed element.	p.113
`patch_solid20`	Patch information of the solid10 elements for plotting.	p.221

shell4



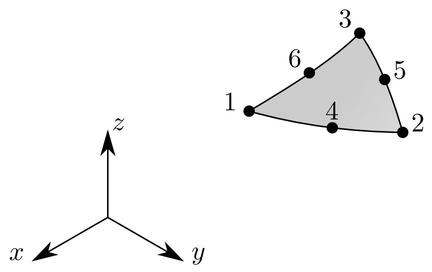
Description

Shell element consisting of a bilinear membrane element and four overlaid DKT triangles for the bending stiffness.

Functions

`dof_shell4`	Element degrees of freedom for a shell4 element.	p.126
`ke_shell4`	Shell4 element stiffness and mass matrix in global coordinate system.	p.178
`kelcs_shell4`	Shell4 element stiffness and mass matrix in local coordinate system.	p.166
`trans_shell4`	Transform coordinate system for a shell4 element.	p.293
`ke_dkt`	DKT plate element stiffness and mass matrix.	p.169
`q_dkt`	Q matrix for a DKT element.	p.247
`sh_qs4`	Shape functions for a quadrilateral serendipity element with 4 nodes.	p.274
`sh_t`	Shape functions for a triangular plate element.	p.276
`se_shell4`	Compute the element stresses for a shell4 element.	p.268
`selcs_shell4`	Compute the element stresses for a shell4 element.	p.261
`accel_shell4`	Compute the distributed loads for a shell due to an acceleration.	p.61
`loads_shell4`	Equivalent nodal forces for a shell4 element in the GCS.	p.194
`loadslcs_shell4`	Equivalent nodal forces for a shell4 element in the LCS.	p.191
`pressure_shell4`	Equivalent nodal forces for a shell4 element in the GCS due to a pressure.	p.239
`coord_shell4`	Coordinates of the shell4 elements for plotting.	p.88
`disp_shell4`	Matrices to compute the displacements of the deformed shell.	p.108
`scontour_shell4`	Matrix to plot contours in a shell4 element.	p.251
`patch_shell4`	Patch information of the shell4 elements for plotting.	p.217
`grid_shell4`	Grid in natural coordinates for mapped meshing.	p.162

shell6

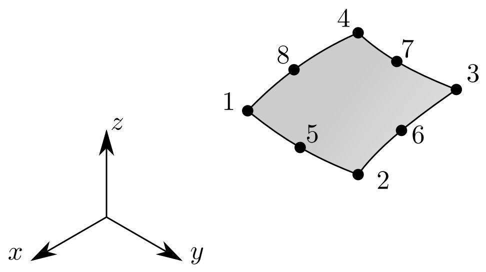


Description

This element is based on chapter 15 of Zienkiewicz [9].

`dof_shell6`	Element degrees of freedom for a shell4 element.	p.127
`ke_shell6`	Shell6 element stiffness and mass matrix in global coordinate system.	p.179
`ke_dkt`	DKT plate element stiffness and mass matrix.	p.169
`q_dkt`	Q matrix for a DKT element.	p.247
`se_shell6`	Compute the element stresses for a shell4 element.	p.269
`accel_shell6`	Compute the distributed loads for a shell due to an acceleration.	p.62
`loads_shell6`	Equivalent nodal forces for a shell6 element in the GCS.	p.195
`pressure_shell6`	Equivalent nodal forces for a shell6 element in the GCS due to a pressure.	p.240
`coord_shell6`	Coordinates of the shell6 elements for plotting.	p.89
`disp_shell6`	Matrices to compute the displacements of the deformed shell.	p.109
`patch_shell6`	Patch information of the shell6 elements for plotting.	p.218

shell8



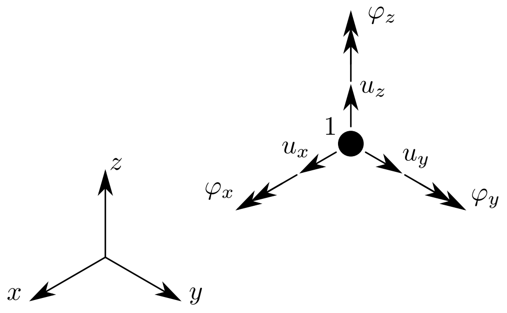
Description

This element is based on chapter 15 of Zienkiewicz [9].

Functions

`dof_shell8`	Element degrees of freedom for a shell8 element.	p.128
`ke_shell8`	shell8 element stiffness and mass matrix in global coordinate system.	p.180
`sh_qs8`	Shape functions for a quadrilateral serendipity element with 8 nodes.	p.275
`b_shell8`	B matrix for a shell8 element in global coordinate system.	p.73
`se_shell8`	Compute the element stresses for a shell8 element.	p.270
`accel_shell8`	Compute the distributed loads for a shell due to an acceleration.	p.63
`loads_shell8`	Equivalent nodal forces for a shell8 element in the GCS.	p.196
`pressure_shell8`	Equivalent nodal forces for a shell8 element in the GCS due to a pressure.	p.241
`coord_shell8`	Coordinates of the shell8 elements for plotting.	p.90
`disp_shell8`	Matrices to compute the displacements of the deformed shell.	p.110
`scontour_shell8`	Matrix to plot contours in a shell8 element.	p.252
`patch_shell8`	Patch information of the shell8 elements for plotting.	p.219
`grid_shell8`	Grid in natural coordinates for mapped meshing.	p.163

mass



Description

Single point concentrated mass element with 6 degrees of freedom.

Functions

`dof_mass`	Element degrees of freedom for a mass element.	p.118
`ke_mass`	Element stiffness and mass matrix in global coordinate system.	p.170
`coord_mass`	Coordinates of the mass element for plotting.	p.80
`disp_mass`	Return matrices to compute the displacements of the deformed element.	p.100
`patch_mass`	Patch information of the mass elements for plotting.	p.212

Chapter 5

Functions — By category

5.1 General functions

`getdof`	Get the vector with the degrees of freedom of the model.	p.159
`asmkm`	Assemble stiffness and mass matrix.	p.70
`removedof`	Remove DOF with Dirichlet boundary conditions equal to zero.	p.249
`addconstr`	Add constraint equations to the stiffness matrix and load vector.	p.67
`tconstr`	Return matrices to apply constraint equations.	p.288
`nodalvalues`	Construct a vector with the values at the selected DOF.	p.210
`elemloads`	Equivalent nodal forces.	p.138
`accel`	Compute the distributed loads due to an acceleration.	p.58
`elemforces`	Compute the element forces.	p.137
`elemdisp`	Select the element displacements from the global displacement vector.	p.136
`selectdof`	Select degrees of freedom.	p.262
`unselectdof`	Unselect degrees of freedom.	p.299
`selectnode`	Select nodes by location.	p.263
`reprow`	Replicate rows from a matrix.	p.250
`multdloads`	Combine distributed loads.	p.204

5.2 Postprocessing

`plotnodes`	Plot the nodes.	p.232
`plotelem`	Plot the elements.	p.227
`plotdisp`	Plot the displacements.	p.225
`plotforc`	Plot the forces.	p.229
`plotlcs`	Plot the local element coordinate systems.	p.231
`plotstress`	Plot the stresses.	p.236
`plotstresscontour`	Plot stress contours.	p.237
`plotstresscontourf`	Plot filled contours of stresses.	p.238
`animdisp`	Animate the displacements.	p.68
`getmovie`	Get the movie from a figure where an animation has been played.	p.160
`printdisp`	Display the displacements in the command window.	p.243
`printforc`	Display the forces in the command window.	p.244

5.3 Dynamics

`eigfem`	Compute eigenmodes and eigenfrequencies.	p.135
`msupt`	Modal superposition in the time domain.	p.203
`msupf`	Modal superposition in the frequency domain.	p.202
`cdiff`	Direct time integration for dynamic systems - central diff. method.	p.74
`newmark`	Direct time integration for dynamic systems - Newmark method	p.207
`wilson`	Direct time integration for dynamic systems - Wilson-theta method	p.302

5.4 General shell functions

`elempressure`	Equivalent nodal forces for pressure $\perp$ shell surface.	p.139
`gaussq`	Gauss points for 2D numerical integration.	p.156
`nodalshellf`	Compute the nodal shell forces/moments from element solution.	p.208
`nodalstress`	Compute the nodal stresses from element solution.	p.209
`plotprincstress`	Plot the principal stresses.	p.233
`plotshellfcontour`	Plot contour lines of forces/moments per unit length.	p.234
`plotstresscontour`	Plot stress contour lines.	p.237
`plotshellfcontourf`	Plot filled contours of shell forces.	p.235
`principalstress`	Compute the principal stresses and directions.	p.242
`printshellf`	Display forces/moments in command window (shell elements).	p.245
`printstress`	Display stresses in command window (shell elements).	p.246

Chapter 6

Functions — Alphabetical list

accel

```
ACCEL   Compute the distributed loads due to an acceleration.

   DLoads=accel(Accelxyz,Nodes,Elements,Types,Sections,Materials)
   computes the distributed loads due to an acceleration.
   In order to simulate gravity, accelerate the structure in the direction
   opposite to gravity.

   Accelxyz   Acceleration            [Ax Ay Az] (1 * 3)
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }
   Sections   Section definitions     [SecID SecProp1 SecProp2 ...]
   Materials  Material definitions    [MatID MatProp1 MatProp2 ... ]
   DLoads     Distributed loads       [EltID n1globalX n1globalY n1globalZ ...]

   See also ELEMLOADS, ACCEL_BEAM, ACCEL_TRUSS.
```

accel_beam

```
ACCEL_BEAM   Compute the distributed loads for a beam due to an acceleration.

   DLoads=accel_beam(Accelxyz,[],Elements,Sections,Materials,Options)
   computes the distributed loads for a beam due to an acceleration.
   In order to simulate gravity, accelerate the structure in the direction
   opposite to gravity.

   Accelxyz   Acceleration            [Ax Ay Az] (1 * 3)
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Sections   Section definitions     [SecID SecProp1 SecProp2 ...]
   Materials  Material definitions    [MatID MatProp1 MatProp2 ... ]
   Options    Element options            {Option1 Option2 ...}
   DLoads     Distributed loads       [EltID n1globalX n1globalY n1globalZ ...]

   See also ACCEL, ACCEL_TRUSS.
```

accel_shell2

```
ACCEL_SHELL2   Compute the distributed loads for a SHELL2 element due to an acceleration.

   DLoads=accel_shell2(Accelxyz,Nodes,Elements,Sections,Materials,Options)
   computes the distributed loads for a SHELL2 element due to an acceleration.
   In order to simulate gravity, accelerate the structure in the direction
   opposite to gravity.

   Accelxyz   Acceleration            [Ax Ay Az] (1 * 3)
   Nodes      Node definitions        [NodeID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Sections   Section definitions     [SecID SecProp1 SecProp2 ...]
   Materials  Material definitions    [MatID MatProp1 MatProp2 ... ]
   Options    Element options            {Option1 Option2 ...}
   DLoads     Distributed loads       [EltID n1globalX n1globalY n1globalZ ...]
```

accel_shell4

```
ACCEL_SHELL4   Compute the distributed loads for shell4 elements due to an
               acceleration.

   DLoads = accel_shell4(Accelxyz,[],Elements,Sections,Materials,Options)
   computes the distributed loads for shell4 elements due to an acceleration.
   In order to simulate gravity, accelerate the structure in the direction
   opposite to gravity.

   Accelxyz   Acceleration            [Ax Ay Az] (1 * 3)
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Sections   Section definitions     [SecID SecProp1 SecProp2 ...]
   Materials  Material definitions    [MatID MatProp1 MatProp2 ... ]
   Options    Element options            {Option1 Option2 ...}
   DLoads     Distributed loads       [EltID n1globalX n1globalY n1globalZ ...]

   See also ACCEL, ACCEL_TRUSS.
```

accel_shell6

```
ACCEL_SHELL6   Compute the distributed loads for shell6 elements due to an acceleration.

   DLoads = accel_shell6(Accelxyz,[],Elements,Sections,Materials,Options)
   computes the distributed loads for shell8 elements due to an acceleration.
   In order to simulate gravity, accelerate the structure in the direction
   opposite to gravity.

   Accelxyz   Acceleration            [Ax Ay Az] (1 * 3)
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Sections   Section definitions     [SecID SecProp1 SecProp2 ...]
   Materials  Material definitions    [MatID MatProp1 MatProp2 ... ]
   Options    Element options struct. Fields:
              -MatType: 'isotropic' (default) or 'orthotropic'
   DLoads     Distributed loads       [EltID n1globalX n1globalY n1globalZ ...]

   See also ACCEL, ACCEL_TRUSS.
```

accel_shell8

```
ACCEL_SHELL8   Compute the distributed loads for shell8 elements due to an acceleration.

   DLoads = accel_shell8(Accelxyz,[],Elements,Sections,Materials,Options)
   computes the distributed loads for shell8 elements due to an acceleration.
   In order to simulate gravity, accelerate the structure in the direction
   opposite to gravity.

   Accelxyz   Acceleration            [Ax Ay Az] (1 * 3)
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Sections   Section definitions     [SecID SecProp1 SecProp2 ...]
   Materials  Material definitions    [MatID MatProp1 MatProp2 ... ]
   Options    Element options struct. Fields:
              -MatType: 'isotropic' (default) or 'orthotropic'
   DLoads     Distributed loads       [EltID n1globalX n1globalY n1globalZ ...]

   See also ACCEL, ACCEL_TRUSS.
```

accel_solid20

```
ACCEL_SOLID20  Compute the distributed loads for solid20 elements due to an
               acceleration.

   DLoads = accel_solid20(Accelxyz,[],Elements,Sections,Materials,Options)
   computes the distributed loads for solid20 elements due to an acceleration.
   In order to simulate gravity, accelerate the structure in the direction
   opposite to gravity.

   Accelxyz   Acceleration            [Ax Ay Az] (1 * 3)
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Sections   Section definitions     [SecID SecProp1 SecProp2 ...]
   Materials  Material definitions    [MatID MatProp1 MatProp2 ... ]
   Options    Element options            {Option1 Option2 ...}
   DLoads     Distributed loads       [EltID n1globalX n1globalY n1globalZ ...]

   See also ACCEL, ACCEL_TRUSS.
```

accel_solid8

```
ACCEL_SOLID8   Compute the distributed loads for solid8 elements due to an
               acceleration.

   DLoads = accel_solid8(Accelxyz,[],Elements,Sections,Materials,Options)
   computes the distributed loads for solid8 elements due to an acceleration.
   In order to simulate gravity, accelerate the structure in the direction
   opposite to gravity.

   Accelxyz   Acceleration            [Ax Ay Az] (1 * 3)
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Sections   Section definitions     [SecID SecProp1 SecProp2 ...]
   Materials  Material definitions    [MatID MatProp1 MatProp2 ... ]
   Options    Element options            {Option1 Option2 ...}
   DLoads     Distributed loads       [EltID n1globalX n1globalY n1globalZ ...]

   See also ACCEL, ACCEL_TRUSS.
```

accel_truss

```
ACCEL_TRUSS   Compute the distributed loads for a truss due to an acceleration.

   DLoads=accel_truss(Accelxyz,[],Elements,Sections,Materials,Options)
   computes the distributed loads for a truss due to an acceleration.
   In order to simulate gravity, accelerate the structure in the direction
   opposite to gravity.

   Accelxyz   Acceleration             [Ax Ay Az] (1 * 3)
   Elements   Element definitions      [EltID TypID SecID MatID n1 n2 ...]
   Sections   Section definitions      [SecID SecProp1 SecProp2 ...]
   Materials  Material definitions     [MatID MatProp1 MatProp2 ... ]
   Options    Element options             {Option1 Option2 ...}
   DLoads     Distributed loads        [EltID n1globalX n1globalY n1globalZ ...]

   See also ACCEL, ACCEL_BEAM.
```

addconstr

```
ADDCONSTR   Add constraint equations to the stiffness matrix and load vector.

     [K,F]=addconstr(Constr,DOF,K,F)
   [K,F,M]=addconstr(Constr,DOF,K,[],M)
   [K,F,M]=addconstr(Constr,DOF,K,F,M)
   modifies the stiffness matrix, the mass matrix and the load vector according
   to the applied constraint equations. The dimensions of the stiffness matrix,
   the mass matrix and the load vector are kept the same. The resulting
   stiffness and mass matrix are not symmetric anymore. This function can be
   used as well to apply imposed displacements.

   Constr     Constraint equation:
               Constant=CoefS*SlaveDOF+CoefM1*MasterDOF1+CoefM2*MasterDOF2+...
              [Constant CoefS SlaveDOF CoefM1 MasterDOF1 CoefM2 MasterDOF2 ...]
   DOF        Degrees of freedom  (nDOF * 1)
   K          Stiffness matrix (nDOF * nDOF)
   F          Load vector  (nDOF * nSteps)
   M          Mass matrix (nDOF * nDOF)
```

animdisp

```
ANIMDISP   Animate the displacements.

   DispScal=animdisp(Nodes,Elements,Types,DOF,U)
   animates the displacements.

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }
   DOF        Degrees of freedom  (nDOF * 1)
   U          Displacements (nDOF * nSteps)
   DispScal   Displacement scaling

   ANIMDISP(...,ParamName,ParamValue) sets the value of the specified
   parameters.  The following parameters can be specified:
   'DispScal'     Displacement scaling.  Default: 'auto'.
   'Handle'       Plots in the axis with this handle.  Default: current axis.
   'Fps'          Frames per second.  Default: 12.
   'CreateMovie'  Saves the movie in the userdata of the axis of the figure.
                  Use getmovie to get the movie from the axis. Default: 'off'.
   'Counter'      Displays the number of the frame for transient displacements.
                  Default: 'on'.
   Additional parameters are redirected to the PLOTDISP function which plots
   the individual frames of the movie.

   See also GETMOVIE, PLOTDISP.
```

argdimchk

```
ARGDIMCHK   Validate input argument dimensions.

   msg = ARGDIMCHK(arg1,size1,arg2,size2,...) returns an appropriate error
   message if the dimensions of arg1,arg2,... do not comply with size1,size2,...
   respectively.  If they do comply, an empty matrix is returned.
   size1,size2,... are cell arrays consisting of at least 2 cells.  Each cell
   corresponds to a dimension of arg1,arg2,... and contains a number (to
   constrain the dimension explicitly) or a string (to constrain the dimension
   implicitly).  E.g. the expression:

   ERROR(ARGDIMCHK( ...
     omega,{'nMode',   1      }, ...
     Phi,  {'nDof',    'nMode' }, ...
     xi,   {'nMode',   1      }, ...
     b,    {'nDof',    1      }, ...
     q,    { 1,        'nOmega'}, ...
     Omega,{ 1,        'nOmega'}, ...
     c,    {'nSelDof','nDof'  }));

   checks if omega,Phi,xi,b,q,Omega,c are all 2-dimensional variables and if
   SIZE(omega,2)==1           SIZE(Phi,2)==SIZE(omega,1)
   SIZE(xi,1)==SIZE(omega,1)  SIZE(xi,2)==1
   SIZE(b,1)==SIZE(Phi,1)     SIZE(b,2)==1
   SIZE(q,1)==1               SIZE(Omega,1)==1
   SIZE(Omega,2)==SIZE(q,2)   SIZE(c,2)==SIZE(Phi,1)
   If not, an appropriate error message is shown.
```

asmkm

```
ASMKM   Assemble stiffness and mass matrix.

   [K,M] = ASMKM(Nodes,Elements,Types,Sections,Materials,DOF)
    K    = ASMKM(Nodes,Elements,Types,Sections,Materials,DOF)
    K    = ASMKM(Nodes,Elements,Types,Sections,Materials)
   assembles the stiffness and the mass matrix using the finite element method.

   [K,~,dKdx] = ASMKM(Nodes,Elements,Types,Sections,Materials,DOF,dNodesdx,dSectionsdx)
   assembles the stiffness matrix using the finite element method and
   additionally computes the derivatives of the stiffness matrix with
   respect to the design variables x. The derivatives of the mass matrix
   have not yet been implemented.

   Nodes       Node definitions          [NodID x y z]
   Elements    Element definitions       [EltID TypID SecID MatID n1 n2 ...]
   Types       Element type definitions  {TypID EltName Option1 ...}
   Sections    Section definitions       [SecID SecProp1 SecProp2 ...]
   Materials   Material definitions      [MatID MatProp1 MatProp2 ...]
   DOF         Degrees of freedom              (nDOF * 1)
   dNodesdx    Node definitions derivatives    (SIZE(Nodes) * nVar)
   dSectionsdx Section definitions derivatives (SIZE(Sections) * nVar)
   K           Stiffness matrix                (nDOF * nDOF)
   M           Mass matrix                     (nDOF * nDOF)
   dKdx        Stiffness matrix derivatives    (CELL(nVar,1))

   See also KE_TRUSS, KE_BEAM.
```

be_plane3

```
e_plane3 is a function.
   [BeGCS] = be_plane3(Node, Section, Material, UeGCS, Options, gcs)
```

b_shell6

```
B_SHELL6   b matrix for a shell6 element in global coordinate system.

   [Bg,J] = b_shell6(Ni,dN_dxi,dN_deta,zetar,Node,h,v1i,v2i,v3i) returns
   the element b matrix in the global coordinate system and the Jacobian
   of the parametric transformation. Both are evaluated in the natural
   coordinates (xi,eta and zetar) which were used to calculate
   Ni,dN_dxi,dN_deta and zetar.

   Node       Node definitions                            [x y z] (6 * 3)
              Nodes should have the following order:
              3
              | \
              6  5
              |    \
              1--4--2
   Ni         Shape functions for quadratic serendipity element    (6 * 1)
   dN_dxi     first derivatives of shape functions Ni             (6 * 1)
   dN_deta    first derivatives of shape functions Ni             (6 * 1)
   h          scalar or vector containing thickness     scalar or (6 * 1)
   v(1,2,3)i  components of the local coordinate system in node i  (6 * 3)
   d          Nodal offset from shell mid plane      scalar (default = 0)
   Bg         b matrix of shell8 element                          (6 * 36)
   J          Jacobian of the parametric transformation           (3 * 3)

   See also SE_SHELL8, KE_SHELL8.
```

b_shell8

```
B_SHELL8   b matrix for a shell8 element in global coordinate system.

   [Bg,J] = b_shell8(Ni,dN_dxi,dN_deta,zetar,Node,h,v1i,v2i,v3i) returns
   the element b matrix in the global coordinate system and the Jacobian
   of the parametric transformation. Both are evaluated in the natural
   coordinates (xi,eta and zetar) which were used to calculate
   Ni,dN_dxi,dN_deta and zetar.

   Node        Node definitions                             [x y z] (8 * 3)
               Nodes should have the following order:
               4----7----3
               |         |
               8         6
               |         |
               1----5----2
   Ni          Shape functions for quadratic serendipity element    (8 * 1)
   dN_dxi      first derivatives of shape functions Ni              (8 * 1)
   dN_deta     first derivatives of shape functions Ni              (8 * 1)
   h           scalar or vector containing thickness      scalar or (8 * 1)
   v(1,2,3)i   components of the local coordinate system in node i  (8 * 3)
   d           Nodal offset from shell mid plane       scalar (default = 0)
   Bg          b matrix of shell8 element                          (6 * 48)
   J           Jacobian of the parametric transformation            (3 * 3)

   See also SE_SHELL8, KE_SHELL8.
```

cdiff

```
CDIFF   Direct time integration for dynamic systems - central diff. method.
   [u,t] = CDIFF(M,C,K,dt,p,u0,u1) applies the central difference method for
   the calculation of the nodal displacements u of the dynamic system with
   the system matrices M, C and K due to the excitation p.

   M    Mass matrix (nDof * nDof)
   C    Damping matrix (nDof * nDof)
   K    Stiffness matrix (nDof * nDof)
   dt   Time step of the integration scheme (1 * 1).  Should be small enough
        to ensure the stability and the precision of the integration scheme.
   p    Excitation (nDof * N).  p(:,k) corresponds to time point t(k).
   u0   Displacements at time point t(1)-dt (nDof * 1).  Defaults to zero.
   u1   Displacements at time point t(1) (nDof * 1).  Defaults to zero.
   u    Displacements (nDof * N).  u(:,k) corresponds to time point t(k).
   t    Time axis (1 * N), defined as t = [0:N-1] * dt.
```

checkunique

```
CHECKUNIQUE   Check if vector contains unique elements.

   checkunique(p,name)
   find non-unique elements in a vector and print error

   p      Vector with elements to be checked
   name   Name of elements in p
```

cmat_isotropic

```
CMAT_ISOTROPIC  Constitutive matrix for isotropic materials.

   [C,C_lambda,C_mu]==cmat_isotropic(problem,Section,Material)
   computes the constitutive matrix for isotropic materials.

problem
Section Section definition
Material Material definition
C Constitutive matrix (nStress * nStrain)
   C_lambda    Contribution of lambda to C (nStress * nStrain)
   C_mu        Contribution of mu to C (nStress * nStrain)
```

cmat_shell8

```
CMAT_SHELL8   Constitutive matrix for shell8 element

   C = cmat_shell8(MatType,Material,k)
   C = cmat_shell8(MatType,Material)
   returns the constitutive matrix for shell8 elements.

   MatType    Material type: 'isotropic' or 'orthotropic'
   Material   Material definition
              isotropic:   [E nu rho]
              orthotropic: [Exx Eyy nuxy muxy muyz muzx theta rho]
   k         Geometric coefficient for non-uniform shear stress (default = 1.2)
   C         Constitutive matrix (5 * 5)

   See also KE_SHELL8
```

coord_beam

```
COORD_BEAM  Coordinates of the beam elements for plotting.

   [X,Y,Z]=coord_beam(Nodes,NodeNum)
   returns the coordinates of the beam elements for plotting.

   Nodes      Node definitions        [NodID x y z] (nNodes * 4)
   NodeNum Node numbers          [NodID1 NodID2 NodID3] (nElem * 3)
   X          X coordinates  (2 * nElem)
   Y          Y coordinates  (2 * nElem)
   Z          Z coordinates  (2 * nElem)

   See also COORD_TRUSS, PLOTELEM.
```

coord_kbeam

```
COORD_BEAM  Coordinates of the beam elements for plotting.

   coord_beam(Nodes,NodeNum)
   returns the coordinates of the beam elements for plotting.

   Nodes      Node definitions        [NodID x y z] (nNodes * 4)
   NodeNum Node numbers          [NodID1 NodID2 NodID3] (nElem * 3)
   X          X coordinates  (2 * nElem)
   Y          Y coordinates  (2 * nElem)
   Z          Z coordinates  (2 * nElem)

   See also COORD_TRUSS, PLOTELEM.
```

coord_mass

```
COORD_MASS   Coordinates of the mass element for plotting.

   [X,Y,Z]=coord_mass(Nodes,NodeNum)
   returns the coordinates of the mass element for plotting.

   Nodes      Node definitions        [NodID x y z] (nNode * 4)
   NodeNum Node numbers               [NodID1] (nElem * 1)
   X          X coordinates  (1 * nElem)
   Y          Y coordinates  (1 * nElem)
   Z          Z coordinates  (1 * nElem)

   See also COORD_BEAM, PLOTELEM.
```

coord_plane10

```
COORD_PLANE10  Coordinates of the plane elements for plotting.

   [X,Y,Z] = coord_plane10(Nodes,NodeNum)
   returns the coordinates of the plane10 elements for plotting.

   Nodes      Node definitions        [NodID x y z] (nNodes * 4)
   NodeNum    Node numbers            [NodID1 NodID2 ...] (nElem * 10)
   X          X coordinates  (15 * nElem)
   Y          Y coordinates  (15 * nElem)
   Z          Z coordinates  (15 * nElem)

   See also COORD_TRUSS, PLOTELEM.
```

coord_plane15

```
COORD_PLANE15  Coordinates of the plane elements for plotting.

   [X,Y,Z] = coord_plane15(Nodes,NodeNum)
   returns the coordinates of the plane15 elements for plotting.

   Nodes      Node definitions        [NodID x y z] (nNodes * 4)
   NodeNum    Node numbers            [NodID1 NodID2 ...] (nElem * 15)
   X          X coordinates  (15 * nElem)
   Y          Y coordinates  (15 * nElem)
   Z          Z coordinates  (15 * nElem)

   See also COORD_TRUSS, PLOTELEM.
```

coord_plane3

```
COORD_PLANE3  Coordinates of PLANE3 element sides for plotting.

   [X,Y,Z]=coord_plane3(Nodes,Nodenumbers)
   returns the coordinates of PLANE3 element sides for plotting.

   Nodes       Node definitions [NodID x y z] (nNodes * 3)
   Nodenumbers Node numbers [NodID1 NodID2 NodID3] (nElem * 3)
   X           X coordinates (3 * nElem)
   Y           Y coordinates (3 * nElem)
   Z           Z coordinates (3 * nElem)
```

coord_plane4

```
COORD_PLANE4  Coordinates of the plane elements for plotting.

   [X,Y,Z] = coord_plane4(Nodes,NodeNum)
   returns the coordinates of the plane4 elements for plotting.

   Nodes      Node definitions        [NodID x y z] (nNodes * 4)
   NodeNum    Node numbers            [NodID1 NodID2 NodID3 NodID4] (nElem * 4)
   X          X coordinates  (4 * nElem)
   Y          Y coordinates  (4 * nElem)
   Z          Z coordinates  (4 * nElem)

   See also COORD_TRUSS, PLOTELEM.
```

coord_plane6

```
COORD_PLANE6  Coordinates of the plane elements for plotting.

   [X,Y,Z] = coord_plane6(Nodes,NodeNum)
   returns the coordinates of the plane6 elements for plotting.

   Nodes      Node definitions        [NodID x y z] (nNodes * 4)
   NodeNum    Node numbers            [NodID1 NodID2 ...] (nElem * 6)
   X          X coordinates  (6 * nElem)
   Y          Y coordinates  (6 * nElem)
   Z          Z coordinates  (6 * nElem)

   See also COORD_TRUSS, PLOTELEM.
```

coord_plane8

```
COORD_PLANE8  Coordinates of the shell8 elements for plotting.

   coord_plane8(Nodes,NodeNum)
   returns the coordinates of the plane8 elements for plotting.

   Nodes      Node definitions        [NodID x y z] (nNodes * 4)
   NodeNum    Node numbers            [NodID1 NodID2 NodID3 NodID4] (nElem * 8)
   X          X coordinates  (20 * nElem)
   Y          Y coordinates  (20 * nElem)
   Z          Z coordinates  (20 * nElem)

   See also COORD_TRUSS, PLOTELEM.
```

coord_shell2

```
COORD_SHELL2  Coordinates of SHELL2 elements for plotting.

   [X,Y,Z]=coord_shell2(Nodes,NodeNum)
   returns the coordinates of the SHELL2 elements for plotting.

   Nodes      Node definitions        [NodID x y z] (nNodes * 4)
   NodeNum    Node numbers          [NodID1 NodID2] (nElem * 2)
   X          X coordinates  (2 * nElem)
   Y          Y coordinates  (2 * nElem)
   Z          Z coordinates  (2 * nElem)
```

coord_shell4

```
COORD_SHELL4  Coordinates of the shell elements for plotting.

   [X,Y,Z] = coord_shell4(Nodes,NodeNum)
   returns the coordinates of the shell4 elements for plotting.

   Nodes      Node definitions        [NodID x y z] (nNodes * 4)
   NodeNum    Node numbers            [NodID1 NodID2 NodID3 NodID4] (nElem * 4)
   X          X coordinates  (4 * nElem)
   Y          Y coordinates  (4 * nElem)
   Z          Z coordinates  (4 * nElem)

   See also COORD_TRUSS, PLOTELEM.
```

coord_shell6

```
COORD_SHELL6  Coordinates of the shell6 elements for plotting.

   [X,Y,Z] = coord_shell6(Nodes,NodeNum)
   returns the coordinates of the shell6 elements for plotting.

   Nodes      Node definitions        [NodID x y z] (nNodes * 4)
   NodeNum    Node numbers            [NodID1 NodID2 NodID3 NodID4] (nElem * 6)
   X          X coordinates  (15 * nElem)
   Y          Y coordinates  (15 * nElem)
   Z          Z coordinates  (15 * nElem)

   See also COORD_TRUSS, PLOTELEM.
```

coord_shell8

```
COORD_SHELL8  Coordinates of the shell8 elements for plotting.

   [X,Y,Z] = coord_shell8(Nodes,NodeNum)
   returns the coordinates of the shell8 elements for plotting.

   Nodes      Node definitions        [NodID x y z] (nNodes * 4)
   NodeNum    Node numbers            [NodID1 NodID2 NodID3 NodID4] (nElem * 8)
   X          X coordinates  (20 * nElem)
   Y          Y coordinates  (20 * nElem)
   Z          Z coordinates  (20 * nElem)

   See also COORD_TRUSS, PLOTELEM.
```

coord_solid10

```
COORD_SOLID10  Coordinates of SOLID10 element sides for plotting.

   [X,Y,Z]=coord_solid10(Nodes,Nodenumbers)
   returns the coordinates of SOLID10 element sides for plotting.

   Nodes       Node definitions [NodID x y z] (nNodes * 4)
   Nodenumbers Node numbers [NodID1 NodID2 NodID3 NodID4] (nElem * 4)
   X           X coordinates (4 * nElem)
   Y           Y coordinates (4 * nElem)
   Z           Z coordinates (4 * nElem)
```

coord_solid15

```
COORD_SOLID8  Coordinates of SOLID15 element sides for plotting.

   [X,Y,Z]=coord_solid15(Nodes,Nodenumbers)
   returns the coordinates of SOLID15 element sides for plotting.

   Nodes       Node definitions [NodID x y z] (nNodes * 4)
   Nodenumbers Node numbers [NodID1 NodID2 NodID3 NodID4] (nElem * 4)
   X           X coordinates (4 * nElem)
   Y           Y coordinates (4 * nElem)
   Z           Z coordinates (4 * nElem)
```

coord_solid20

```
COORD_SOLID20  Coordinates of SOLID20 element sides for plotting.

   [X,Y,Z]=coord_solid20(Nodes,Nodenumbers)
   returns the coordinates of SOLID20 element sides for plotting.

   Nodes       Node definitions [NodID x y z] (nNodes * 4)
   Nodenumbers Node numbers [NodID1 NodID2 NodID3 NodID4] (nElem * 4)
   X           X coordinates (4 * nElem)
   Y           Y coordinates (4 * nElem)
   Z           Z coordinates (4 * nElem)
```

coord_solid4

```
COORD_SOLID4  Coordinates of SOLID4 element sides for plotting.

   [X,Y,Z]=coord_rshell(Nodes,Nodenumbers)
   returns the coordinates of RSHELL element sides for plotting.

   Nodes       Node definitions [NodID x y z] (nNodes * 4)
   Nodenumbers Node numbers [NodID1 NodID2 NodID3 NodID4] (nElem * 4)
   X           X coordinates (4 * nElem)
   Y           Y coordinates (4 * nElem)
   Z           Z coordinates (4 * nElem)
```

coord_solid8

```
COORD_SOLID8  Coordinates of SOLID8 element sides for plotting.

   [X,Y,Z]=coord_solid8(Nodes,Nodenumbers)
   returns the coordinates of the SOLID8 element sides for plotting.

   Nodes       Node definitions [NodID x y z] (nNodes * 4)
   Nodenumbers Node numbers [NodID1 NodID2 NodID3 NodID4] (nElem * 4)
   X           X coordinates (4 * nElem)
   Y           Y coordinates (4 * nElem)
   Z           Z coordinates (4 * nElem)
```

coord_truss

```
COORD_TRUSS   Coordinates of the truss elements for plotting.

   [X,Y,Z]=coord_truss(Nodes,NodeNum)
   returns the coordinates of the truss elements for plotting.

   Nodes      Node definitions        [NodID x y z] (nNodes * 4)
   NodeNum Node numbers          [NodID1 NodID2] (nElem * 2)
   X          X coordinates  (2 * nElem)
   Y          Y coordinates  (2 * nElem)
   Z          Z coordinates  (2 * nElem)

   See also COORD_BEAM, PLOTELEM.
```

dispgcs2lcs_beam

```
DISPGCS2LCS_BEAM   Transform the element displacements to the LCS for a beam.

   UeLCS=dispgcs2lcs_beam(UeGCS,Node)
   transforms the element displacements from the GCS to the LCS for a beam
   element.

   Node       Node definitions           [x y z] (3 * 3)
   UeGCS      Displacements in the GCS (12 * 1)
   UeLCS      Displacements in the LCS (12 * 1)

   See also DISPGCS2LCS_TRUSS.
```

dispgcs2lcs_truss

```
DISPGCS2LCS_TRUSS  Transform the element displacements to the LCS for a truss.

   UeLCS=dispgcs2lcs_truss(UeGCS,Node)
   transforms the element displacements from the GCS to the LCS for a truss
   element.

   Node       Node definitions           [x y z] (2 * 3)
   UeGCS      Displacements in the GCS (6 * 1)
   UeLCS      Displacements in the LCS (6 * 1)

   See also DISPGCS2LCS_BEAM.
```

disp_beam

```
DISP_BEAM   Return matrices to compute the displacements of the deformed beams.

   [Ax,Ay,Az,B,Cx,Cy,Cz] = DISP_BEAM(Nodes,Elements,DOF,DLoads,Sections,Materials,Points)
   [Ax,Ay,Az,B,Cx,Cy,Cz] = DISP_BEAM(Nodes,Elements,DOF,DLoads,Sections,Materials)
   [Ax,Ay,Az,B] = DISP_BEAM(Nodes,Elements,DOF,[],Sections,Materials)
   [Ax,Ay,Az,B] = DISP_BEAM(Nodes,Elements,DOF)
       returns the matrices to compute the displacements of the deformed
       beams. The coordinates of the specified points along the deformed
       beam elements are computed using X=Ax*U+Cx*DLoad+B(:,1);
       Y=Ay*U+Cy*DLoad+B(:,2) and Z=Az*U+Cz*DLoad+B(:,3). The matrices
       Cx,Cy and Cz superimpose the displacements that occur due to the
       distributed loads if all nodes are fixed.

   [Ax,Ay,Az,B,Cx,Cy,Cz,dAxdx,dAydx,dAzdx,dCxdx,dCydx,dCzdx]
           = DISP_BEAM(Nodes,Elements,DOF,DLoads,Sections,Materials,Points,dNodesdx,
                                                           dDLoadsdx(,dSectionsdx))
       additionally computes the derivatives of the displacements with
       respect to the design variables x.

   Nodes      Node definitions         [NodID x y z]
   Elements   Element definitions      [EltID TypID SecID MatID n1 n2 ...]
   DOF        Degrees of freedom       (nDOF * 1)
   DLoads     Distributed loads        [EltID n1globalX n1globalY n1globalZ ...]
                                  (use an empty array [] when shear deformation is
                                                  considered but no DLoads are present)
   Sections   Section definitions    [SecID SecProp1 SecProp2 ...]
   Materials  Material definitions     [MatID MatProp1 MatProp2 ... ]
   Points     Points in the local coordinate system  (1 * nPoints)
   dNodesdx   Node definitions derivatives   (SIZE(Node) * nVar)
   dDLoadsdx  Distributed loads derivatives  (SIZE(DLoad) * nVar)
   Ax         Matrix to compute the x-coordinates of the deformations
   Ay         Matrix to compute the y-coordinates of the deformations
   Az         Matrix to compute the z-coordinates of the deformations
   B          Matrix which contains the x-, y- and z-coordinates of the
              undeformed structure
   Cx         Matrix to compute the x-coordinates of the deformations
   Cy         Matrix to compute the y-coordinates of the deformations
   Cz         Matrix to compute the z-coordinates of the deformations
   dAxdx, dAydx, dAzdx, dCxdx, dCydx, dCzdx
       Derivatives of the matrices to compute the coordinates of the
       interpolation points after deformation

   See also DISP_TRUSS, PLOTDISP, NELCS_BEAM, NEDLOADLCS_BEAM.
```

disp_mass

```
DISP_MASS    Matrices to compute the displacements of the deformed mass element

   [Ax,Ay,Az,B,Cx,Cy,Cz]
           =disp_beam(Nodes,Elements,DOF,EltIDDLoad,Sections,Materials,Points)
   [Ax,Ay,Az,B,Cx,Cy,Cz]
           =disp_beam(Nodes,Elements,DOF,EltIDDLoad,Sections,Materials)
   [Ax,Ay,Az,B]
           =disp_beam(Nodes,Elements,DOF,[],Sections,Materials)
   [Ax,Ay,Az,B]
           =disp_beam(Nodes,Elements,DOF)

   returns the matrices to compute the displacements of the deformed mass.
   The coordinates of the node of the mass element are
   computed using X=Ax*U+B(:,1); Y=Ay*U+B(:,2) and
   Z=Az*U+B(:,3). The matrices Cx,Cy and Cz superimpose the
   displacements that occur due to the distributed loads if all nodes are fixed.

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   DOF        Degrees of freedom  (nDOF * 1)
   Ax         Matrix to compute the x-coordinates of the deformations
   Ay         Matrix to compute the y-coordinates of the deformations
   Az         Matrix to compute the z-coordinates of the deformations
   B          Matrix which contains the x-, y- and z-coordinates of the
              undeformed structure
   Cx         Matrix to compute the x-coordinates of the deformations
   Cy         Matrix to compute the y-coordinates of the deformations
   Cz         Matrix to compute the z-coordinates of the deformations

   See also DISP_TRUSS, PLOTDISP, DISP_SHELL8.
```

disp_plane10

```
DISP_PLANE10   Matrices to compute the displacements of the deformed plane.

   [Ax,Ay,Az,B] = disp_plane10(Nodes,Elements,DOF,U)
   returns the matrices to compute the displacements of the deformed plane.
   The coordinates of the nodes of the plane10 element are
   computed using X=Ax*U+B(:,1); Y=Ay*U+B(:,2) and
   Z=Az*U+B(:,3).

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   DOF        Degrees of freedom  (nDOF * 1)
   Ax         Matrix to compute the x-coordinates of the deformations
   Ay         Matrix to compute the y-coordinates of the deformations
   Az         Matrix to compute the z-coordinates of the deformations
   B          Matrix which contains the x-, y- and z-coordinates of the
              undeformed structure

   See also DISP_TRUSS, PLOTDISP, DISP_SHELL4.
```

disp_plane15

```
DISP_PLANE15   Matrices to compute the displacements of the deformed plane.

   [Ax,Ay,Az,B] = disp_plane15(Nodes,Elements,DOF,U)
   returns the matrices to compute the displacements of the deformed plane.
   The coordinates of the nodes of the plane15 element are
   computed using X=Ax*U+B(:,1); Y=Ay*U+B(:,2) and
   Z=Az*U+B(:,3).

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   DOF        Degrees of freedom  (nDOF * 1)
   Ax         Matrix to compute the x-coordinates of the deformations
   Ay         Matrix to compute the y-coordinates of the deformations
   Az         Matrix to compute the z-coordinates of the deformations
   B          Matrix which contains the x-, y- and z-coordinates of the
              undeformed structure

   See also DISP_TRUSS, PLOTDISP, DISP_SHELL4.
```

disp_plane3

```
DISP_PLANE3    Matrices to compute the displacements of the deformed plane3.

   [Ax,Ay,Az,B] = disp_plane3(Nodes,Elements,DOF,U)
   returns the matrices to compute the displacements of the deformed plane3 element.
   The coordinates of the nodes of the plane3 element are
   computed using X=Ax*U+B(:,1); Y=Ay*U+B(:,2) and
   Z=Az*U+B(:,3).

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   DOF        Degrees of freedom  (nDOF * 1)
   Ax         Matrix to compute the x-coordinates of the deformations
   Ay         Matrix to compute the y-coordinates of the deformations
   Az         Matrix to compute the z-coordinates of the deformations
   B          Matrix which contains the x-, y- and z-coordinates of the
              undeformed structure

   See also DISP_TRUSS, PLOTDISP, DISP_SHELL8.
```

disp_plane4

```
DISP_PLANE4    Matrices to compute the displacements of the deformed plane4.

   [Ax,Ay,Az,B] = disp_plane4(Nodes,Elements,DOF,U)
   returns the matrices to compute the displacements of the deformed plane4.
   The coordinates of the nodes of the plane4 element are
   computed using X=Ax*U+B(:,1); Y=Ay*U+B(:,2) and
   Z=Az*U+B(:,3).

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   DOF        Degrees of freedom  (nDOF * 1)
   Ax         Matrix to compute the x-coordinates of the deformations
   Ay         Matrix to compute the y-coordinates of the deformations
   Az         Matrix to compute the z-coordinates of the deformations
   B          Matrix which contains the x-, y- and z-coordinates of the
              undeformed structure

   See also DISP_TRUSS, PLOTDISP, DISP_SHELL8.
```

disp_plane6

```
DISP_PLANE6   Matrices to compute the displacements of the deformed plane.

   [Ax,Ay,Az,B] = disp_plane6(Nodes,Elements,DOF,U)
   returns the matrices to compute the displacements of the deformed plane.
   The coordinates of the nodes of the plane6 element are
   computed using X=Ax*U+B(:,1); Y=Ay*U+B(:,2) and
   Z=Az*U+B(:,3).

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   DOF        Degrees of freedom  (nDOF * 1)
   Ax         Matrix to compute the x-coordinates of the deformations
   Ay         Matrix to compute the y-coordinates of the deformations
   Az         Matrix to compute the z-coordinates of the deformations
   B          Matrix which contains the x-, y- and z-coordinates of the
              undeformed structure

   See also DISP_TRUSS, PLOTDISP, DISP_SHELL4.
```

disp_plane8

```
DISP_PLANE8  Return matrices to compute the displacements of the deformed elements.

   [Ax,Ay,Az,B]=disp_plane8(Nodes,Elements,DOF)
   returns the matrices to compute the displacements of the deformed elements.
   The coordinates of the specified points along the deformed beams element are
   computed using
      X=Ax*U+B(:,1)
      Y=Ay*U+B(:,2)
      Z=Az*U+B(:,3)

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   DOF        Degrees of freedom  (nDOF * 1)
   Points     Points in local coordinate system (1 * nPoints)
   Ax         Matrix to compute the x-coordinates of the deformations
   Ay         Matrix to compute the y-coordinates of the deformations
   Az         Matrix to compute the z-coordinates of the deformations
   B          Matrix which contains the x-, y- and z-coordinates of the
              undeformed structure
```

disp_shell2

```
DISP_SHELL2   Return matrices to compute the displacements of deformed SHELL2 elements.

   [Ax,Ay,Az,B,Cx,Cy,Cz]
            =disp_shell2(Nodes,Elements,DOF,EltIDDLoad,Sections,Materials,Points)
   [Ax,Ay,Az,B,Cx,Cy,Cz]
            =disp_shell2(Nodes,Elements,DOF,EltIDDLoad,Sections,Materials)
   [Ax,Ay,Az,B]
            =disp_shell2(Nodes,Elements,DOF,[],Sections,Materials)
   [Ax,Ay,Az,B]
            =disp_shell2(Nodes,Elements,DOF)

   returns the matrices to compute the displacements of deformed SHELL2 elements.
   The coordinates of the specified points along the deformed SHELL2 element are
   computed using X=Ax*U+Cx*DLoad+B(:,1); Y=Ay*U+Cy*DLoad+B(:,2) and
   Z=Az*U+Cz*DLoad+B(:,3). The matrices Cx,Cy and Cz superimpose the
   displacements that occur due to the distributed loads if all nodes are fixed.
   In the current implementation, Cx=Cy=Cz=0, i.e. the local effect of the
   distributed loads is ignored.

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   DOF        Degrees of freedom  (nDOF * 1)
   EltIDDLoad Elements with distributed loads [EltID]
              (use empty array [] when shear deformation is considered
                                                  but no DLoads are present)
   Sections   Section definitions       [SecID SecProp1 SecProp2 ...]
   Materials  Material definitions      [MatID MatProp1 MatProp2 ... ]
   Points     Points in the local coordinate system (1 * nPoints)
   Ax         Matrix to compute the x-coordinates of the deformations
   Ay         Matrix to compute the y-coordinates of the deformations
   Az         Matrix to compute the z-coordinates of the deformations
   B          Matrix which contains the x-, y- and z-coordinates of the
              undeformed structure
   Cx         Matrix to compute the x-coordinates of the deformations
   Cy         Matrix to compute the y-coordinates of the deformations
   Cz         Matrix to compute the z-coordinates of the deformations
```

disp_shell4

```
DISP_SHELL4    Matrices to compute the displacements of the deformed shell4.

   [Ax,Ay,Az,B] = disp_shell4(Nodes,Elements,DOF,U)
   returns the matrices to compute the displacements of the deformed shell4.
   The coordinates of the nodes of the shell4 element are
   computed using X=Ax*U+B(:,1); Y=Ay*U+B(:,2) and
   Z=Az*U+B(:,3).

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   DOF        Degrees of freedom  (nDOF * 1)
   Ax         Matrix to compute the x-coordinates of the deformations
   Ay         Matrix to compute the y-coordinates of the deformations
   Az         Matrix to compute the z-coordinates of the deformations
   B          Matrix which contains the x-, y- and z-coordinates of the
              undeformed structure

   See also DISP_TRUSS, PLOTDISP, DISP_SHELL8.
```

disp_shell6

```
DISP_SHELL6   Matrices to compute the displacements of the deformed shell.

   [Ax,Ay,Az,B] = disp_shell6(Nodes,Elements,DOF,U)
   returns the matrices to compute the displacements of the deformed shell.
   The coordinates of the nodes of the shell6 element are
   computed using X=Ax*U+B(:,1); Y=Ay*U+B(:,2) and
   Z=Az*U+B(:,3).

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   DOF        Degrees of freedom  (nDOF * 1)
   Ax         Matrix to compute the x-coordinates of the deformations
   Ay         Matrix to compute the y-coordinates of the deformations
   Az         Matrix to compute the z-coordinates of the deformations
   B          Matrix which contains the x-, y- and z-coordinates of the
              undeformed structure

   See also DISP_TRUSS, PLOTDISP, DISP_SHELL4.
```

disp_shell8

```
DISP_SHELL8   Matrices to compute the displacements of the deformed shell.

   [Ax,Ay,Az,B] = disp_shell8(Nodes,Elements,DOF,U)
   returns the matrices to compute the displacements of the deformed shell.
   The coordinates of the nodes of the shell8 element are
   computed using X=Ax*U+B(:,1); Y=Ay*U+B(:,2) and
   Z=Az*U+B(:,3).

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   DOF        Degrees of freedom  (nDOF * 1)
   Ax         Matrix to compute the x-coordinates of the deformations
   Ay         Matrix to compute the y-coordinates of the deformations
   Az         Matrix to compute the z-coordinates of the deformations
   B          Matrix which contains the x-, y- and z-coordinates of the
              undeformed structure

   See also DISP_TRUSS, PLOTDISP, DISP_SHELL4.
```

disp_solid10

```
DISP_SOLID10  Return matrices to compute the displacements of the deformed elements.

   [Ax,Ay,Az,B]=disp_solid10(Nodes,Elements,DOF,Points)
   [Ax,Ay,Az,B]=disp_solid10(Nodes,Elements,DOF)
   returns the matrices to compute the displacements of the deformed elements.
   The coordinates of the specified points along the deformed beams element are
   computed using X=Ax*U+B(:,1); Y=Ay*U+B(:,2) and Z=Az*U+B(:,3).

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   DOF        Degrees of freedom  (nDOF * 1)
   Points     Points in local coordinate system (1 * nPoints)
   Ax         Matrix to compute the x-coordinates of the deformations
   Ay         Matrix to compute the y-coordinates of the deformations
   Az         Matrix to compute the z-coordinates of the deformations
   B          Matrix which contains the x-, y- and z-coordinates of the
              undeformed structure

   See also DISP_TRUSS, PLOTDISP.
```

disp_solid15

```
DISP_SOLID15  Return matrices to compute the displacements of the deformed elements.

   [Ax,Ay,Az,B]=disp_solid15(Nodes,Elements,DOF,Points)
   [Ax,Ay,Az,B]=disp_solid15(Nodes,Elements,DOF)
   returns the matrices to compute the displacements of the deformed elements.
   The coordinates of the specified points along the deformed beams element are
   computed using X=Ax*U+B(:,1); Y=Ay*U+B(:,2) and Z=Az*U+B(:,3).

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   DOF        Degrees of freedom  (nDOF * 1)
   Points     Points in local coordinate system (1 * nPoints)
   Ax         Matrix to compute the x-coordinates of the deformations
   Ay         Matrix to compute the y-coordinates of the deformations
   Az         Matrix to compute the z-coordinates of the deformations
   B          Matrix which contains the x-, y- and z-coordinates of the
              undeformed structure

   See also DISP_TRUSS, PLOTDISP.
```

disp_solid20

```
DISP_SOLID20  Return matrices to compute the displacements of the deformed elements.

   [Ax,Ay,Az,B]=disp_solid20(Nodes,Elements,DOF,Points)
   [Ax,Ay,Az,B]=disp_solid20(Nodes,Elements,DOF)
   returns the matrices to compute the displacements of the deformed elements.
   The coordinates of the specified points along the deformed beams element are
   computed using X=Ax*U+B(:,1); Y=Ay*U+B(:,2) and Z=Az*U+B(:,3).

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   DOF        Degrees of freedom  (nDOF * 1)
   Points     Points in local coordinate system (1 * nPoints)
   Ax         Matrix to compute the x-coordinates of the deformations
   Ay         Matrix to compute the y-coordinates of the deformations
   Az         Matrix to compute the z-coordinates of the deformations
   B          Matrix which contains the x-, y- and z-coordinates of the
              undeformed structure

   See also DISP_TRUSS, PLOTDISP.
```

disp_solid8

```
DISP_SOLID8  Return matrices to compute the displacements of the deformed elements.

   [Ax,Ay,Az,B]=disp_solid8(Nodes,Elements,DOF,Points)
   [Ax,Ay,Az,B]=disp_solid8(Nodes,Elements,DOF)
   returns the matrices to compute the displacements of the deformed elements.
   The coordinates of the specified points along the deformed beams element are
   computed using X=Ax*U+B(:,1); Y=Ay*U+B(:,2) and Z=Az*U+B(:,3).

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   DOF        Degrees of freedom  (nDOF * 1)
   Points     Points in local coordinate system (1 * nPoints)
   Ax         Matrix to compute the x-coordinates of the deformations
   Ay         Matrix to compute the y-coordinates of the deformations
   Az         Matrix to compute the z-coordinates of the deformations
   B          Matrix which contains the x-, y- and z-coordinates of the
              undeformed structure

   See also DISP_TRUSS, PLOTDISP.
```

disp_truss

```
DISP_TRUSS  Return matrices to compute the displacements of the deformed trusses.

   [Ax,Ay,Az,B]=disp_truss(Nodes,Elements,DOF,[],[],[],Points)
   [Ax,Ay,Az,B]=disp_truss(Nodes,Elements,DOF)
       returns the matrices to compute the displacements of the deformed
       trusses. The coordinates of the specified points along the deformed
       truss elements are computed using X=Ax*U+B(:,1); Y=Ay*U+B(:,2)
       and Z=Az*U+B(:,3).

   [Ax,Ay,Az,B,Cx,Cy,Cz,dAxdx,dAydx,dAzdx,dCxdx,dCydx,dCzdx]
           = DISP_TRUSS(Nodes,Elements,DOF,EltIDDLoad,Sections,Materials,Points,dNodesdx,
                                                                       dDLoadsdx(,dSectionsdx))
       additionally computes the derivatives of the displacements with
       respect to the design variables x.

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   DOF        Degrees of freedom  (nDOF * 1)
   Points     Points in the local coordinate system  (1 * nPoints)
   dNodesdx   Node definitions derivatives   (SIZE(Node) * nVar)
   dDLoads    Distributed loads derivatives  (SIZE(DLoad) * nVar)
   Ax         Matrix to compute the x-coordinates of the deformations
   Ay         Matrix to compute the y-coordinates of the deformations
   Az         Matrix to compute the z-coordinates of the deformations
   B          Matrix which contains the x-, y- and z-coordinates of the
              undeformed structure
   dAxdx, dAydx, dAzdx, dCxdx, dCydx, dCzdx
       Derivatives of the matrices to compute the coordinates of the
       interpolation points after deformation

   See also DISP_BEAM, PLOTDISP.
```

dloadgcs2lcs

```
DLOADGCS2LCS   Distributed load transformation from GCS to LCS.

   DLoadLCS = DLOADGCS2LCS(T,DLoad)
       transforms the distributed load definitions in the global
       coordinate system (algebraic convention) to the local coordinate
       system (beam convention).

   [DLoadLCS,dDLoadLCSdx] = DLOADGCS2LCS(T,DLoad,dTdx,dDLoaddx)
       transforms the distributed load definitions in the global
       coordinate system (algebraic convention) to the local coordinate
       system (beam convention), and additionally computes the derivatives
       of the distributed load information with respect to the design
       variables x.

   T           Element transformation matrix        (6 * 6)
   DLoad       Distributed loads in GCS             [n1globalX; n1globalY; n1globalZ; ...]
                (6/8 * nLC * nDLoads)
   dTdx        Transformation matrix derivatives    (6 * 6 * nVar)
   dDLoaddx    Distributed loads derivatives (GCS)  (SIZE(DLoad) * nVar)
   DLoadLCS    Distributed loads in LCS             [n1localX; n1localY; n1localZ; ...]
                (6/8 * nLC * nDLoads)
   dDLoadLCSdx Distributed loads derivatives (LCS)  (SIZE(DLoadLCS) * nVar)
```

dof_beam

```
DOF_BEAM   Element degrees of freedom for a beam element.

   DOF = dof_beam(NodeNum) builds the vector with the
   builds the vector with the degrees of freedom for the beam element.

   NodeNum Node definitions           [NodID1 NodID2]  (1 * 2)
   DOF     Degrees of freedom                          (12 * 1)

   See also GETDOF.
```

dof_mass

```
DOF_MASS   Element degrees of freedom for a mass element.

   dof = dof_truss(NodeNum) builds the vector with the
   labels of the degrees of freedom for which stiffness is present in the
   mass element.

   NodeNum Node numbers           [NodID1] (1)
   dof         Degrees of freedom  (6 * 1)

   See also GETDOF.
```

dof_plane10

```
DOF_PLANE10   Element degrees of freedom for a plane10 element.

   dof = dof_plane10(NodeNum) builds the vector with the
   labels of the degrees of freedom for which stiffness is present in the
   plane10 element.

   NodeNum Node definitions           [NodID1 NodID2 ... NodIDn]   (1 * 10)
   dof     Degrees of freedom                                      (20 * 1)

   See also GETDOF.
```

dof_plane15

```
DOF_PLANE15   Element degrees of freedom for a plane15 element.

   dof = dof_plane15(NodeNum) builds the vector with the
   labels of the degrees of freedom for which stiffness is present in the
   plane15 element.

   NodeNum Node definitions           [NodID1 NodID2 ... NodIDn]   (1 * 15)
   dof     Degrees of freedom                                      (30 * 1)

   See also GETDOF.
```

dof_plane3

```
DOF_PLANE3   Element degrees of freedom for a plane3 element.

   dof = dof_plane3(NodeNum) builds the vector with the
   labels of the degrees of freedom for which stiffness is present in the
   plane3 element.

   NodeNum Node definitions           [NodID1 NodID2 NodID3]   (1 * 3)
   dof     Degrees of freedom                                  (6 * 1)

   See also GETDOF.
```

dof_plane4

```
DOF_PLANE4   Element degrees of freedom for a plane4 element.

   dof = dof_plane4(NodeNum) builds the vector with the
   labels of the degrees of freedom for which stiffness is present in the
   plane4 element.

   NodeNum Node definitions           [NodID1 NodID2 ... NodIDn]   (1 * 4)
   dof     Degrees of freedom                                      (8 * 1)

   See also GETDOF.
```

dof_plane6

```
DOF_PLANE6   Element degrees of freedom for a plane6 element.

   dof = dof_plane6(NodeNum) builds the vector with the
   labels of the degrees of freedom for which stiffness is present in the
   plane6 element.

   NodeNum Node definitions           [NodID1 NodID2 ... NodIDn]   (1 * 6)
   dof     Degrees of freedom                                      (12 * 1)

   See also GETDOF.
```

dof_plane8

```
of_plane8 is a function.
   dof = dof_plane8(Nodenumbers)
```

dof_shell2

```
DOF_SHELL2   Element degrees of freedom for a SHELL2 element.

   DOF = dof_shell2(NodeNum) returns a vector with the degrees of freedom for
   a SHELL2 element.

   NodeNum Node definitions           [NodID1 NodID2]  (1 * 2)
   DOF     Degrees of freedom                          (12 * 1)
```

dof_shell4

```
DOF_SHELL4   Element degrees of freedom for a shell4 element.

   dof = dof_shell4(NodeNum) builds the vector with the
   labels of the degrees of freedom for which stiffness is present in the
   shell4 element.

   NodeNum Node definitions           [NodID1 NodID2 ... NodIDn]   (1 * 4)
   dof         Degrees of freedom                                 (24 * 1)

   See also GETDOF.
```

dof_shell6

```
DOF_SHELL6   Element degrees of freedom for a shell6 element.

   dof = dof_shell6(NodeNum) builds the vector with the
   labels of the degrees of freedom for which stiffness is present in the
   shell6 element.

   NodeNum Node definitions           [NodID1 NodID2 ... NodIDn]   (1 * 6)
   dof     Degrees of freedom                                     (36 * 1)

   See also GETDOF.
```

dof_shell8

```
DOF_SHELL8   Element degrees of freedom for a shell8 element.

   dof = dof_shell8(NodeNum) builds the vector with the
   labels of the degrees of freedom for which stiffness is present in the
   shell8 element.

   NodeNum Node definitions           [NodID1 NodID2 ... NodIDn]   (1 * 8)
   dof     Degrees of freedom                                     (48 * 1)

   See also GETDOF.
```

dof_solid10

```
DOF_SOLID10   Element degrees of freedom for a solid10 element.

   dof = dof_solid10(NodeNum) builds the vector with the
   labels of the degrees of freedom for which stiffness is present in the
   solid10 element.

   NodeNum Node definitions           [NodID1 NodID2 ... NodIDn]   (1 * 4)
   dof         Degrees of freedom                                 (30 * 1)

   See also GETDOF.
```

dof_solid15

```
DOF_SOLID15   Element degrees of freedom for a solid10 element.

   dof = dof_solid15(NodeNum) builds the vector with the
   labels of the degrees of freedom for which stiffness is present in the
   solid15 element.

   NodeNum Node definitions           [NodID1 NodID2 ... NodIDn]   (1 * 4)
   dof         Degrees of freedom                                 (45 * 1)

   See also GETDOF.
```

dof_solid20

```
DOF_SOLID20   Element degrees of freedom for a solid20 element.

   dof = dof_solid20(NodeNum) builds the vector with the
   labels of the degrees of freedom for which stiffness is present in the
   solid20 element.

   NodeNum Node definitions           [NodID1 NodID2 ... NodIDn]   (1 * 4)
   dof         Degrees of freedom                                 (60 * 1)

   See also GETDOF.
```

dof_solid4

```
DOF_SOLID4   Element degrees of freedom for a solid4 element.

   dof = dof_solid4(NodeNum) builds the vector with the
   labels of the degrees of freedom for which stiffness is present in the
   solid4 element.

   NodeNum   Node definitions    [NodID1 NodID2 NodID3 NodID4]   (1 * 4).
   dof       Degrees of freedom                                 (12 * 1).

   See also GETDOF.
```

dof_solid8

```
DOF_SOLID8   Element degrees of freedom for a solid8 element.

   dof = dof_solid8(NodeNum) builds the vector with the
   labels of the degrees of freedom for which stiffness is present in the
   solid8 element.

   NodeNum Node definitions           [NodID1 NodID2 ... NodIDn]   (1 * 4)
   dof         Degrees of freedom                                 (24 * 1)

   See also GETDOF.
```

dof_truss

```
DOF_TRUSS   Element degrees of freedom for a truss element.

   dof = dof_truss(NodeNum) builds the vector with the degrees of freedom
   for the truss element.

   NodeNum Node numbers           [NodID1 NodID2] (1 * 2)
   dof         Degrees of freedom  (6 * 1)

   See also GETDOF.
```

eigfem

```
EIGFEM  Compute the eigenmodes and eigenfrequencies of the finite element model.

   [phi,omega]=eigfem(K,M,nMode)
   [phi,omega]=eigfem(K,M)
   computes the eigenmodes and eigenfrequencies of the finite element model.

   K           Stiffness matrix (nDOF * nDOF)
   M           Mass matrix (nDOF * nDOF)
   nMode       Number of eigenmodes and eigenfrequencies (default: all)
   phi         Eigenmodes (in columns) (nDOF * nMode)
   omega       Eigenfrequencies [rad/s] (nMode * 1)
```

elemdisp

```
ELEMDISP  Select the element displacements from the global displacement vector.

   UeGCS = elemdisp(Type,NodeNum,DOF,U) selects the element
   displacements from the global displacement vector.

   Type       Element type e.g. 'beam','truss', ...
   NodeNum    Node numbers (1 * nNodes)
   DOF        Degrees of freedom  (nDOF * 1)
   U          Displacements (nDOF * nLC)
   UeGCS      Element displacements

   See also ELEMFORCES, DOF_BEAM, DOF_TRUSS.
```

elemforces

```
ELEMFORCES   Compute the element forces.

   [ForcesLCS,ForcesGCS] = ELEMFORCES(Nodes,Elements,Types,Sections,Materials,DOF,U,DLoads,TLoads)
   [ForcesLCS,ForcesGCS] = ELEMFORCES(Nodes,Elements,Types,Sections,Materials,DOF,U)
       computes the element forces in the local (beam convention) and the
       global (algebraic convention) coordinate system.

   [ForcesLCS,ForcesGCS,dForcesLCSdx,dForcesGCSdx]
           = ELEMFORCES(Nodes,Elements,Types,Sections,Materials,DOF,U,DLoads,TLoads,
                                                  dNodesdx,dSectionsdx,dUdx,dDLoadsdx)
       additionally computes the derivatives of the element forces with
       respect to the design variables x.

   Nodes          Node definitions          [NodID x y z]
   Elements       Element definitions       [EltID TypID SecID MatID n1 n2 ...]
   Types          Element type definitions  {TypID EltName Option1 ...}
   Sections       Section definitions       [SecID SecProp1 SecProp2 ...]
   Materials      Material definitions      [MatID MatProp1 MatProp2 ...]
   DOF            Degrees of freedom        (nDOF * 1)
   U              Displacements             (nDOF * nLC)
   DLoads         Distributed loads         [EltID n1globalX n1globalY n1globalZ ...]
   TLoads         Temperature loads         [EltID Tyt Tyb Tzt Tzb]
   dNodesdx       Node definitions derivatives    (SIZE(Nodes) * nVar)
   dSectionsdx    Section definitions derivatives (SIZE(Sections) * nVar)
   dUdx           Displacements derivatives       (nDOF * nLC * nVar)
   dDLoadsdx      Distributed loads derivative (SIZE(DLoads) * nVar)
   ForcesLCS      Element forces in LCS (beam convention)  [N Vy Vz T My Mz] (nElem * 12 * nLC)
   ForcesGCS      Element forces in GCS (algebraic convention)             (nElem * 12 * nLC)
   dForcesLCSdx   Element forces derivatives in LCS  (nElem * 12 * nLC * nVar)
   dForcesGCSdx   Element forces derivatives in GCS  (nElem * 12 * nLC * nVar)

   See also FORCES_TRUSS, FORCES_BEAM.
```

elemloads

```
ELEMLOADS   Equivalent nodal forces.

   F = ELEMLOADS(DLoads,Nodes,Elements,Types,DOF)
   computes the equivalent nodal forces of a distributed load
   (in the global coordinate system).

   [F,dFdx] = ELEMLOADS(DLoads,Nodes,Elements,Types,DOF,dDLoadsdx,dNodesdx)
   additionally computes the derivatives of the equivalent nodal forces
   with respect to the design variables x.

   DLoads     Distributed loads               [EltID n1globalX n1globalY n1globalZ ...]
   Nodes      Node definitions                [NodID x y z]
   Elements   Element definitions             [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions        {TypID EltName Option1 ... }
   DOF        Degrees of freedom              (nDOF * 1)
   dDLoadsdx  Distributed loads derivatives   (SIZE(DLoads) * nVar)
   dNodesdx   Node definitions derivatives    (SIZE(Nodes) * nVar)
   F          Load vector             (nDOF * nLC)
   dFdx       Load vector derivatives  (nDOF * nLC * nVar)

   See also LOADS_TRUSS, LOADS_BEAM, NODALVALUES.
```

elempressure

```
ELEMPRESSURE   Equivalent nodal forces for a pressure load on a shell element.

   F = elempressure(Pressures,Nodes,Elements,Types,DOF)
   computes the equivalent nodal forces of a distributed pressure
   (in the local coordinate system xyz, with z perpendicular to the surface).

   Pressures  Pressure on surface or edge   [EltID n1localZ n2localZ n3localZ ...]
   Nodes      Node definitions              [NodID x y z]
   Elements   Element definitions           [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions      {TypID EltName Option1 ... }
   DOF        Degrees of freedom            (nDOF * 1)
   F          Load vector                   (nDOF * 1)

   See also PRESSURE_SHELL8, PRESSURE_SHELL4, NODALVALUES.
```

elemsize

```
ELEMSIZE   Compute element length/area/volume.

   S = ELEMSIZE(Nodes,Elements,Types)
   computes the size of all elements, depending on element type. For a 1D
   line element it computes length, for a 2D plate element it computes
   area, and for a 3D solid element it computes volume.

   [S,dSdx] = ELEMSIZE(Nodes,Elements,Types,dNodesdx)
   additionaly computes the derivatives of the size with respect to the
   design variables x.

   Nodes       Node definitions               [NodID x y z]
   Elements    Element definitions            [EltID TypID SecID MatID n1 n2 ...]
   Types       Element type definitions       {TypID EltName Option1 ... }
   dNodesdx    Node definitions derivatives  (SIZE(Nodes) * nVar)
   S           Element sizes
   dSdx        Element sizes derivatives

   See also SIZE_BEAM, SIZE_TRUSS, ELEMVOLUMES.
```

elemstress

```
ELEMSTRESS   Compute the element stresses.

   [SeGCS,SeLCS,vLCS] = elemstress(Nodes,Elements,Types,Sections,Materials,DOF,U)
   [SeGCS,SeLCS]      = elemstress(Nodes,Elements,Types,Sections,Materials,DOF,U)
    SeGCS             = elemstress(Nodes,Elements,Types,Sections,Materials,DOF,U)
   computes the element stresses in the global and the local coordinate system.

   Nodes      Node definitions          [NodID x y z]
   Elements   Element definitions       [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }
   Sections   Section definitions       [SecID SecProp1 SecProp2 ...]
   Materials  Material definitions      [MatID MatProp1 MatProp2 ... ]
   DOF        Degrees of freedom  (nDOF * 1)
   U          Displacements (nDOF * 1)
   elemstress(...,ParamName,ParamValue) sets the value of the specified
   parameters.  The following parameters can be specified:
   'GCS'          Sets the gcs in which SeGCS is computed.
                  Default: 'cart'. Type of values:
                  'cart': cartesian coordinate system
                  'cyl' : cylindrical coordinate system
                  'sph' : spherical coordinate system

   SeGCS      Element stresses in GCS in corner nodes IJKL and
              at top/mid/bot of shell (nElem * 72)
              72 = 6 stresscomp. * 4 nodes * 3 locations
                                        [sxx syy szz sxy syz sxz]
   SeLCS      Element stresses in LCS in corner nodes IJKL and
              at top/mid/bot of shell (nElem * 72)
                                        [sxx syy szz sxy syz sxz]
   vLCS       Unit vectors of LCS       (nElem * 9)

   See also SE_SHELL8, SE_SHELL4.
```

elemtloads

```
ELEMTLOADS   Equivalent nodal forces for temperature loading.

   F=elemtloads(TLoads,Nodes,Elements,Types,Sections,Materials,DOF)
   computes the equivalent nodal forces of a temperature gradient
   (in the global coordinate system).

   TLoads     Temperature gradient         [EltID Tyt Tyb Tzt Tzb]
              Tkt and Tkb correspond to the temperatures at the top and
              the bottom of the profile when the k-axis (LCS) points up (k=y,z)
   Nodes      Node definitions             [NodID x y z]
   Elements   Element definitions          [EltID TypID SecID MatID n1 n2 ...]
   Types      Type definitions             {TypID EltName Option1 ... }
   Sections   Section definitions          [SecID A ky kz Ixx Iyy Izz yt yb zt zb]
   Materials  Material definitions         [MatID E nu rho alpha]
   DOF        Degrees of freedom           (nDOF * 1)
   F          Load vector                  (nDOF * 1)

   See also TLOADS_TRUSS, TLOADS_BEAM.
```

elemvolumes

```
ELEMVOLUMES   Compute element volumes.

   V = ELEMVOLUMES(Nodes,Elements,Types,Sections)
   computes the volume for all elements.

   [V,dVdx] = ELEMVOLUMES(Nodes,Elements,Types,Sections,dNodesdx,dSectionsdx)
   computes the volume for all elements, as well as the derivatives of the
   volume with respect to the design variables x.

   Nodes       Node definitions          [NodID x y z]
   Elements    Element definitions       [EltID TypID SecID MatID n1 n2 ...]
   Types       Element type definitions  {TypID EltName Option1 ... }
   Sections    Section definitions       [SecID SecProp1 SecProp2 ...]
   dNodesdx    Node definitions derivatives     (SIZE(Nodes) * nVar)
   dSectionsdx Section dervinitions derivatives (SIZE(Sections) * nVar)
   V           Element volumes              (nElem * 1)
   dVdx        Element volumes derivatives  (nElem * nVar)

   See also VOLUME_BEAM, VOLUME_TRUSS, ELEMSIZES.
```

fdiagrgcs_beam

```
FDIAGRGCS_BEAM   Return matrices to plot the forces in a beam element.

   [ElemGCS,FdiagrGCS,ElemExtGCS,ExtremaGCS,Extrema]
                              = fdiagrgcs_beam(ftype,Forces,Node,[],[],DLoad,Points)
   [ElemGCS,FdiagrGCS,ElemExtGCS,ExtremaGCS,Extrema]
                              = fdiagrgcs_beam(ftype,Forces,Node,[],[],DLoad)
   returns the coordinates of the points along the beam in the global
   coordinate system and the coordinates of the forces with respect to the beam
   in the global coordinate system. These can be added in order to plot the
   forces: ElemGCS+FdiagrGCS. The coordinates of the points with extreme values
   and the coordinates of the extreme values with respect to the beam are given
   as well and can be similarly added:  ElemExtGCS+ExtremaGCS. Extrema is the
   list with the correspondig extreme values.

   ftype      'norm'       Normal force (in the local x-direction)
              'sheary'     Shear force in the local y-direction
              'shearz'     Shear force in the local z-direction
              'momx'       Torsional moment (around the local x-direction)
              'momy'       Bending moment around the local y-direction
              'momz'       Bending moment around the local z-direction
   Forces     Element forces in LCS (beam convention) [N; Vy; Vz; T; My; Mz]
                                                                       (12 * 1)
   Node       Node definitions        [x y z] (3 * 3)
   DLoad      Distributed loads       [n1globalX; n1globalY; n1globalZ; ...]
                                                                        (6 * 1)
   Points     Points in the local coordinate system (1 * nPoints)
   ElemGCS    Coordinates of the points along the beam in GCS (nPoints * 3)
   FdiagrGCS  Coordinates of the force with respect to the beam in GCS
                                                                  (nValues * 3)
   ElemExtGCS Coordinates of the points with extreme values in GCS (nValues * 3)
   ExtremaGCS Coordinates of the extreme values with respect to the beam in GCS
                                                                  (nValues * 3)
   Extrema    Extreme values (nValues * 1)

   See also PLOTFORC, FDIAGRLCS_BEAM, FDIAGRGCS_TRUSS.
```

fdiagrgcs_shell2

```
FDIAGRGCS_SHELL2   Return matrices to plot the forces in a SHELL2 element.

   [ElemGCS,FdiagrGCS,ElemExtGCS,ExtremaGCS,Extrema]
                  = fdiagrgcs_shell2(ftype,Forces,Node,Section,Material,DLoad,Points)
   [ElemGCS,FdiagrGCS,ElemExtGCS,ExtremaGCS,Extrema]
                  = fdiagrgcs_shell2(ftype,Forces,Node,Section,Material,DLoad)
   returns the coordinates of the points along the SHELL2 in the global
   coordinate system and the coordinates of the forces with respect to the element
   in the global coordinate system. These can be added in order to plot the
   forces: ElemGCS+FdiagrGCS. The coordinates of the points with extreme values
   and the coordinates of the extreme values with respect to the element are given
   as well and can be similarly added:  ElemExtGCS+ExtremaGCS. Extrema is the
   list with the correspondig extreme values.

   ftype      'Nphi'       Normal force (per unit length) in meridional direction
              'Qphi'       Transverse force (per unit length) in meridional direction
              'Mphi'       Bending moment (per unit length) in meridional direction
              'Ntheta'     Normal force (per unit length) in circumferential direction
              'Mtheta'     Bending moment (per unit length) in circumferential direction
   Forces     Element forces in LCS (beam convention) [N; Vy; 0; 0; 0; Mz] (12 * 1)
   Node       Node definitions        [x y z] (3 * 3)
   DLoad      Distributed loads       [n1globalX; n1globalY; n1globalZ; ...] (6 * 1)
   Points     Points in the local coordinate system (1 * nPoints)
   ElemGCS    Coordinates of the points along the element in GCS (nPoints * 3)
   FdiagrGCS  Coordinates of the force with respect to the element in GCS (nValues * 3)
   ElemExtGCS Coordinates of the points with extreme values in GCS (nValues * 3)
   ExtremaGCS Coordinates of the extreme values with respect to the element in GCS
                                                                     (nValues * 3)
   Extrema    Extreme values (nValues * 1)
```

fdiagrgcs_truss

```
FDIAGRGCS_TRUSS   Return matrices to plot the forces in a truss element.

   [ElemGCS,FdiagrGCS,ElemExtGCS,ExtremaGCS,Extrema]
                           = fdiagrgcs_truss(ftype,Forces,Node,[],[],[],Points)
   [ElemGCS,FdiagrGCS,ElemExtGCS,ExtremaGCS,Extrema]
                           = fdiagrgcs_truss(ftype,Forces,Node)
   returns the coordinates of the points along the truss in the global
   coordinate system and the coordinates of the forces with respect to the truss
   in the global coordinate system. These can be added in order to plot the
   forces: ElemGCS+FdiagrGCS. The coordinates of the points with extreme values
   and the coordinates of the extreme values with respect to the truss are given
   as well and can be similarly added:  ElemExtGCS+ExtremaGCS. Extrema is the
   list with the correspondig extreme values.

   ftype      'norm'       Normal force (in the local x-direction)
   Forces     Element forces in LCS [N; 0; 0; 0; 0; 0](12 * 1)
   Node       Node definitions        [x y z] (3 * 3)
   Points     Points in the local coordinate system (1 * nPoints)
   ElemGCS    Coordinates of the points along the truss in GCS (nPoints * 3)
   FdiagrGCS  Coordinates of the force with respect to the truss in GCS
                                                                 (nValues * 3)
   ElemExtGCS Coordinates of the points with extreme values in GCS (nValues * 3)
   ExtremaGCS Coordinates of the extreme values with respect to the truss in GCS
                                                                 (nValues * 3)
   Extrema    Extreme values (nValues * 1)

   See also PLOTFORC, FDIAGRGCS_BEAM.
```

fdiagrlcs

```
FDIAGRLCS   Return force diagrams in LCS.

   FdiagrLCS = FDIAGRLCS(ftype,Nodes,Elements,Types,Forces,DLoads,Points)
   FdiagrLCS = FDIAGRLCS(ftype,Nodes,Elements,Types,Forces,DLoads)
   FdiagrLCS = FDIAGRLCS(ftype,Nodes,Elements,Types,Forces)
       computes element force values in all interpolation points (in beam
       convention) for beam and truss elements.

   [FdiagrLCS,dFdiagrLCSdx]
           = FDIAGRLCS(ftype,Nodes,Elements,Types,Forces,DLoads,...
                                         Points,dNodesdx,dForcesdx,dDLoadsdx)
       additionally computes the derivatives of the force values with
       respect to the design variables x.

   ftype           'norm'       Normal force (in the local x-direction)
                   'sheary'     Shear force in the local y-direction
                   'shearz'     Shear force in the local z-direction
                   'momx'       Torsional moment (around the local x-direction)
                   'momy'       Bending moment around the local y-direction
                   'momz'       Bending moment around the local z-direction
   Nodes           Node definitions          [NodID x y z]
   Elements        Element definitions       [EltID TypID SecID MatID n1 n2 ...]
   Types           Element type definitions  {TypID EltName Option1 ... }
   Forces          Element forces in LCS  (beam convention) [N Vy Vz T My Mz]
                                                            (nElem * 12)
   DLoads          Distributed loads         [EltID n1globalX n1globalY n1globalZ ...]
   Points          Points in the local coordinate system  (1 * nPoints)
   dNodesdx        Node definitions derivatives        (SIZE(Nodes) * nVar)
   dForcesdx       Element forces in LCS derivatives   (SIZE(Forces) * nVar)
   dDLoadsdx       Distributed loads derivatives       (SIZE(DLoads) * nVar)
   FdiagrLCS       Element force values at the points  (nElem * nPoints * nLC)
   dFdiagrLCSdx    Element force values derivatives    (nElem * nPoints * nLC * nVar)

   See also PLOTFORC, FDIAGRGCS_BEAM, FDIAGRGCS_TRUSS.
```

fdiagrlcs_beam

```
FDIAGRLCS_BEAM   Force diagram for a beam element in LCS.

   [FdiagrLCS,loc,Extrema] = FDIAGRLCS_BEAM(ftype,Forces,DLoadLCS,L,Points)
   computes the elements forces at the specified points. The extreme
   values for an element with a single DLoad are analytically determined.
   The extreme values for an element with multiple DLoads are calculated
   in the interpolation points only.

   [FdiagrLCS,loc,Extrema,dFdiagrLCSdx]
       = FDIAGRLCS_BEAM(ftype,Forces,DLoadLCS,L,Points,dForcesdx,dDLoadLCSdx,dLdx)
   additionally computes the derivatives of the element force values with
   respect to the design variables x.

   ftype           'norm'       Normal force (in the local x-direction)
                   'sheary'     Shear force in the local y-direction
                   'shearz'     Shear force in the local z-direction
                   'momx'       Torsional moment (around the local x-direction)
                   'momy'       Bending moment around the local y-direction
                   'momz'       Bending moment around the local z-direction
   Forces          Element forces in LCS (beam convention) [N; Vy; Vz; T; My; Mz](12 * nLC)
   DLoadLCS        Distributed loads in LCS [n1localX; n1localY; n1localZ; ...] (6 * nLC)
   L               Beam length
   Points          Points in the local coordinate system (1 * nPoints)
   dForcesdx       Element forces in LCS derivatives      (SIZE(Forces) * nVar)
   dDLoadLCSdx     Distributed loads derivatives          (SIZE(DLoadLCS) * nVar)
   dLdx            Beam length derivatives                (1 * nVar)
   FdiagrLCS       Element forces at the points           (1 * nPoints * nLC)
   loc             Locations of the extreme values        (nValues * nLC)
   Extrema         Extreme values                         (nValues * nLC)
                       loc and Extrema are only calculated when nLC = 1 (for plotting).
                       If this is not the case their calculation is omitted for effiency.
   dFdiagrLCSdx    Element force value derivatives        (1 * nPoints * nLC * nVar)

   See also FDIAGRGCS_BEAM.
```

fdiagrlcs_shell2

```
FDIAGRLCS_SHELL2   Force diagram for a SHELL2 element in LCS.

   [FdiagrLCS,loc,Extrema] = fdiagrlcs_shell2(ftype,Forces,DLoadLCS,L,Points)
   computes the elements forces at the specified points. The extreme values are
   obtained by enumeration.

   ftype      'Nphi'       Normal force (per unit length) in meridional direction
              'Qphi'       Transverse force (per unit length) in meridional direction
              'Mphi'       Bending moment (per unit length) in meridional direction
              'Ntheta'     Normal force (per unit length) in circumferential direction
              'Mtheta'     Bending moment (per unit length) in circumferential direction
   Forces     Element forces in LCS (beam convention) [N; Vy; 0; 0; 0; Mz] (12 * 1)
   DLoadLCS   Distributed loads in LCS [n1localX; n1localY; n1localZ; ...]
                                                              (6 * 1) or (12 * 1)
   Points     Points in the local coordinate system (1 * nPoints)
   FdiagrLCS  Element forces at the points (1 * nPoints)
   loc        Locations of the extreme values (nValues * 1)
   Extrema    Extreme values (nValues * 1)
```

fdiagrlcs_truss

```
FDIAGRLCS_TRUSS   Force diagram for a truss element in LCS.

   [FdiagrLCS,loc,Extrema] = FDIAGRLCS_TRUSS(ftype,Forces,DLoadLCS,L,Points)
   computes the element forces at the specified points.

   [FdiagrLCS,loc,Extrema,dFdiagrLCSdx]
       = FDIAGRLCS_TRUSS(ftype,Forces,DLoadLCS,L,Points,dForcesdx,dDLoadLCSdx,dLdx)
   additionally computes the derivatives of the element force values with
   respect to the design variables x.

   ftype           'norm'       Normal force (in the local x-direction)
   Forces          Element forces in LCS   [N; 0; 0; 0; 0; 0] (12 * 1)
   Node            Node definitions        [x y z] (3 * 3)
   Points          Points in the local coordinate system  (1 * nPoints)
   dForcesdx       Element forces in LCS derivatives      (SIZE(Forces) * nVar)
   FdiagrLCS       Element forces at the points           (1 * nPoints * nLC)
   loc             Locations of the extreme values        (nValues * nLC)
   Extrema         Extreme values                         (nValues * nLC)
   dFdiagrLCSdx    Element force value derivatives        (1 * nPoints * nLC * nVar)

   See also FDIAGRGCS_TRUSS.
```

forceslcs_beam

```
FORCESLCS_BEAM   Compute the element forces for a beam element in the LCS.

   Forces = FORCESLCS_BEAM(KeLCS,UeLCS,DLoadLCS,L,TLoadLCS,A,E,alpha,Iyy,Izz,hy,hz)
   Forces = FORCESLCS_BEAM(KeLCS,UeLCS,DLoadLCS,L)
   Forces = FORCESLCS_BEAM(KeLCS,UeLCS)
       computes the element forces for the beam element in the local
       coordinate system (algebraic convention).

   [Forces,dForcesdx] = FORCESLCS_BEAM(KeLCS,UeLCS,DLoadLCS,L,[],[],[],...
                                 [],[],[],[],[],dKeLCSdx,dUeLCSdx,dDLoadLCSdx,dLdx)
   [Forces,dForcesdx] = FORCESLCS_BEAM(KeLCS,UeLCS,[],[],[],[],[],[],[],...
                                       [],[],[],dKeLCSdx,dUeLCSdx,dDLoadLCSdx,dLdx)
       additionally computes the derivatives of the element forces with
       respect to the design variables x.

   KeLCS          Element stiffness matrix (12 * 12)
   UeLCS          Displacements            (12 * nLC)
   DLoadLCS       Distributed loads        [n1localX; n1localY; n1localZ; ...]
   L              Beam length
   DLoadLCS       Distributed loads       [n1localX; n1localY; n1localZ; ...]
                                                                  (6 * 1)
   TLoadLCS       Temperature load        [dTm; dTy; dTz] (3 * 1)
   A              Cross-sectional area
   E              Young's modulus
   alpha          Linear thermal expansion coefficient
   Iyy            Area moment of inertia for bending around local y-axis
   Izz            Area moment of inertia for bending around local z-axis
   hy             Profile height (in local y-direction)
   hz             Profile height (in local z-direction)
   dKeLCSdx       Element stiffness matrix derivatives    (CELL(nVar,1))
   dUeLCSdx       Displacements derivatives               (SIZE(UeLCS) * nVar)
   dDLoadLCSdx    Distributed loads derivatives           (SIZE(DLoadLCS) * nVar)
   dLdx           Beam length derivatives                 (1 * nVar)
   Forces         Element forces                [N; Vy; Vz; T; My; Mz] (12 * nLC)
   dForcesdx      Element forces derivatives    (12 * nLC * nVar)

   See also FORCES_BEAM, FORCESLCS_TRUSS, ELEMFORCES
```

forceslcs_truss

```
FORCESLCS_TRUSS   Compute the element forces for a truss element in the LCS.

   Forces = FORCESLCS_TRUSS(KeLCS,UeLCS,dTm,A,E,alpha)
   Forces = FORCESLCS_TRUSS(KeLCS,UeLCS)
       computes the element forces for the truss element in the local coordinate
       system (algebraic convention).

   [Forces,dForcesdx] = FORCESLCS_TRUSS(KeLCS,UeLCS,[],[],[],[],dKeLCSdx,dUeLCSdx)
       additionally computes the derivatives of the element forces with
       respect to the design variables x.

   KeLCS      Element stiffness matrix (6 * 6)
   UeLCS      Displacements            (6 * nLC)
   dKeLCSdx   Element stiffness matrix derivatives    (CELL(nVar,1))
   dUeLCSdx   Displacements derivatives               (SIZE(UeLCS) * nVar)
   Forces     Element forces                          [N; 0; 0](6 * nLC)
   dForcesdx  Element forces derivatives              (6 * nLC * nVar)

   See also FORCES_TRUSS, FORCESLCS_BEAM, ELEMFORCES
```

forces_beam

```
FORCES_BEAM   Compute the element forces for a beam element.

   [ForcesLCS,ForcesGCS] = FORCES_BEAM(Node,Section,Material,UeGCS,DLoad,TLoad,Options)
   [ForcesLCS,ForcesGCS] = FORCES_BEAM(Node,Section,Material,UeGCS,DLoad)
   [ForcesLCS,ForcesGCS] = FORCES_BEAM(Node,Section,Material,UeGCS)
       computes the element forces for the beam element in the local and
       the global coordinate system (algebraic convention).

   [ForcesLCS,ForcesGCS,dForcesLCSdx,dForcesGCSdx]
           = FORCES_BEAM(Node,Section,Material,UeGCS,DLoad,TLoad,Options,dNodedx,...
                                                      dSectiondx,dUeGCSdx,dDLoaddx)
           = FORCES_BEAM(Node,Section,Material,UeGCS,[],[],Options,dNodedx,...
                                                               dSectiondx,dUeGCSdx)
       additionally computes the derivatives of the element forces with
       respect to the design variables x.

   Node            Node definitions           [x y z] (3 * 3)
   Section         Section definition         [A ky kz Ixx Iyy Izz]
   Material        Material definition        [E nu]
   UeGCS           Displacements              (12 * nLC)
   DLoad           Distributed loads       [n1globalX; n1globalY; n1globalZ; ...]
                                                                        (6 * 1)
   TLoad           TLoad                      [dTm; dTy; dTz] (3 * 1)
   Options         Element options            {Option1 Option2 ...}
   dNodedx         Node definitions derivatives        (SIZE(Node) * nVar)
   dSectiondx      Section definitions derivatives     (SIZE(Section) * nVar)
   dUeGCSdx        Displacements derivatives           (SIZE(UeGCS) * nVar)
   dDLoaddx        Distributed loads derivatives       (SIZE(DLoad) * nVar)
   ForcesLCS       Element forces in the LCS           (12 * nLC)
   ForcesGCS       Element forces in the GCS           (12 * nLC)
   dForcesLCSdx    Element forces derivatives in LCS   (12 * nLC * nVar)
   dForcesGCSdx    Element forces derivatives in GCS   (12 * nLC * nVar)

   See also FORCESLCS_BEAM, ELEMFORCES.
```

forces_shell2

```
FORCES_BEAM   Compute the element forces for a SHELL2 element.

   [ForcesLCS,ForcesGCS]=forces_shell2(Node,Section,Material,UeGCS,DLoad,TLoad,Options)
   [ForcesLCS,ForcesGCS]=forces_shell2(Node,Section,Material,UeGCS,DLoad)
   [ForcesLCS,ForcesGCS]=forces_shell2(Node,Section,Material,UeGCS)
   computes the element forces for the SHELL2 element in the local and the
   global coordinate system (algebraic convention).

   Node       Node definitions           [x y z] (3 * 3)
   Section    Section definition         [h]
   Material   Material definition        [E nu]
   UeGCS      Displacements (12 * 1)
   DLoad      Distributed loads       [n1globalX; n1globalY; n1globalZ; ...]
                                                                 (6 * 1)
   TLoad      TLoad                      [dTm; dTy; dTz] (3 * 1)
   Options    Element options            {Option1 Option2 ...}
   ForcesLCS  Element forces in the LCS  (12 * 1)
   ForcesGCS  Element forces in the GCS  (12 * 1)
```

forces_truss

```
FORCES_TRUSS   Compute the element forces for a truss element.

   [ForcesLCS,ForcesGCS] = FORCES_TRUSS(Node,Section,Material,UeGCS,[],TLoad)
   [ForcesLCS,ForcesGCS] = FORCES_TRUSS(Node,Section,Material,UeGCS)
       computes the element forces for the truss element in the local and the
       global coordinate system (algebraic convention).

   [ForcesLCS,ForcesGCS,dForcesLCSdx,dForcesGCSdx]
           = FORCES_TRUSS(Node,Section,Material,UeGCS,[],TLoad,[],dNodedx,...
                                                             dSectiondx,dUeGCSdx)
       additionally computes the derivatives of the element forces with
       respect to the design variables x.

   Node            Node definitions           [x y z] (2 * 3)
   Section         Section definition         [A]
   Material        Material definition        [E]
   UeGCS           Displacements              (6 * nLC)
   TLoad           Temperature load           [dTm]
   Options         Element options            {Option1 Option2 ...}
   dNodedx         Node definitions derivatives        (SIZE(Node) * nVar)
   dSectiondx      Section definitions derivatives     (SIZE(Section) * nVar)
   dUeGCSdx        Displacements derivatives           (SIZE(UeGCS) * nVar)
   dDLoaddx        Distributed loads derivatives       (SIZE(DLoad) * nVar)
   ForcesLCS       Element forces in the LCS           (12 * nLC)
   ForcesGCS       Element forces in the GCS           (12 * nLC)
   dForcesLCSdx    Element forces derivatives in LCS   (12 * nLC * nVar)
   dForcesGCSdx    Element forces derivatives in GCS   (12 * nLC * nVar)

   See also FORCESLCS_TRUSS, ELEMFORCES.
```

gaussq

```
GAUSSQ   Gauss points for 2D numerical integration.

   [x,H] = gaussq(n) returns the coordinates and weights for a 2D
   gauss-legendre quadrature.

   n   number of points in one direction = 2 or 3
   x   coordinates of gauss-points         (n^2 * 2)
   H   weights used in summation           (1 * n^2)

   See also KE_SHELL8, KELCS_SHELL4
```

gaussqtet

```
GAUSSQTET   Gauss points for 3D numerical integration on a tetrahedron.

   [x,H] = gaussqtet(n) returns the coordinates and weights for a 3D
   gauss-legendre quadrature on a tetrahedron. The coordinates are
   returned in natural coordinates (i.e. no volume coordinates)

   n   number of integration points
   x   coordinates of the integration points  (n * 3)
   H   weights used in summation              (1 * n)

   See also GAUSSQ, GAUSSQTRI
```

gaussqtri

```
GAUSSQTRI   Gauss points for 2D numerical integration on a triangle.

   [x,H] = gaussqtri(n) returns the coordinates and weights for a 2D
   triangular gauss-legendre quadrature.

   n   number of integration points
   x   coordinates of the integration points  (n * 2)
   H   weights used in summation              (1 * n)

   See also GAUSSQ
```

getdof

```
GETDOF   Get the vector with the degrees of freedom of the model.

   DOF=getdof(Elements,Types) builds the vector with the
   labels of the degrees of freedom for which stiffness is present in the
   finite element model.

   Elements   Element definitions       [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }
   DOF        Degrees of freedom  (nDOF * 1)

   See also DOF_TRUSS, DOF_BEAM, GETDOF.
```

getmovie

```
GETMOVIE   Get the movie from a figure where an animation has been played.

   mov=getmovie(h)
   gets the movie from the userdata of the axis of a figure where an animation
   has been played. In order to save the movie in the userdata the animation
   should have been played using animdisp(...,'CreateMovie','on'). This
   function blocks the command prompt until the movie has become available.

   h           Axis handle.
   mov         Structured array with movie frames.

   The movie can be played with movie(gcf,mov).

   See also ANIMDISP, MOVIE.
```

grid_plane4

```
GRID_PLANE4  Grid in natural coordinates for mapped meshing.

   [s,t,NodeNum,Elements] = grid_plane4(m,n,Type,Section,Material)
   returns matrices of a grid in the natural coordinate system, which can
   be used for mapped meshing.

   s           s-coordinate of nodes  (1 * nNodes)
   t           t-coordinate of nodes  (1 * nNodes)
   NodeNum     Node numbers order on grid ((m+1) * (n+1))
   Elements    Node numbers are saved per element here (nElem * 8)
   Type        Type ID of meshed elements
   Section     Section ID of meshed elements
   Material    Material ID of meshed elements

   See also MAKEMESH, GRID_SHELL8.
```

grid_shell4

```
GRID_SHELL4  Grid in natural coordinates for mapped meshing.

   [s,t,NodeNum,Elements] = grid_shell4(m,n,Type,Section,Material)
   returns matrices of a grid in the natural coordinate system, which can
   be used for mapped meshing.

   s           s-coordinate of nodes  (1 * nNodes)
   t           t-coordinate of nodes  (1 * nNodes)
   NodeNum     Node numbers order on grid ((m+1) * (n+1))
   Elements    Node numbers are saved per element here (nElem * 8)
   Type        Type ID of meshed elements
   Section     Section ID of meshed elements
   Material    Material ID of meshed elements

   See also MAKEMESH, GRID_SHELL8.
```

grid_shell8

```
GRID_SHELL8  Grid in natural coordinates for mapped meshing.

   [s,t,NodeNum,Elements] = grid_shell8(m,n,Type,Section,Material)
   returns matrices of a grid in the natural coordinate system, which can
   be used for mapped meshing.

   s           s-coordinate of nodes  (1 * nNodes)
   t           t-coordinate of nodes  (1 * nNodes)
   NodeNum     Node numbers order on grid ((m+1) * (n+1))
   Elements    Node numbers are saved per element here (nElem * 12)
   Type        Type ID of meshed elements
   Section     Section ID of meshed elements
   Material    Material ID of meshed elements

   See also MAKEMESH, GRID_SHELL4.
```

kelcs_beam

```
KELCS_BEAM   Beam element stiffness and mass matrix in local coordinate system.

   [KeLCS,MeLCS] = KELCS_BEAM(L,A,ky,kz,Ixx,Iyy,Izz,E,nu,rho,Options)
   [KeLCS,MeLCS] = KELCS_BEAM(L,A,ky,kz,Ixx,Iyy,Izz,E,nu,rho)
    KeLCS        = KELCS_BEAM(L,A,ky,kz,Ixx,Iyy,Izz,E,nu)
   returns the element stiffness and mass matrix in the local coordinate system
   for a two node beam element (isotropic material).

   [KeLCS,~,dKeLCSdx] = KELCS_BEAM(L,A,ky,kz,Ixx,Iyy,Izz,E,nu,rho,Options,dLdx,dAdx,...
                                                    dkydx,dkzdx,dIxxdx,dIyydx,dIzzdx)
   returns the element stiffness matrix in the local coordinate system
   for a two node beam element (isotropic material), and additionally
   computes the derivatives of the stiffness matrix with respect to the
   design variables x. The derivatives of the mass matrix have not yet
   been implemented.

   L           Beam length
   A           Beam cross section area
   ky          Shear deflection factor A_sy = ky * A
   kz          Shear deflection factor A_sz = kz * A
   Ixx         Moment of inertia
   Iyy         Moment of inertia
   Izz         Moment of inertia
   E           Young's modulus
   nu          Poisson coefficient
   rho         Mass density
   Options     Options for the mass matrix: {'lumped'}, {'norotaroryinertia'}
   dLdx        Beam length derivatives              (1 * nVar)
   dAdx        Beam cross section area derivatives  (1 * nVar)
   dkydx       Shear deflection factor derivatives  (1 * nVar)
   dkzdx       Shear deflection factor derivatives  (1 * nVar)
   dIxxdx      Moment of inertia derivatives        (1 * nVar)
   dIyydx      Moment of inertia derivatives        (1 * nVar)
   dIzzdx      Moment of inertia derivatives        (1 * nVar)
   KeLCS       Element stiffness matrix             (12 * 12)
   MeLCS       Element mass matrix                  (12 * 12)
   dKeLCSdx    Element stiffness matrix derivatives (CELL(nVar,1))

   See also KE_BEAM, KELCS_TRUSS.
```

kelcs_shell2

```
KELCS_SHELL2   SHELL2 element stiffness matrix in local coordinate system.

   KeLCS = kelcs_shell2(r1,phi,L,h,E,nu) returns the element stiffness matrix
   in the local coordinate system for a two-node axisymmetric shell element.
   The global y-axis is assumed to be the axis of symmetry.

   r1         Radial coordinate of node 1
   phi        Slope with respect to the xz-plane
   h          Shell thickness
   E          Young's modulus
   nu         Poisson coefficient
   KeLCS      Element stiffness matrix (6 * 6)
```

kelcs_shell4

```
KELCS_SHELL4   shell element stiffness and mass matrix in element
               coordinate system.

   [Ke,Me] = kelcs_shell4(Node_lc,h,E,nu,rho)
    Ke     = kelcs_shell4(Node_lc,h,E,nu)
   returns the element stiffness and mass matrix in the element
   coordinate system for a four node shell element (isotropic material).

   Node       Node definitions           [x y z] (4 * 3)
              Nodes should have the following order:
              4---------3
              |         |
              |         |
              |         |
              1---------2

   h          Shell thickness
   E          Young's modulus
   nu         Poisson coefficient
   rho        Mass density
   Options    Element options [NOT available yet]    {Option1 Option2 ...}
   KeLCS      Element stiffness matrix (24 * 24)
   MeLCS      Element mass matrix (24 * 24)

   This element is a flat shell element that consists of a bilinear
   membrane element and four overlaid DKT triangles for the bending
   stiffness.

   See also KE_SHELL8, ASMKM, KE_DKT.
```

kelcs_truss

```
KELCS_TRUSS   Truss element stiffness and mass matrix in local coordinate system.

   [KeLCS,MeLCS] = KELCS_TRUSS(L,A,E,rho,Options)
   [KeLCS,MeLCS] = KELCS_TRUSS(L,A,E,rho)
    KeLCS        = KELCS_TRUSS(L,A,E)
   returns the element stiffness and mass matrix in the local coordinate system
   for a two node truss element (isotropic material).

   [KeLCS,~,dKeLCSdx] = KELCS_TRUSS(L,A,E,rho,Options,dLdx,dAdx)
   returns the element stiffness matrix in the local coordinate system
   for a two node truss element (isotropic material), and additionally
   computes the derivatives of the stiffness matrix with respect to the
   design variables x. The derivatives of the mass matrix have not yet
   been implemented.

   L          Truss length
   A          Truss cross section area
   E          Young's modulus
   rho        Mass density
   Options    Options for the mass matrix: {'lumped'}
   dLdx       Truss length derivatives                (1 * nVar)
   dAdx       Truss cross section area derivatives    (1 * nVar)
   KeLCS      Element stiffness matrix                (6 * 6)
   MeLCS      Element mass matrix                     (6 * 6)
   dKeLCSdx   Element stiffness matrix derivatives    (CELL(nVar,1))

   See also KE_TRUSS, KELCS_BEAM.
```

ke_beam

```
KE_BEAM   Beam element stiffness and mass matrix in global coordinate system.

   [Ke,Me] = KE_BEAM(Node,Section,Material,Options) returns the element
   stiffness and mass matrix in the global coordinate system
   for a two node beam element (isotropic material).

   [Ke,~,dKedx] = KE_BEAM(Node,Section,Material,Options,dNodedx,dSectiondx)
   returns the element stiffness matrix in the global coordinate system
   for a two node beam element (isotropic material), and additionally
   computes the derivatives of the stiffness matrix with respect to the
   design variables x. The derivatives of the mass matrix have not yet
   been implemented.

   Node       Node definitions           [x y z] (3 * 3)
   Section    Section definition         [A ky kz Ixx Iyy Izz]
   Material   Material definition        [E nu rho]
   Options    Struct containing element options. Fields:
      .lumped            Construct lumped mass matrix:
                         {true | false (default)}
      .rotationalInertia Include rotational inertia:
                         {true (default)| false}
   dNodedx    Node definitions derivatives         (SIZE(Node) * nVar)
   dSectiondx Section definitions derivatives      (SIZE(Section) * nVar)
   Ke         Element stiffness matrix             (12 * 12)
   Me         Element mass matrix                  (12 * 12)
   dKedx      Element stiffness matrix derivatives (CELL(nVar,1))

   See also KELCS_BEAM, TRANS_BEAM, ASMKM, KE_TRUSS.
```

ke_dkt

```
KE_DKT   DKT plate element stiffness and mass matrix.

   [Ke,Me] = ke_dkt(Node,h,E,nu,rho)
    Ke     = ke_dkt(Node,h,E,nu)
   returns the element stiffness and mass matrix in the global coordinate system
   for a three node plate element (isotropic material) in the xy-plane.

   Node       Node definitions                              [x y] (3 * 2)
   h          Plate thickness (uniform or defined in nodes) [h]/[h1 h2 h3]
   E          Young's modulus
   nu         Poisson coefficient
   rho        Mass density
   Ke         Element stiffness matrix (24 * 24)
   Me         Element mass matrix (24 * 24)

   This element is a Discrete Kirchoff Triangle plate element. This code
   is a MATLAB code based on the E-2 element FORTRAN coding, described in:
   Construction of new efficient three-node triangular thin plate bending
   elements, C. Jeyachandrabose and J. Kirkhope, Computers & Structures
   Vol. 23, No. 5, pp. 587-603, 1986.

   See also Q_DKT, SH_T, KE_DKT4.
```

ke_mass

```
KE_MASS   mass element system matrices in the global coordinate system.

   [Ke,Me] = ke_mass(Node,Section,Material,Options) returns the element
   stiffness and mass matrix in the global coordinate system for a
   concentrated mass element

   Node       Node definitions           [x1 y1 z1] (1 * 3)
   Section    Section definition         [m]
   Material   Material definition        []
   Options    Element options            {Option1 Option2 ...}
   Ke         Element stiffness matrix   (6 * 6)
   Me         Element mass matrix        (6 * 6)
```

ke_plane10

```
KE_PLANE10   plane element stiffness and mass matrix in global coordinate system.

   [Ke,Me] = ke_plane10(Node,Section,Material,Options) returns the element
   stiffness and mass matrix in the global coordinate system for a 10-node
   plane triangular element. Plane10 only operates in the 2D xy-plane so that
   z-coordinates should be equal to zero.

   Node       Node definitions           [x y z] (10 * 3)
              Nodes should have the following order:

               3
               | \
               8  7
               |    \
               9 10  6
               |      \
               1--4--5--2

   Section    Section definition         [h]  (only used in plane stress)
   Material   Material definition        [E nu rho]
   Options    Struct containing optional parameters. Fields:
      .problem Plane stress or plane strain
               {'2dstress' (default) | '2dstrain'}
      .nXi     Number of Gauss integration points
               {1 | 3 | 4 | 6 | 7 (default) | 12 | 13 | 16 | 19 | 28 | 33 | 37}
   Ke         Element stiffness matrix (20 * 20)
   Me         Element mass matrix (20 * 20)

   See also KE_BEAM, ASMKM, KE_TRUSS.
```

ke_plane15

```
KE_PLANE15   plane element stiffness and mass matrix in global coordinate system.

   [Ke,Me] = ke_plane15(Node,Section,Material,Options) returns the element
   stiffness and mass matrix in the global coordinate system for a 15-node
   plane triangular element. Plane15 only operates in the 2D xy-plane so that
   z-coordinates should be equal to zero.

   Node       Node definitions           [x y z] (15 * 3)
              Nodes should have the following order:

                  3
                  | \
                 10   9
                  |     \
                 11  15   8
                  |         \
                 12  13   14  7
                  |             \
                  1---4---5---6---2

   Section    Section definition         [h] (only used in plane stress)
   Material   Material definition        [E nu rho]
   Options    Struct containing optional parameters. Fields:
      .problem Plane stress or plane strain
               {'2dstress' (default) | '2dstrain'}
      .nXi     Number of Gauss integration points
               {1 | 3 | 4 | 6 | 7  | 12 | 13 (default) | 16 | 19 | 28 | 33 | 37}
   Ke         Element stiffness matrix (30 * 30)
   Me         Element mass matrix (30 * 30)

   See also KE_BEAM, ASMKM, KE_TRUSS.
```

ke_plane3

```
KE_PLANE3   plane element stiffness and mass matrix in global coordinate system.

   [Ke,Me] = ke_plane3(Node,Section,Material,Options) returns the element
   stiffness and mass matrix in the global coordinate system for a 3-node
   CST element. Plane3 only operates in the 2D xy-plane so that
   z-coordinates should be equal to zero.

   Node       Node definitions           [x y z] (3 * 3)
   Section    Section definitions        [h] (only used in plane stress)
   Material   Material definition        [E nu rho]
   Options    Struct containing optional parameters. Fields:
      .problem Plane stress or plane strain
                {'2dstress' (default) | '2dstrain'}
   Ke         Element stiffness matrix (6 * 6)
   Me         Element mass matrix (6 * 6)
```

ke_plane4

```
KE_PLANE4   plane element stiffness and mass matrix in global coordinate system.

   [Ke,Me] = ke_plane4(Node,Section,Material,Options) returns the element
   stiffness and mass matrix in the global coordinate system for a 4-node
   plane element. Plane4 only operates in the 2D xy-plane so that
   z-coordinates should be equal to zero.

   Node       Node definitions           [x y z] (4 * 3)
              Nodes should have the following order:

                 4---3
                 |   |
                 1---2

   Section    Section definition         [h] (only used in plane stress)
   Material   Material definition        [E nu rho]
   Options    Struct containing optional parameters. Fields:
      .problem      Plane stress, plane strain or axisymmetrical
                    {'2dstress' (default) | '2dstrain' | 'axisym'}
      .bendingmodes Include (non-conforming) bending modes
                    {true (default) | false}
      .integration  Full (2x2) or reduced (1x1) integration
                    {'full' (default) | 'reduced'}
   Ke         Element stiffness matrix (8 * 8)
   Me         Element mass matrix (8 * 8)

   See also KE_BEAM, ASMKM, KE_TRUSS.
```

ke_plane6

```
KE_PLANE6   plane element stiffness and mass matrix in global coordinate system.

   [Ke,Me] = ke_plane6(Node,Section,Material,Options) returns the element
   stiffness and mass matrix in the global coordinate system for a 6-node
   plane triangular element. Plane6 only operates in the 2D xy-plane so that
   z-coordinates should be equal to zero.

   Node       Node definitions           [x y z] (6 * 3)
              Nodes should have the following order:

              3
              | \
              6  5
              |    \
              1--4--2

   Section    Section definition         [h] (only used in plane stress)
   Material   Material definition        [E nu rho]
   Options    Struct containing optional parameters. Fields:
      .problem Plane stress or plane strain
               {'2dstress' (default) | '2dstrain'}
      .nXi     Number of Gauss integration points
               {1 | 3 | 4 | 6 | 7 (default) | 12 | 13 | 16 | 19 | 28 | 33 | 37}
   Ke         Element stiffness matrix (12 * 12)
   Me         Element mass matrix (12 * 12)

   See also KE_BEAM, ASMKM, KE_TRUSS.
```

ke_plane8

```
KE_PLANE8   plane element stiffness and mass matrix in global coordinate system.

   [Ke,Me] = ke_plane8(Node,Section,Material,Options) returns the element
   stiffness and mass matrix in the global coordinate system for a 8-node
   plane element. Plane8 only operates in the 2D xy-plane so that
   z-coordinates should be equal to zero.

   Node       Node definitions           [x y z] (8 * 3)
   Section    Section definition         [h] (only used in plane stress)
   Material   Material definition        [E nu rho]
   Options    Struct containing optional parameters. Fields:
      .problem Plane stress or plane strain
                {'2dstress' (default) | '2dstrain' | 'axisym'}
   Ke         Element stiffness matrix (16 * 16)
   Me         Element mass matrix (16 * 16)
```

ke_shell2

```
KE_SHELL2   SHELL2 element stiffness matrix in global coordinate system.

   Ke = ke_shell2(Node,Section,Material) returns the element stiffness
   matrix in the global coordinate system for a two-node axisymmetric shell
   element. The global y-axis is assumed to be the axis of symmetry.

   Node       Node definitions           [x y z] (3 * 3)
   Section    Section definition         [h]
   Material   Material definition        [E nu]
   Ke         Element stiffness matrix (6 * 6)
```

ke_shell4

```
KE_SHELL4   shell element stiffness and mass matrix in global coordinate system.

   [Ke,Me] = ke_shell4(Node,Section,Material,Options)
    Ke     = ke_shell4(Node,Section,Material,Options)
   returns the element stiffness and mass matrix in the global coordinate system
   for a four node shell element (isotropic material).

   Node       Node definitions           [x y z] (4 * 3)
              Nodes should have the following order:
              4---------3
              |         |
              |         |
              |         |
              1---------2

   Section    Section definition         [h]
   Material   Material definition        [E nu rho]
   Options    Element options            {Option1 Option2 ...}
   Ke         Element stiffness matrix (24 * 24)
   Me         Element mass matrix (24 * 24)

   This shell element consists of a bilinear membrane element and
   four overlaid DKT triangles for the bending stiffness.

   See also KE_BEAM, ASMKM, KE_TRUSS.
```

ke_shell6

```
KE_SHELL6   shell element stiffness and mass matrix in global coordinate system.

   [Ke,Me] = ke_shell6(Node,Section,Material,Options)
    Ke     = ke_shell6(Node,Section,Material,Options)
   returns the element stiffness and mass matrix in the global coordinate system
   for an eight node shell element.

   Node       Node definitions           [x y z] (6 * 3)
              Nodes should have the following order:
              3
              | \
              6  5
              |    \
              1--4--2

   Section    Section definition         [h] or [h1 h2 h3]
              (uniform thickness or defined in corner nodes(1,2,3))
   Material   Material definition        [E nu rho] or
                                         [Exx Eyy nuxy muxy muyz muzx theta rho]
   Options    Element options struct. Fields:
              -LCSType: determine the reference local element
                        coordinate system. Values:
                        'element' (default) or 'global'
              -MatType: 'isotropic' (default) or 'orthotropic'
              -Offset: nodal offset from shell midplane. Values:
                      'top', 'mid' (default), 'bot' or numerical value
   Ke         Element stiffness matrix (36 * 36)
   Me         Element mass matrix (36 * 36)

   This element is based on chapter 15 of
   The Finite Element Method: for Solid and Structural Mechanics,
   Zienkiewicz (2005).

   See also KE_BEAM, ASMKM, KE_TRUSS.
```

ke_shell8

```
KE_SHELL8   Shell element stiffness and mass matrix in global coordinate system.

   [Ke,Me] = ke_shell8(Node,Section,Material,Options)
    Ke     = ke_shell8(Node,Section,Material,Options)
   returns the element stiffness and mass matrix in the global coordinate system
   for an eight node shell element.

   Node       Node definitions           [x y z] (8 * 3)
              Nodes should have the following order:
              4----7----3
              |         |
              8         6
              |         |
              1----5----2

   Section    Section definition         [h] or [h1 h2 h3 h4]
              (uniform thickness or defined in corner nodes(1,2,3,4))
   Material   Material definition        [E nu rho] or
                                         [Exx Eyy nuxy muxy muyz muzx theta rho]
   Options    Element options struct. Fields:
              -LCSType: determine the reference local element
                        coordinate system. Values:
                        'element' (default) or 'global'
              -MatType: 'isotropic' (default) or 'orthotropic'
              -Offset: nodal offset from shell midplane. Values:
                      'top', 'mid' (default), 'bot' or numerical value
              -RotaryInertia: 0 (default) or 1
   Ke         Element stiffness matrix (48 * 48)
   Me         Element mass matrix (48 * 48)

   This element is based on chapter 15 of
   The Finite Element Method: for Solid and Structural Mechanics,
   Zienkiewicz (2005).

   See also KE_BEAM, ASMKM, KE_TRUSS.
```

ke_solid10

```
KE_SOLID10   Compute the element stiffness and mass matrix for a solid10 element.

   [Ke,Me] = ke_solid10(Node,Section,Material,Options) computes element
   stiffness and mass matrix in the global coordinate system for a solid10
   element.

   Node       Node definitions           [x y z] (10 * 3)
              Nodes should have the following order:

                    4
                  / | \
                 8  | 10
                /   9   \
               1--- |-7--3
                \   |   /
                 5  |  6
                  \ | /
                    2

   Section    Section definition         []
   Material   Material definition        [E nu rho]
   Options    Struct containing optional parameters. Fields:
      .nXi Number of Gauss integration points
            {1 | 4 (default) | 5}
   Ke         Element stiffness matrix (30 * 30)
   Me         Element mass matrix (30 * 30)

   See also KE_BEAM, ASMKM, KE_TRUSS.
```

ke_solid15

```
KE_SOLID15   Compute the element stiffness and mass matrix for a solid15 element.

   [Ke,Me] = ke_solid15(Node,Section,Material,Options)   computes element
   stiffness and mass matrix in the global coordinate system for a solid15
   prismatic element.

   Node       Node definitions           [x y z] (15 * 3)
              Nodes should have the following order:
                    6
                  / | \
                12  |  11
                /  15   \
               /    |    \
              4 ---10 ----5
              |     3     |
              |   /   \   |
             13  9     8  14
              | /       \ |
              |/         \|
              1 --- 7 --- 2

   Section    Section definition         []
   Material   Material definition        [E nu rho]
   Options    Element options struct. Fields: []
   Ke         Element stiffness matrix (45 * 45)
   Me         Element mass matrix (45 * 45)
```

ke_solid20

```
KE_SOLID20   Compute the element stiffness and mass matrix for a solid20 element.

   [Ke,Me] = ke_solid20(Node,Section,Material,Options)   computes element
   stiffness and mass matrix in the global coordinate system for a solid20
   element.

   Node       Node definitions           [x y z] (20 * 3)
              Nodes should have the following order:
                   8---15----7
                  /|        /|
                16 |      14 |
                / 20      / 19
               /   |     /   |
              5---13----6    |
              |    4--11|----3
              |   /     |   /
             17 12     18  10
              | /       | /
              |/        |/
              1----9----2

   Section    Section definition         []
   Material   Material definition        [E nu rho]
   Options    Element options struct. Fields: []
   Ke         Element stiffness matrix (60 * 60)
   Me         Element mass matrix (60 * 60)

   See also KE_BEAM, ASMKM, KE_TRUSS.
```

ke_solid4

```
KE_SOLID4   Compute the element stiffness and mass matrix for a solid4 element.

   [Ke,Me] = ke_solid4(Node,Section,Material,Options) computes element
   stiffness and mass matrix in the global coordinate system for a solid4
   element.

   Node       Node definitions           [x y z] (4 * 3)
              Nodes should have the following order:

                    4
                  / | \
                 /  |  \
                /   |   \
               1--- |----3
                \   |   /
                 \  |  /
                  \ | /
                    2

   Section    Section definition         []
   Material   Material definition        [E nu rho]
   Options    Element options struct. Fields: []
   Ke         Element stiffness matrix (12 * 12)
   Me         Element mass matrix (12 * 12)

   See also KE_BEAM, ASMKM, KE_TRUSS.
```

ke_solid8

```
KE_SOLID8   Compute the element stiffness and mass matrix for a solid8 element.

   [Ke,Me] = ke_solid8(Node,Section,Material,Options)
   [Ke,Me] = ke_solid8(Node,Section,Material,Options)
   computes element stiffness and mass matrix in the global coordinate system
                                                           for a solid8 element.

   Node       Node definitions           [x y z] (8 * 3)
              Nodes should have the following order:
                  8---------7
                 /|        /|
                / |       / |
               /  |      /  |
              5---------6   |
              |   4-----|---3
              |  /      |  /
              | /       | /
              |/        |/
              1---------2

   Section    Section definition         []
   Material   Material definition        [E nu rho]
   Options    Element options struct. Fields: []
   Ke         Element stiffness matrix (24 * 24)
   Me         Element mass matrix (24 * 24)

   See also KE_BEAM, ASMKM, KE_TRUSS.
```

ke_truss

```
KE_TRUSS   Truss element stiffness and mass matrix in global coordinate system.

   [Ke,Me] = KE_TRUSS(Node,Section,Material,Options) returns the element
   stiffness and mass matrix in the global coordinate system
   for a two node truss element (isotropic material).

   [Ke,~,dKedx] = KE_TRUSS(Node,Section,Material,Options,dNodedx,dSectiondx)
   returns the element stiffness matrix in the global coordinate system
   for a two node truss element (isotropic material), and additionally
   computes the derivatives of the stiffness matrix with respect to the
   design variables x. The derivatives of the mass matrix have not yet
   been implemented.

   Node       Node definitions           [x1 y1 z1; x2 y2 z2] (2 * 3)
   Section    Section definition         [A]
   Material   Material definition        [E nu rho]
   Options    Struct containing element options. Fields:
      .lumped Construct lumped mass matrix:
              {true | false (default)}
   dNodedx    Node definitions derivatives     (SIZE(Node) * nVar)
   dSectiondx Section definitions derivatives  (SIZE(Section) * nVar)
   Ke         Element stiffness matrix         (6 * 6)
   Me         Element mass matrix              (6 * 6)
   dKedx      Element stiffness matrix         (CELL(nVar,1))

   See also KELCS_TRUSS, TRANS_TRUSS, ASMKM, KE_BEAM.
```

lconstr

```
LCONSTR   Add linear constraint equations to the stiffness matrix and load vector
                                                              using Lagrange multipliers.

   [L,K,F]=lconstr(Constr,DOF,K,F)
   [L,K,F,M]=lconstr(Constr,DOF,K,[],M)
   [L,K,F,M]=lconstr(Constr,DOF,K,F,M)
   modifies the stiffness matrix, the mass matrix and the load vector according
   to the applied constraint equations. The dimensions of the stiffness matrix,
   the mass matrix and the load vector increase with the number of constraints.
   The resulting stiffness and mass matrix are symmetric.

   Constr      Constraint equation:
                Constant=CoefS*SlaveDOF+CoefM1*MasterDOF1+CoefM2*MasterDOF2+...
               [Constant CoefS SlaveDOF CoefM1 MasterDOF1 CoefM2 MasterDOF2 ...]
   DOF         Degrees of freedom  (nDOF * 1)
   L           Selection matrix for displacements: U = L*(K\F);     (nDOF * (nDOF+nConstr))
   K           Stiffness matrix (nDOF * nDOF)
   F           Load vector  (nDOF * nSteps)
   M           Mass matrix (nDOF * nDOF)
```

linkandcheck

```
LINKANDCHECK   Link two matrices and check for presence.

   p3=linkandcheck(p1,p2,name)
   Link two matrices and check for presence

   p1     Vector with elements to be link and checked
   p2     Vector with elements to be link and checked
   name   Name of elements in p2
   p3     Lowest absolute index in p2 for each element in p1
```

loadslcs_beam

```
LOADSLCS_BEAM   Equivalent nodal forces for a beam element in the LCS.

   FLCS = LOADSLCS_BEAM(DLoadLCS,L)
   computes the equivalent nodal forces of a distributed load
   (in the local coordinate system).

   [FLCS,dFLCSdx] = LOADSLCS_BEAM(DLoadLCS,L,dDLoadLCSdx,dLdx)
   additionally computes the derivatives of the equivalent nodal forces
   with respect to the design variables x.

   DLoadLCS     Distributed loads        [n1localX; n1localY; n1localZ; ...]  (6/8 * nLC)
   L            Beam length
   dDLoadLCSdx  Distributed loads derivatives (6/8 * nLC * nVar)
   dLdx         Beam length derivatives        (1 * nVar)
   FLCS         Load vector                    (12 * nLC)
   dFLCSdx      Load vector derivatives        (12 * nLC * nVar)

   See also LOADS_BEAM.
```

loadslcs_shell2

```
LOADSLCS_SHELL2   Equivalent nodal forces for a SHELL2 element in the LCS.

   FLCS = loadslcs_shell2(DLoadLCS,L)
   computes the equivalent nodal forces of a distributed load
   (in the local coordinate system).

   DLoadLCS   Distributed loads        [n1localX; n1localY; n1localZ; ...]
                                                                  (6 * 1)
   L          Element length
   FLCS       Load vector  (12 * 1)
```

loadslcs_shell4

```
LOADSLCS_SHELL4   Equivalent nodal forces for a shell4 element in the LCS.

   F = loadslcs_shell4(DLoadLCS,Node)
   computes the equivalent nodal forces of a distributed load
   (in the local coordinate system).

   DLoadLCS   Distributed loads      [n1localX; n1localY; n1localZ; ...]
              in corner Nodes         (12 * 1)
   Node       Node definitions       [x y z] (4 * 3)
   FLCS       Load vector  (24 * 1)

   See also LOADSLCS_BEAM, ELEMLOADS, LOADS_TRUSS.
```

loads_beam

```
LOADS_BEAM   Equivalent nodal forces for a beam element in the GCS.

   F = LOADS_BEAM(DLoad,Node)
   computes the equivalent nodal forces of a distributed load
   (in the global coordinate system).

   [F,dFdx] = LOADS_BEAM(DLoad,Node,dDLoaddx,dNodedx)
   additionally computes the derivatives of the equivalent nodal forces
   with respect to the design variables x.

   DLoad      Distributed loads    [n1globalX; n1globalY; n1globalZ; ...]
                   (6/8 * nLC * nDLoads)
   Node       Node definitions     [x y z] (3 * 3)
   dDLoaddx   Distributed loads derivatives    (6/8 * nLC * nVar * nDLoads)
   dNodedx    Node definitions derivatives     (SIZE(Node) * nVar)
   F          Load vector              (12 * nLC)
   dFdx       Load vector derivatives  (12 * nLC * nVar)

   See also LOADSLCS_BEAM, ELEMLOADS, LOADS_TRUSS.
```

loads_shell2

```
LOADS_SHELL2   Equivalent nodal forces for a SHELL2 element in the GCS.

   F = loads_shell2(DLoad,Node)
   computes the equivalent nodal forces of a distributed load
   (in the global coordinate system).

   DLoad      Distributed loads     [n1globalX; n1globalY; n1globalZ; ...]
                                                                 (6 * 1)
   Node       Node definitions           [x y z] (3 * 3)
   F          Load vector  (6 * 1)
```

loads_shell4

```
LOADS_SHELL4   Equivalent nodal forces for a shell4 element in the GCS.

   F = loads_shell4(DLoad,Node)
   computes the equivalent nodal forces of a distributed load
   (in the global coordinate system).

   DLoad      Distributed loads      [n1globalX; n1globalY; n1globalZ; ...]
              in corner Nodes         (12 * 1)
   Node       Node definitions       [x y z] (4 * 3)
   F          Load vector  (24 * 1)

   See also LOADSLCS_BEAM, ELEMLOADS, LOADS_TRUSS.
```

loads_shell6

```
LOADS_SHELL6   Equivalent nodal forces for a shell6 element in the GCS.

   F = loads_shell6(DLoad,Node)
   computes the equivalent nodal forces of a distributed load
   (in the global coordinate system).

   DLoad      Distributed loads      [n1globalX; n1globalY; n1globalZ; ...]
              in corner Nodes         (9 * 1)
   Node       Node definitions       [x y z] (6 * 3)
   F          Load vector  (36 * 1)

   See also LOADSLCS_BEAM, ELEMLOADS, LOADS_TRUSS.
```

loads_shell8

```
LOADS_SHELL8   Equivalent nodal forces for a shell8 element in the GCS.

   F = loads_shell8(DLoad,Node)
   computes the equivalent nodal forces of a distributed load
   (in the global coordinate system).

   DLoad      Distributed loads      [n1globalX; n1globalY; n1globalZ; ...]
              in corner Nodes         (12 * 1)
   Node       Node definitions       [x y z] (8 * 3)
   F          Load vector  (48 * 1)

   See also LOADSLCS_BEAM, ELEMLOADS, LOADS_TRUSS.
```

loads_solid20

```
LOADS_SOLID20   Equivalent nodal forces for a solid20 element.

   F = loads_solid20(DLoad,Node)
   computes the equivalent nodal forces of a distributed load
   (in the global coordinate system).

   DLoad      Distributed loads      [n1globalX; n1globalY; n1globalZ; ...]
              in corner Nodes                       (24 * 1)
   Node       Node definitions       [x y z] (8 * 3)
   F          Load vector  (24 * 1)

   See also LOADSLCS_BEAM, ELEMLOADS, LOADS_TRUSS.
```

loads_solid8

```
LOADS_SOLID8   Equivalent nodal forces for a solid8 element.

   F = loads_solid8(DLoad,Node)
   computes the equivalent nodal forces of a distributed load
   (in the global coordinate system).

   DLoad      Distributed loads      [n1globalX; n1globalY; n1globalZ; ...]
              in Nodes                       (24 * 1)
   Node       Node definitions       [x y z] (8 * 3)
   F          Load vector  (24 * 1)

   See also LOADSLCS_BEAM, ELEMLOADS, LOADS_TRUSS.
```

loads_truss

```
LOADS_TRUSS   Equivalent nodal forces for a truss element in the GCS.

   F = LOADS_TRUSS(DLoad,Node)
   computes the equivalent nodal forces of a distributed load
   (in the global coordinate system).

   [F,dFdx] = LOADS_TRUSS(DLoad,Node,dDLoaddx,dNodedx)
   additionally computes the derivatives of the equivalent nodal forces
   with respect to the design variables x.

   DLoad      Distributed load          [n1globalX n1globalY n1globalZ ...]
                   (6/8 * nLC)
   Node       Node definitions          [x y z] (2 * 3)
   dDLoaddx   Distributed load derivatives     (6/8 * nLC * nVar)
   dNodedx    Node definitions derivatives     (SIZE(Node) * nVar)
   F          Load vector              (6 * nLC)
   dFdx       Load vector derivatives  (6 * nLC * nVar)

   See also ELEMLOADS, LOADS_BEAM.
```

lsolver

```
LSOLVER   Solves linear system of equations.

   [U,fK] = lsolver(K,P,save_factor,check_crpoint,fK)
   solves the linear system K*U=P.

   K     Coefficient matrix [n * n]
   P    Right-hand side      [n * 1]
   save_factor Save factorization {true | false}
   check_crpoint Check critical points {true | false}
   fK   Struct containing factorization of K
```

meshcat

```
eshcat is a function.
   [Nodes, Elements] = meshcat(varargin)
```

msupf

```
MSUPF   Modal superposition in the frequency domain.

   [X,H] = MSUPF(omega,xi,Omega,Pm) calculates the modal transfer functions H
   and the modal displacements x(t) = X * EXP(i * Omega * t) of a dynamic
   system with the eigenfrequencies omega and the modal damping ratios xi due
   to the modal excitation pm(t) = Pm * EXP(i * Omega * t).

   omega   Eigenfrequencies [rad/s] (nMode * 1)
   xi      Modal damping ratios [ - ] (nMode * 1) or (1 * 1), constant modal
           damping is assumed in the (1 * 1) case.
   Omega   Excitation frequencies [rad/s] (1 * N).
   Pm      Complex amplitude of the modal excitation (nMode * N).
   X       Complex amplitude of the modal displacements (nMode * N).
   H       Modal transfer functions (nMode * N).
```

msupt

```
MSUPT   Modal superposition in the time domain.

   x = MSUPT(omega,xi,t,pm,x1,y1,interp) calculates the modal displacements
   x(t) of a dynamic system with the eigenfrequencies omega and the modal
   damping ratios xi due to the modal excitation pm(t), given the initial
   conditions x1 and y1.
   The solution of the modal differential equations is performed by means of
   the piecewise exact method.  Interp can be 'foh' (first order hold) or 'zoh'
   (zero order hold).  In the first case the excitation is assumed to vary
   linearly within each time step while in the second case it is assumed to be
   constant: if t(k) <= t < t(k+1) then p(t) = p(k).

   omega   Eigenfrequencies [rad/s] (nMode * 1)
   xi      Modal damping ratios [ - ] (nMode * 1) or (1 * 1), constant modal
           damping is assumed in the (1 * 1) case.
   t       Time points (1 * N) of the sampling of p, x and t.
   pm      Modal excitation (nMode * N).
   x1      Modal displacements at time point t(1) (nMode * 1), defaults to zero.
   y1      Modal velocities at time point t(1) (nMode * 1), defaults to zero.
   interp  Interpolation scheme: 'foh' or 'zoh', defaults to 'foh'.
   x       Modal displacements (nMode * N).
```

multdloads

```
MULTDLOADS   Combine distributed loads.

   DLoads = MULTDLOADS(DLoads_1,DLoads_2,...,DLoads_k)
   combines the distributed loads of multiple load cases into one 3D array.
   Each 3D-plane corresponds to a single load case. In the presence of
   partial distributed loads, every row will have identical starting
   and ending points in the 3rd dimension to allow for accurate combination
   of load cases. A distributed load with a starting and/or ending point
   value of 'NaN' will be considered as a load on the entire element
   Calculations will be more efficient compared to a partial distributed
   load with starting point equal to zero and ending point equal to the
   length of the element.

   DLoads_k   Distributed loads       [EltID n1globalX n1globalY n1globalZ ...]
                                         without partial DLoads:  (nElem_k * 7)
                                         with partial DLoads:     (nElem_k * 9)
   DLoads     Distributed loads       [EltID n1globalX n1globalY n1globalZ ...]
                                                           (maxnElem * 7 * k)
                                                           (maxnElem * 9 * k)


   See also ELEMLOADS.
```

nedloadlcs_beam

```
NEDLOADLCS_BEAM    Interpolation functions for a distributed load on a beam element.

   NeLCS = NEDLOADLCS_BEAM(Points)
   NeLCS = NEDLOADLCS_BEAM(Points,phi_y,phi_z)
   determines the values of the interpolation functions for a distributed
   load in the specified points (LCS). These are used to compute the
   displacements that occur due to the distributed loads if all nodes are
   fixed.

   NeLCS = NEDLOADLCS_BEAM(Points,[],[],a,b,L)
   NeLCS = NEDLOADLCS_BEAM(Points,phi_y,phi_z,a,b,L)
   determines the values of the interpolation functions for a partial
   distributed load in the specified points (LCS). The load starts at a
   distance 'a' and ends at distance 'b' from the first node of the element:
       1) from 0 to a: no DLoad,
       2) from a to b: DLoad,
       3) from b to L: no DLoad.

   [NeLCS,dNeLCSdx] = NEDLOADLCS_BEAM(Points,[],[],a,b,L,dadx,dbdx,dLdx)
   [NeLCS,dNeLCSdx] = NEDLOADLCS_BEAM(Points,phi_y,phi_z,a,b,L,dadx,dbdx,dLdx)
   additionally computes the derivatives of the interpolation functions of
   a partial distributed load with respect to the design variables x.
   Note: the derivatives of the interpolation functions for a distributed
   load on the complete element are zero.

   Points     Points in the local coordinate system (1 * nPoints)
   phi_y      Shear deformation constant in y dir   (scalar)
   phi_z      Shear deformation constant in z dir   (scalar)
   a          Local starting point for the DLoad    (scalar)
   b          Local ending point for the DLoad      (scalar)
   L          Element length                        (scalar)
   dadx       Local starting point derivatives      (1 * nVar)
   dbdx       Local ending point derivatives        (1 * nVar)
   dLdx       Element length derivatives            (1 * nVar)
   NeDLoad    Interpolated values                   (nPoints * 6)
   dNeDLoaddx Interpolated values derivatives       (nPoints * 6 *nVar)

   See also DISP_BEAM, NELCS_BEAM.
```

nelcs_beam

```
NELCS_BEAM   Shape functions for a beam element.

   NeLCS = nelcs_beam(Points)
   NeLCS = nelcs_beam(Points,phi_y,phi_z)
   determines the values of the shape functions in the specified points.

   Points     Points in the local coordinate system (1 * nPoints).
   phi_y      Shear deformation constant in the y-direction (1 * 1).
   phi_z      Shear deformation constant in the z-direction (1 * 1).
   NeLCS      Values (nPoints * 12).

   See also DISP_BEAM.
```

newmark

```
NEWMARK   Direct time integration for dynamic systems - Newmark method
   [u,v,a,t] = NEWMARK(M,C,K,dt,p,u0,v0,a0,[alpha delta]) applies the Newmark
   method for the calculation of the nodal displacements u, velocities v and
   accelerations a of the dynamic system with the system matrices M, C and K
   due to the excitation p.

   M    Mass matrix (nDof * nDof)
   C    Damping matrix (nDof * nDof)
   K    Stiffness matrix (nDof * nDof)
   dt   Time step of the integration scheme (1 * 1).  Should be small enough
        to ensure the stability and the precision of the integration scheme.
   p    Excitation (nDof * N).  p(:,k) corresponds to time point t(k).
   u0   Displacements at time point t(1)-dt (nDof * 1).  Defaults to zero.
   v0   Velocities at time point t(1)-dt (nDof * 1).  Defaults to zero.
   a0   Accelerations at time point t(1)-dt (nDof * 1).  Defaults to zero.
   u    Displacements (nDof * N).  u(:,k) corresponds to time point t(k).
   t    Time axis (1 * N), defined as t = [0:N-1] * dt.
```

nodalshellf

```
NODALSHELLF   Compute the nodal shell forces/moments per unit length
                                                  from the element solution.

   [FnLCS,FnLCS2] = nodalshellf(Nodes,Elements,Types,FeLCS) computes the
   nodal forces from the element solution.

   Nodes      Node definitions          [NodID x y z]
   Elements   Element definitions       [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }
   Sections   Section definitions       [SecID SecProp1 SecProp2 ...]
   FeLCS      Element forces/moments per unit length in corner nodes IJKL
              (nElem * 32)              [Nx Ny Nxy Mx My Mxy Vx Vy]
   FnLCS      Nodal forces/moments per unit length in corner nodes IJKL
              (nElem * 32)              [Nx Ny Nxy Mx My Mxy Vx Vy]
   FnLCS2      Nodal forces per node (nNodes * 9)
              (9 = NodID + 8 fcomp.)
                                        [NodID Nx Ny Nxy Mx My Mxy Vx Vy]
   See also ELEMSHELLF.
```

nodalstress

```
NODALSTRESS   Compute the nodal stresses from the element solution.

   [Sn,Sn2] = nodalstress(Nodes,Elements,Types,Se)
   computes the nodal stresses from the element solution.

   Nodes      Node definitions          [NodID x y z]
   Elements   Element definitions       [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }
   Sections   Section definitions       [SecID SecProp1 SecProp2 ...]
   Se         Element stresses in corner nodes IJKL and
              at top/mid/bot of shell (nElem * 72)
              72 = 6 stresscomp. * 4 nodes * 3 locations
                                        [sxx syy szz sxy syz sxz ...]
   Sn         Nodal stress in corner nodes IJKL and
              at top/mid/bot of shell (nElem * 72)
              72 = 6 stresscomp. * 4 nodes * 3 locations
                                        [sxx syy szz sxy syz sxz ...]
   Sn2        Nodal stresses per node
              at top/mid/bot of shell (nNodes * 19)
              (19 = NodID + 3 locations * 6 scomp.)
                                        [NodID sxx syy szz sxy syz sxz ...]
   See also ELEMSTRESS.
```

nodalvalues

```
NODALVALUES   Construct a vector with the values at the selected DOF.

   F=nodalvalues(DOF,seldof,values)
   constructs a vector with the values at the selected DOF. This function can
   be used to obtain a load vector, initial displacements, velocities or
   accelerations.

   DOF        Degrees of freedom  (nDOF * 1)
   seldof     Selected degrees of freedom    [NodID.dof] (nValues * 1)
   values     Corresponding values           [Value]     (nValues * nSteps)
   F          Load vector  (nDOF * nSteps)

   See also ELEMLOADS.
```

patch_beam

```
PATCH_BEAM  Patch information of the beam elements for plotting.

   [pxyz,pind,pvalue] = patch_beam(Nodes,NodeNum,Values) returns matrices
   to plot patches of beam elements.

   Nodes      Node definitions [NodID x y z].
   NodeNum    Node numbers [NodID1 NodID2 NodID3] (nElem * 3).
   Values     Values assigned to nodes used for coloring (nElem * 3).
   pxyz       Coordinates of Nodes  (3*nElem * 3).
   pind       Indices of Nodes  (nElem * 3).
   pvalue     Values arranged per Node (3*nElem * 1).
```

patch_mass

```
PATCH_MASS  Patch information of the mass elements for plotting.

   [pxyz,pind,pvalue] = patch_mass(Nodes,NodeNum,Values) returns matrices
   to plot patches of mass elements.

   Nodes      Node definitions [NodID x y z].
   NodeNum    Node numbers [NodID1 NodID2 NodID3] (nElem).
   Values     Values assigned to nodes used for coloring (nElem).
   pxyz       Coordinates of Nodes  (nElem * 3).
   pind       Indices of Nodes  (nElem).
   pvalue     Values arranged per Node (nElem).
```

patch_plane3

```
PATCH_PLANE4  Patch information of the plane4 elements for plotting.

   [pxyz,pind,pvalue] = patch_plane4(Nodes,NodeNum,Values) returns matrices
   to plot patches of plane4 elements.

   Nodes      Node definitions        [NodID x y z]
   NodeNum    Node numbers       [NodID1 NodID2 NodID3 NodID4] (nElem * 4)
   Values     Values assigned to nodes used for coloring    (nElem * 4)
   pxyz       Coordinates of Nodes                          (4*nElem * 3)
   pind       indices of Nodes                                (nElem * 4)
   pvalue     Values arranged per Node                      (4*nElem * 1)

   See also PLOTSTRESSCONTOURF, PLOTSHELLFCONTOURF.
```

patch_plane4

```
PATCH_PLANE4  Patch information of the plane4 elements for plotting.

   [pxyz,pind,pvalue] = patch_plane4(Nodes,NodeNum,Values) returns matrices
   to plot patches of plane4 elements.

   Nodes      Node definitions        [NodID x y z]
   NodeNum    Node numbers       [NodID1 NodID2 NodID3 NodID4] (nElem * 4)
   Values     Values assigned to nodes used for coloring    (nElem * 4)
   pxyz       Coordinates of Nodes                          (4*nElem * 3)
   pind       Indices of Nodes                                (nElem * 4)
   pvalue     Values arranged per Node                      (4*nElem * 1)

   See also PLOTSTRESSCONTOURF, PLOTSHELLFCONTOURF.
```

patch_plane6

```
PATCH_PLANE4  Patch information of the plane4 elements for plotting.

   [pxyz,pind,pvalue] = patch_plane4(Nodes,NodeNum,Values) returns matrices
   to plot patches of plane4 elements.

   Nodes      Node definitions        [NodID x y z]
   NodeNum    Node numbers       [NodID1 NodID2 NodID3 NodID4] (nElem * 4)
   Values     Values assigned to nodes used for coloring    (nElem * 4)
   pxyz       Coordinates of Nodes                          (4*nElem * 3)
   pind       indices of Nodes                                (nElem * 4)
   pvalue     Values arranged per Node                      (4*nElem * 1)

   See also PLOTSTRESSCONTOURF, PLOTSHELLFCONTOURF.
```

patch_plane8

```
PATCH_PLANE4  Patch information of the plane4 elements for plotting.

   [pxyz,pind,pvalue] = patch_plane4(Nodes,NodeNum,Values) returns matrices
   to plot patches of plane4 elements.

   Nodes      Node definitions        [NodID x y z]
   NodeNum    Node numbers       [NodID1 NodID2 NodID3 NodID4] (nElem * 4)
   Values     Values assigned to nodes used for coloring    (nElem * 4)
   pxyz       Coordinates of Nodes                          (4*nElem * 3)
   pind       indices of Nodes                                (nElem * 4)
   pvalue     Values arranged per Node                      (4*nElem * 1)

   See also PLOTSTRESSCONTOURF, PLOTSHELLFCONTOURF.
```

patch_shell4

```
PATCH_SHELL4  Patch information of the shell4 elements for plotting.

   [pxyz,pind,pvalue] = patch_shell4(Nodes,NodeNum,Values) returns matrices
   to plot patches of shell4 elements.

   Nodes      Node definitions        [NodID x y z]
   NodeNum    Node numbers       [NodID1 NodID2 NodID3 NodID4] (nElem * 4)
   Values     Values assigned to nodes used for coloring    (nElem * 4)
   pxyz       Coordinates of Nodes                          (4*nElem * 3)
   pind       indices of Nodes                                (nElem * 4)
   pvalue     Values arranged per Node                      (4*nElem * 1)

   See also PLOTSTRESSCONTOURF, PLOTSHELLFCONTOURF.
```

patch_shell6

```
PATCH_SHELL6  Patch information of the shell6 elements for plotting.

   [pxyz,pind,pvalue] = patch_shell6(Nodes,NodeNum,Values) returns matrices
   to plot patches of shell6 elements.

   Nodes      Node definitions        [NodID x y z]
   NodeNum    Node numbers       [NodID1 NodID2 NodID3 NodID4] (nElem * 6)
   Values     Values assigned to nodes used for coloring    (nElem * 6)
   pxyz       Coordinates of Nodes                          (6*nElem * 3)
   pind       indices of Nodes                                (nElem * 8)
   pvalue     Values arranged per Node                      (6*nElem * 1)

   See also PLOTSTRESSCONTOURF, PLOTSHELLFCONTOURF.
```

patch_shell8

```
PATCH_SHELL8  Patch information of the shell8 elements for plotting.

   [pxyz,pind,pvalue] = patch_shell8(Nodes,NodeNum,Values)
   returns matrices to plot patches of shell8 elements.

   Nodes      Node definitions        [NodID x y z]
   NodeNum    Node numbers       [NodID1 NodID2 NodID3 NodID4] (nElem * 8)
   Values     Values assigned to nodes used for coloring    (nElem * 8)
   pxyz       Coordinates of Nodes                          (8*nElem * 3)
   pind       indices of Nodes                                (nElem * 8)
   pvalue     Values arranged per Node                      (8*nElem * 1)

   See also PLOTSTRESSCONTOURF, PLOTSHELLFCONTOURF.
```

patch_solid10

```
PATCH_SOLID20  Patch information of the solid10 elements for plotting.

   [pxyz,pind,pvalue] = patch_solid20(Nodes,NodeNum,Values)
   returns matrices to plot patches of solid10 elements.

   Nodes      Node definitions        [NodID x y z]
   NodeNum    Node numbers       [NodID1 NodID2 NodID3 NodID4] (nElem * 10)
   Values     Values assigned to nodes used for coloring    (nElem * 4)
   pxyz       Coordinates of Nodes                          (4*nElem * 3)
   pind       indices of Nodes                                (nElem * 4)
   pvalue     Values arranged per Node                      (4*nElem * 1)

   See also PATCH_SHELL20.
```

patch_solid20

```
PATCH_SOLID20  Patch information of the solid20 elements for plotting.

   [pxyz,pind,pvalue] = patch_solid20(Nodes,NodeNum,Values)
   returns matrices to plot patches of solid20 elements.

   Nodes      Node definitions        [NodID x y z]
   NodeNum    Node numbers       [NodID1 NodID2 NodID3 NodID4] (nElem * 20)
   Values     Values assigned to nodes used for coloring    (nElem * 8)
   pxyz       Coordinates of Nodes                          (8*nElem * 3)
   pind       indices of Nodes                                (nElem * 8)
   pvalue     Values arranged per Node                      (8*nElem * 1)

   See also PATCH_SHELL8.
```

patch_solid4

```
PATCH_SOLID4  Patch information of the solid4 elements for plotting.

   [pxyz,pind,pvalue] = patch_solid20(Nodes,NodeNum,Values)
   returns matrices to plot patches of solid4 elements.

   Nodes      Node definitions        [NodID x y z]
   NodeNum    Node numbers       [NodID1 NodID2 NodID3 NodID4] (nElem * 10)
   Values     Values assigned to nodes used for coloring    (nElem * 4)
   pxyz       Coordinates of Nodes                          (4*nElem * 3)
   pind       indices of Nodes                                (nElem * 4)
   pvalue     Values arranged per Node                      (4*nElem * 1)

   See also PATCH_SHELL20.
```

patch_solid8

```
PATCH_SOLID8  Patch information of the solid8 elements for plotting.

   [pxyz,pind,pvalue] = patch_solid8(Nodes,NodeNum,Values)
   returns matrices to plot patches of solid8 elements.

   Nodes      Node definitions        [NodID x y z]
   NodeNum    Node numbers       [NodID1 NodID2 NodID3 NodID4] (nElem * 8)
   Values     Values assigned to nodes used for coloring    (nElem * 8)
   pxyz       Coordinates of Nodes                          (8*nElem * 3)
   pind       indices of Nodes                                (nElem * 8)
   pvalue     Values arranged per Node                      (8*nElem * 1)

   See also PATCH_SHELL8.
```

patch_truss

```
PATCH_TRUSS  Patch information of the truss elements for plotting.

   [pxyz,pind,pvalue] = patch_truss(Nodes,NodeNum,Values) returns matrices
   to plot patches of truss elements.

   Nodes      Node definitions [NodID x y z].
   NodeNum    Node numbers [NodID1 NodID2 NodID3] (nElem * 2).
   Values     Values assigned to nodes used for coloring (nElem * 2).
   pxyz       Coordinates of Nodes  (2*nElem * 3).
   pind       Indices of Nodes  (nElem * 2).
   pvalue     Values arranged per Node (2*nElem * 1).
```

plotdisp

```
PLOTDISP   Plot the displacements.

            plotdisp(Nodes,Elements,Types,DOF,U,DLoads,Sections,Materials)
            plotdisp(Nodes,Elements,Types,DOF,U,[],Sections,Materials)
            plotdisp(Nodes,Elements,Types,DOF,U)
   DispScal=plotdisp(Nodes,Elements,Types,DOF,U,DLoads,Sections,Materials)
   plots the displacements. If DLoads, Sections and Materials are supplied, the
   displacements that occur due to the distributed loads if all nodes are fixed,
   are superimposed.

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }
   DOF        Degrees of freedom  (nDOF * 1)
   U          Displacements (nDOF * 1)
   DLoads     Distributed loads       [EltID n1globalX n1globalY n1globalZ ...]
              (use empty array [] when shear deformation (in beam element)
               is considered but no DLoads are present)
   Sections   Section definitions     [SecID SecProp1 SecProp2 ...]
   Materials  Material definitions    [MatID MatProp1 MatProp2 ... ]
   DispScal   Displacement scaling

   plotdisp(...,ParamName,ParamValue) sets the value of the specified
   parameters.  The following parameters can be specified:
   'DispScal'     Displacement scaling.  Default: 'auto'.
   'DispMax'      Mention maximal displacement.  Default: 'on'.
   'Undeformed'   Plots the undeformed mesh.  Default: 'k:'.
   'Handle'       Plots in the axis with this handle.  Default: current axis.
   Additional parameters are redirected to the PLOT3 function which plots
   the deformations.

   [DispScal,h] = plotdisp(...) returns a struct h with handles to all the
   objects in the plot.

   See also DISP_TRUSS, DISP_BEAM, PLOTELEM.
```

plotdispx

```
PLOTDISPX   Plot element quantity on deformed elements.

   plotdispx(Nodes,Elements,Types,DOF,U,x);
   [h,DispScal] = plotdispx(Nodes,Elements,Types,DOF,U,x,varargin)
    plots element quantity on deformed elements.

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }
   DOF        Degrees of freedom  (nDOF * 1)
   U          Displacements (nDOF * 1)
   x          Elements quantity to plot   (nElem * 1)
   plotdispx(...,ParamName,ParamValue) sets the value of the
   specified parameters.  The following parameters can be specified:
   'GCS'          Plot the GCS. Default: 'on'.
   'minmax'       Add location of min and max stress. Default: 'off'.
   'colorbar'     Add colorbar. Default: 'off'.
   'Undeformed'   Plots the undeformed mesh.  {'on' | 'off' (default)}
   'ncolor'       Number of colors in colormap. Default: 10.
   'Handle'       Plots in the axis with this handle.  Default: current axis.
   Additional parameters are redirected to the PATCH function which plots
   the elements.
```

plotelem

```
PLOTELEM   Plot the elements.

    plotelem(Nodes,Elements,Types)
   plots the elements.

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }

   plotelem(...,ParamName,ParamValue) sets the value of the specified
   parameters.  The following parameters can be specified:
   'Numbering'    Plots the element numbers.  Default: 'on'.
   'GCS'          Plots the global coordinate system.  Default: 'on'.
   'Handle'       Plots in the axis with this handle.  Default: current axis.
   Additional parameters are redirected to the PLOT3 function which plots
   the elements.

   h = PLOTELEM(...) returns a struct h with handles to all the objects
   in the plot.

   See also COORD_TRUSS, COORD_BEAM, PLOTDISP.
```

plotelemx

```
PLOTELEMX  Plot element quantity on elements.

   plotelemx(Nodes,Elements,Types,x) plots element quantity on elements.

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }
   x          Elements quantity to plot   (nElem * 1)
   plotelemx(...,ParamName,ParamValue) sets the value of the
   specified parameters.  The following parameters can be specified:
   'GCS'          Plot the GCS. Default: 'on'.
   'minmax'       Add location of min and max stress. Default: 'off'.
   'colorbar'     Add colorbar. Default: 'off'.
   'Undeformed'   Plots the undeformed mesh.  Default: 'k-'.
   'ncolor'       Number of colors in colormap. Default: 10.
   'Handle'       Plots in the axis with this handle.  Default: current axis.
   Additional parameters are redirected to the PATCH function which plots
   the elements.

   See also ELEMSTRESS.
```

plotforc

```
PLOTFORC   Plot element member forces.

   plotforc(ftype,Nodes,Elements,Types,Sections,Materials,Forces,DLoads)
   plotforc(ftype,Nodes,Elements,Types,Sections,Materials,Forces)
   plots the element member forces (in accordance to the beam convention).

   ftype      'norm'       Normal force (in the local x-direction)
   (for truss 'sheary'     Shear force in the local y-direction
    and beam  'shearz'     Shear force in the local z-direction
    elements) 'momx'       Torsional moment (around the local x-direction)
              'momy'       Bending moment around the local y-direction
              'momz'       Bending moment around the local z-direction
   ftype      'Nphi'       Normal force (per unit length) in meridional direction
   (for       'Qphi'       Transverse force (per unit length) in meridional direction
    shell2    'Mphi'       Bending moment (per unit length) in meridional direction
    elements) 'Ntheta'     Normal force (per unit length) in circumferential direction
              'Mtheta'     Bending moment (per unit length) in circumferential direction
   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }
   Sections   Section definitions       [SecID SecProp1 SecProp2 ...]
   Materials  Material definitions      [MatID MatProp1 MatProp2 ... ]
   Forces     Element forces in LCS (beam convention) [N Vy Vz T My Mz]
                                                               (nElem * 12)
   DLoads     Distributed loads       [EltID n1globalX n1globalY n1globalZ ...]

   plotforc(...,ParamName,ParamValue) sets the value of the specified
   parameters.  The following parameters can be specified:
   'ForcScal'     Force scaling.  Default: 'auto'.
   'MinMax'       Display minimum and maximum value.  Default: 'off'.
   'Values'       Force values.  Default: 'on'.
   'Undeformed'   Plots the undeformed mesh.  Default: 'k-'.
   'Handle'       Plots in the axis with this handle.  Default: current axis.
   Additional parameters are redirected to the PLOT3 function which plots
   the forces.

   [ForcScal,h] = plotforc(...) returns a struct h with handles to all the
   objects in the plot.

   See also FDIAGRGCS_BEAM, FDIAGRGCS_TRUSS.
```

plotgcs

```
lotgcs is a function.
   h = plotgcs(lref, h)
```

plotlcs

```
PLOTLCS   Plot the local element coordinate systems.

   [h,vLCS] = plotlcs(Nodes,Elements,Types)
   [h,vLCS] = plotlcs(Nodes,Elements,Types,[],varargin)
   plotlcs(Nodes,Elements,Types,vLCS,varargin)

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }
   vLCS       Element coordinate systems (nElem * 9)
   plotlcs(...,ParamName,ParamValue) sets the value of the specified
   parameters.  The following parameters can be specified:
   'GCS'          Plot the GCS. Default: 'on'.
   'Undeformed'   Plots the undeformed mesh.  Default: 'k-'.
   'Handle'       Plots in the axis with this handle.  Default: current
   axis.

   See also ELEMSTRESS
```

plotnodes

```
PLOTNODES   Plot the nodes.

    plotnodes(Nodes)
   plots the nodes.

   Nodes      Node definitions        [NodID x y z]

   plotnodes(...,ParamName,ParamValue) sets the value of the specified
   parameters.  The following parameters can be specified:
   'Numbering'    Plots the node numbers.  Default: 'on'.
   'GCS'          Plots the global coordinate system.  Default: 'on'.
   'Handle'       Plots in the axis with this handle.  Default: current axis.
   Additional parameters are redirected to the PLOT3 function which plots
   the nodes.

   h = PLOTNODES(...) returns a struct h with handles to all the objects
   in the plot.

   See also PLOTELEM, PLOTDISP.
```

plotprincstress

```
PLOTPRINCSTRESS   Plot the principal stresses in shell elements.

   plotprincstress(Nodes,Elements,Types,Spr,Vpr)
   plots the principal stresses in shell elements with a vector plot.

   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }
   Spr        Principal stresses (nElem * 72) [s3 s2 s1 0 0 0 ...]
   Vpr        Principal dir. matrix (output from principalstress)
              {nElem * 12}
   plotprincdir(...,ParamName,ParamValue) sets the value of the specified
   parameters.  The following parameters can be specified:
   'location'     Location (top,mid,bot). Default: 'top'.
   'GCS'          Plot the GCS. Default: 'on'.
   'VectScal'     Vector scaling.  Default: 'auto'.
   'Undeformed'   Plots the undeformed mesh.  Default: 'k-'.
   'Handle'       Plots in the axis with this handle.  Default: current
   axis.

   See also PRINCIPALSTRESS.
```

plotshellfcontour

```
PLOTSHELLFCONTOUR   Plot forces/moments per unit length in shell elements.

    plotshellfcontour(ftype,Nodes,Elements,Types,F)
    plots force contours (output from ELEMSHELLF/NODALSHELLF).

   ftype      'Nx'       Membrane forces
              'Ny'
              'Nxy'
              'Mx'       Bending moments
              'My'
              'Mxy'
              'Vx'       Shear forces
              'Vy'
   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }
   F          Forces/moments per unit length
              (nElem * 32)              [Nx Ny Nxy Mx My Mxy Vx Vy]
   plotshellfcontour(...,ParamName,ParamValue) sets the value of the specified
   parameters.  The following parameters can be specified:
   'Ncontour'     Number of contours. Default: '10'.
   'GCS'          Plot the GCS. Default: 'on'.
   'Undeformed'   Plots the undeformed mesh.  Default: 'k-'.
   'Handle'       Plots in the axis with this handle.  Default: current axis.

   See also ELEMSHELLF, SCONTOUR_SHELL8, SCONTOUR_SHELL4.
```

plotshellfcontourf

```
PLOTSHELLFCONTOURF   Plot filled contours of shell forces in shell elements.

    plotshellfcontourf(stype,Nodes,Elements,Types,F)
    plotshellfcontourf(stype,Nodes,Elements,Types,F,DOF,U)
    plots force contours (output from ELEMSTRESS).

   ftype      'Nx'       Membrane forces
              'Ny'
              'Nxy'
              'Mx'       Bending moments
              'My'
              'Mxy'
              'Vx'       Shear forces
              'Vy'
   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }
   F          Forces/moments per unit length
              (nElem * 32)             [Nx Ny Nxy Mx My Mxy Vx Vy]
   DOF        Degrees of freedom  (nDOF * 1)
   U          Displacements (nDOF * 1)
   plotshellfcontourf(...,ParamName,ParamValue) sets the value of the
   specified parameters.  The following parameters can be specified:
   'GCS'          Plot the GCS. Default: 'on'.
   'ncolor'       Number of colors in colormap. Default: 10.
   'Handle'       Plots in the axis with this handle.  Default: current axis.

   See also ELEMSHELLF, PATCH_SHELL8, PATCH_SHELL4.
```

plotstress

```
PLOTSTRESS   Plot the stresses.

   plotstress(stype,Nodes,Elements,Types,Sections,Materials,Forces,DLoads)
   plotstress(stype,Nodes,Elements,Types,Sections,Materials,Forces)
   plots the stresses.

   stype      'snorm'      Normal stress due to normal force
   (for truss 'smomyt'     Normal stress due to bending moment
    and beam               around the local y-direction at the top
    elements) 'smomyb'     Normal stress due to bending moment
                           around the local y-direction at the bottom
              'smomzt'     Normal stress due to bending moment
                           around the local z-direction at the top
              'smomzb'     Normal stress due to bending moment
                           around the local z-direction at the bottom
              'smax'       Maximal normal stress (normal force and bending moment)
              'smin'       Minimal normal stress (normal force and bending moment)
   stype      'sNphi'      Stress due to normal force in meridional direction
   (for       'sMphiT'     Stress at the top due to bending moment in
                                                               meridional direction
    shell2    'sMphiB'     Stress at the bottom due to bending moment in
                                                               meridional direction
    elements) 'sNtheta'    Stress due to normal force in circumferential direction
              'sMthetaT'   Stress at the top due to bending moment in
                                                          circumferential direction
              'sMthetaB'   Stress at the bottom due to bending moment in
                                                          circumferential direction
   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }
   Sections   Section definitions      [A ky kz Ixx Iyy Izz yt yb zt zb]
   Materials  Material definitions      [MatID MatProp1 MatProp2 ... ]
   Forces     Element forces in LCS (beam convention) [N Vy Vz T My Mz]
                                                                     (nElem * 12)
   DLoads     Distributed loads       [EltID n1globalX n1globalY n1globalZ ...]

   plotstress(...,ParamName,ParamValue) sets the value of the specified
   parameters.  The following parameters can be specified:
   'StressScal'   Stress scaling.  Default: 'auto'.
   'MinMax'       Display minimum and maximum value.  Default: 'off'.
   'Values'       Stress values.  Default: 'on'.
   'Undeformed'   Plots the undeformed mesh.  Default: 'k-'.
   'Handle'       Plots in the axis with this handle.  Default: current axis.
   Additional parameters are redirected to the PLOT3 function which plots
   the stresses.

   [StressScal,h] = plotstress(...) returns a struct h with handles to all the
   objects in the plot.

   See also SDIAGRGCS_BEAM, SDIAGRGCS_TRUSS.
```

plotstresscontour

```
PLOTSTRESSCONTOUR   Plot stress contour lines in shell elements.

    plotstresscontour(stype,Nodes,Elements,Types,S)
    plots stress contours (output from ELEMSTRESS/NODALSTRESS).

   stype      'sx'       Normal stress (in the global/local x-direction)
              'sy'
              'sz'
              'sxy'      Shear stress
              'syz'
              'sxz'
   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }
   S          Element stresses in GCS/LCS  [sxx syy szz sxy syz sxz]
                                                             (nElem * 72)
   plotstresscontour(...,ParamName,ParamValue) sets the value of the specified
   parameters.  The following parameters can be specified:
   'location'     Location (top,mid,bot). Default: 'top'.
   'Ncontour'     Number of contours. Default: '10'.
   'GCS'          Plot the GCS. Default: 'on'.
   'Undeformed'   Plots the undeformed mesh.  Default: 'k-'.
   'Handle'       Plots in the axis with this handle.  Default: current axis.

   See also ELEMSTRESS, SCONTOUR_SHELL8, SCONTOUR_SHELL4.
```

plotstresscontourf

```
PLOTSTRESSCONTOURF  Plot filled contours of stresses.

    plotstresscontourf(stype,Nodes,Elements,Types,S)
    plotstresscontourf(stype,Nodes,Elements,Types,S,DOF,U)
    plots stress contours (output from ELEMSTRESS).

   stype      'sx'       Normal stress (in the global/local x-direction)
              'sy'
              'sz'
              'sxy'      Shear stress
              'syz'
              'sxz'
   Nodes      Node definitions        [NodID x y z]
   Elements   Element definitions     [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }
   S          Element stresses in GCS/LCS  [sxx syy szz sxy syz sxz]
                                                        (nElem * 72)
   plotstresscontourf(...,ParamName,ParamValue) sets the value of the
   specified parameters.  The following parameters can be specified:
   'location'     Location (top,mid,bot). Default: 'top'.
   'GCS'          Plot the GCS. Default: 'on'.
   'minmax'       Add location of min and max stress. Default: 'on'.
   'colorbar'     Add colorbar. Default: 'on'.
   'ncolor'       Number of colors in colormap. Default: 10.
   'Handle'       Plots in the axis with this handle.  Default: current axis.

   See also ELEMSTRESS.
```

pressure_shell4

```
PRESSURE_SHELL4   Equivalent nodal forces for a shell4 element in the GCS
                  due to a pressure normal to the element surface.

   F = pressure_shell4(Pressure,Node)
   computes the equivalent nodal forces of a pressure load normal to the
   elements surface.

   Pressure   Distributed pressure in corner Nodes    [p1lobalZ; ...]
                                                             (4 * 1)
   Node       Node definitions           [x y z] (4 * 3)
   F          Load vector  (24 * 1)

   See also  ELEMPRESSURE, PRESSURE_SHELL8.
```

pressure_shell6

```
PRESSURE_SHELL8   Equivalent nodal forces for a shell6 element in the GCS
                  due to a pressure normal to the element surface.

   F = pressure_shell(Pressure,Node)
   computes the equivalent nodal forces of a pressure load normal to
   the elements surface.

   Pressure   Distributed loads in corner Nodes    [p1localZ; ...]
                                                                 (3 * 1)
   Node       Node definitions           [x y z] (6 * 3)
   F          Load vector  (36 * 1)

   See also  ELEMPRESSURE, PRESSURE_SHELL4.
```

pressure_shell8

```
PRESSURE_SHELL8   Equivalent nodal forces for a shell8 element in the GCS
                  due to a pressure normal to the element surface.

   F = pressure_shell(Pressure,Node)
   computes the equivalent nodal forces of a pressure load normal to
   the elements surface.

   Pressure   Distributed loads in corner Nodes    [p1localZ; ...]
                                                                 (4 * 1)
   Node       Node definitions           [x y z] (8 * 3)
   F          Load vector  (48 * 1)

   See also  ELEMPRESSURE, PRESSURE_SHELL4.
```

principalstress

```
PRINCIPALSTRESS    Compute the principal stresses and directions in shell elements.

   [Spr,Vpr] = principalstress(Elements,SeGCS)
   computes the principal stresses and directions.

   Elements   Element definitions   [EltID TypID SecID MatID n1 n2...]
   SeGCS      Element stresses in GCS in corner nodes IJKL and
              at top/mid/bot of shell (nElem * 72)
              72 = 6 stresscomp. * 4 nodes * 3 locations
                                          [sxx syy szz sxy syz sxz ...]
   Spr        Principal stress (nElem * 72)
                                          [s1 s2 s3 0 0 0 ...]
   Vpr        Principal stress directions {nElem * 12}

   See also ELEMSTRESS, PLOTPRINCDIR.
```

printdisp

```
PRINTDISP   Display the displacements in the command window.

    printdisp(Nodes,DOF,U)
   displays the displacements in the command window.

   Nodes      Node definitions          [NodID x y z]
   DOF        Degrees of freedom  (nDOF * 1)
   U          Displacements (nDOF * 1)

   See also PRINTFORC, PLOTDISP.
```

printforc

```
PRINTFORC   Display the forces in the command window.

    printforc(Elements,Forces)
   displays the forces in the command window.

   Elements   Element definitions       [EltID TypID SecID MatID n1 n2 ...]
   Forces     Element forces          [N Vy Vz T My Mz] (nElem * 12)

   See also PRINTDISP.
```

printshellf

```
PRINTSHELLF Display forces/moments in command window (shell elements).

   printshellf(Elements,F)
   displays shell forces in command window.

   Elements   Element definitions       [EltID TypID SecID MatID n1 n2 ...]
   F          Shellf matrix          [Nx Ny Nxy Mx My Mxy Vx Vy] (nElem * 32)

   See also PRINTSTRESS.
```

printstress

```
PRINTSTRESS Display stress in command window (shell elements).

   printstress(Elements,S,location) displays stress in command window.

   Elements   Element definitions       [EltID TypID SecID MatID n1 n2 ...]
   S          Stress matrix          [sx sy sz sxy syz szx] (nElem * 72)

   See also PRINTSHELLF.
```

q_dkt

```
Q_DKT   Q matrix for a DKT element.

   Q = Q_DKT(b,c,det) returns the Q matrix of the DKT element,
   which is used to compute the curvatures of the plate element

   b       Geometrical property of the triangle (see ke_dkt) (3 * 1)
   c       Geometrical property of the triangle              (3 * 1)
   det     Determinant of the parametric transformation (=2*area of
           triangle)

   See also KE_DKT, KELCS_SHELL4, SE_SHELL4.
```

reaction

```
REACTION   Compute the reaction forces for a degree of freedom.

   Freac=reaction(Elements,ForcesGCS,seldof)
   computes the reaction forces for a degree of freedom.

   Elements   Element definitions       [EltID TypID SecID MatID n1 n2 ...]
   ForcesGCS  Element forces in GCS (nElem * 12)
   seldof     Selected DOF labels (kDOF * 1)
   Freac      Reaction force

   See also ELEMFORCES.
```

removedof

```
REMOVEDOF    Remove DOF with Dirichlet boundary conditions equal to zero.

   DOF=removedof(DOF,seldof) removes the specified Dirichlet boundary
   conditions from the degrees of freedom vector.

   DOF        Degrees of freedom  (nDOF * 1)
   seldof  Dirichlet boundary conditions equal to zero      [NodID.dof]

   See also GETDOF.
```

reprow

```
REPROW   Replicate rows from a matrix.

    Matrix=reprow(Matrix,RowSel,nTime,RowInc)
   replicates the selected rows from a matrix a number of times and adds them
   below the existing rows. The k-th time the increment RowInc is added k times
   to the copied rows.

   Matrix     Matrix (nRow * nCol)
   RowSel     Rows to be copied (1 * kRow)
   nTime      Number of times
   RowInc     Increments that are added to the different columns of the copied
              rows (1 * nCol)
```

scontour_shell4

```
SCONTOUR_SHELL4   Matrix to plot contours in a shell4 element.

   Scontour = scontour_shell4(Node,Stress,Svalues) returns a matrix used
   for plotting contours in a shell4 element.

   Node       Node definitions           [x y z] (4 * 3)
   Stress     Stress in nodes                    (4 * 1)
   Svalues    Values of contours          (nContour * 1)
   Scontour   Coordinates of contours in GCS

   See also PLOTSTRESSCONTOUR, PLOTSHELLFCONTOUR.
```

scontour_shell8

```
SCONTOUR_SHELL8   Matrix to plot contours in a shell8 element.

   Scontour = scontour_shell8(Node,Stress,Svalues) returns a matrix used
   for plotting contours in a shell4 element.

   Node       Node definitions           [x y z] (4 * 3)
   Stress     Stress in nodes         (4 * 1) or (8 * 1)
   Svalues    Values of contours          (nContour * 1)
   Scontour   Coordinates of contours in GCS

   See also PLOTSTRESSCONTOUR, PLOTSHELLFCONTOUR.
```

sdiagrgcs_beam

```
SDIAGRGCS_BEAM   Return matrices to plot the stresses in a beam element.

   [ElemGCS,SdiagrGCS,ElemExtGCS,ExtremaGCS,Extrema]
                              = sdiagrgcs_beam(stype,Forces,Node,Section,[],DLoad,Points)
   [ElemGCS,SdiagrGCS,ElemExtGCS,ExtremaGCS,Extrema]
                              = sdiagrgcs_beam(Forces,Node,Section,[],DLoad)
   returns the coordinates of the points along the beam in the global
   coordinate system and the coordinates of the stresses with respect to the beam
   in the global coordinate system. These can be added in order to plot the
   stresses: ElemGCS+SdiagrGCS. The coordinates of the points with extreme values
   and the coordinates of the extreme values with respect to the beam are given
   as well and can be similarly added:  ElemExtGCS+ExtremaGCS. Extrema is the
   list with the corresponding extreme values.

   stype      'snorm'      Normal stress due to normal force
              'smomyt'     Normal stress due to bending moment
                           around the local y-direction at the top
              'smomyb'     Normal stress due to bending moment
                           around the local y-direction at the bottom
              'smomzt'     Normal stress due to bending moment
                           around the local z-direction at the top
              'smomzb'     Normal stress due to bending moment
                           around the local z-direction at the bottom
              'smax'       Maximal normal stress (normal force and bending moment)
              'smin'       Minimal normal stress (normal force and bending moment)
   Forces     Element forces in LCS (beam convention) [N; Vy; Vz; T; My; Mz]
                                                                          (12 * 1)
   Node       Node definitions        [x y z] (3 * 3)
   Section    Section definition      [A ky kz Ixx Iyy Izz yt yb zt zb]
   DLoad      Distributed loads       [n1globalX; n1globalY; n1globalZ; ...]
                                                                           (6 * 1)
   Points     Points in the local coordinate system (1 * nPoints)
   ElemGCS    Coordinates of the points along the beam in GCS (nPoints * 3)
   SdiagrGCS  Coordinates of the stress with respect to the beam in GCS
                                                                     (nValues * 3)
   ElemExtGCS Coordinates of the points with extreme values in GCS (nValues * 3)
   ExtremaGCS Coordinates of the extreme values with respect to the beam in GCS
                                                                     (nValues * 3)
   Extrema    Extreme values (nValues * 1)

   See also PLOTSTRESS, SDIAGRLCS_BEAM, SDIAGRGCS_TRUSS.
```

sdiagrgcs_shell2

```
SDIAGRGCS_SHELL2   Return matrices to plot the stresses in a SHELL2 element.

   [ElemGCS,SdiagrGCS,ElemExtGCS,ExtremaGCS,Extrema]
                 = sdiagrgcs_shell2(ftype,Forces,Node,Section,Material,DLoad,Points)
   [ElemGCS,SdiagrGCS,ElemExtGCS,ExtremaGCS,Extrema]
                 = sdiagrgcs_shell2(ftype,Forces,Node,Section,Material,DLoad)
   returns the coordinates of the points along the SHELL2 in the global
   coordinate system and the coordinates of the stresses with respect to the element
   in the global coordinate system. These can be added in order to plot the
   stresses: ElemGCS+SdiagrGCS. The coordinates of the points with extreme values
   and the coordinates of the extreme values with respect to the element are given
   as well and can be similarly added:  ElemExtGCS+ExtremaGCS. Extrema is the
   list with the correspondig extreme values.

   ftype      'sNphi'      Stress due to normal force in meridional direction
              'sMphiT'     Stress at the top due to bending moment in
                                                               meridional direction
              'sMphiB'     Stress at the bottom due to bending moment in
                                                               meridional direction
              'sNtheta'    Stress due to normal force in circumferential direction
              'sMthetaT'   Stress at the top due to bending moment in
                                                           circumferential direction
              'sMthetaB'   Stress at the bottom due to bending moment in
                                                           circumferential direction
   Forces     Element forces in LCS (beam convention) [N; Vy; 0; 0; 0; Mz] (12 * 1)
   Node       Node definitions        [x y z] (3 * 3)
   DLoad      Distributed loads       [n1globalX; n1globalY; n1globalZ; ...] (6 * 1)
   Points     Points in the local coordinate system (1 * nPoints)
   ElemGCS    Coordinates of the points along the element in GCS (nPoints * 3)
   FdiagrGCS  Coordinates of the force with respect to the element in GCS (nValues * 3)
   ElemExtGCS Coordinates of the points with extreme values in GCS (nValues * 3)
   ExtremaGCS Coordinates of the extreme values with respect to the element in GCS
                                                                     (nValues * 3)
   Extrema    Extreme values (nValues * 1)
```

sdiagrgcs_truss

```
SDIAGRGCS_TRUSS   Return matrices to plot the stresses in a truss element.

   [ElemGCS,SdiagrGCS,ElemExtGCS,ExtremaGCS,Extrema]
                           = sdiagrgcs_truss(stype,Forces,Node,Section,[],[],Points)
   [ElemGCS,SdiagrGCS,ElemExtGCS,ExtremaGCS,Extrema]
                           = sdiagrgcs_truss(stype,Forces,Node,Section)
   returns the coordinates of the points along the truss in the global
   coordinate system and the coordinates of the stresses with respect to the truss
   in the global coordinate system. These can be added in order to plot the
   stresses: ElemGCS+SdiagrGCS. The coordinates of the points with extreme values
   and the coordinates of the extreme values with respect to the truss are given
   as well and can be similarly added:  ElemExtGCS+ExtremaGCS. Extrema is the
   list with the corresponding extreme values.

   stype      'snorm'      Normal stress due to normal force
   Forces     Element forces in LCS [N; 0; 0; 0; 0; 0](12 * 1)
   Node       Node definitions        [x y z] (3 * 3)
   Section    Section definition      [A ky kz Ixx Iyy Izz yt yb zt zb]
   Points     Points in the local coordinate system (1 * nPoints)
   ElemGCS    Coordinates of the points along the truss in GCS (nPoints * 3)
   SdiagrGCS  Coordinates of the force with respect to the truss in GCS
                                                               (nValues * 3)
   ElemExtGCS Coordinates of the points with extreme values in GCS (nValues * 3)
   ExtremaGCS Coordinates of the extreme values with respect to the truss in GCS
                                                               (nValues * 3)
   Extrema    Extreme values (nValues * 1)

   See also PLOTSTRESS, SDIAGRGCS_BEAM.
```

sdiagrlcs_beam

```
SDIAGRLCS_BEAM   Stress diagram for a beam element in LCS.

   [SdiagrLCS,loc,Extrema] = sdiagrlcs_beam(ftype,Forces,DLoadLCS,L,Points)
   computes the stresses at the specified points. The extreme values are
   analytically determined.

   stype      'snorm'      Normal stress due to normal force
              'smomyt'     Normal stress due to bending moment
                           around the local y-direction at the top
              'smomyb'     Normal stress due to bending moment
                           around the local y-direction at the bottom
              'smomzt'     Normal stress due to bending moment
                           around the local z-direction at the top
              'smomzb'     Normal stress due to bending moment
                           around the local z-direction at the bottom
              'smax'       Maximal normal stress (normal force and bending moment)
              'smin'       Minimal normal stress (normal force and bending moment)
   Forces     Element forces in LCS (beam convention) [N; Vy; Vz; T; My; Mz](12 * 1)
   DLoadLCS   Distributed loads in LCS [n1localX; n1localY; n1localZ; ...](6 * 1)
   Points     Points in the local coordinate system (1 * nPoints)
   Section    Section definition         [A ky kz Ixx Iyy Izz yt yb zt zb]
   SdiagrLCS  Stresses at the points (1 * nPoints)
   loc        Locations of the extreme values (nValues * 1)
   Extrema    Extreme values (nValues * 1)

   See also SDIAGRGCS_BEAM.
```

selcs_plane3

```
SELCS_PLANE3   Compute the element stresses for a plane3 element.

   [SeLCS] = selcs_plane3(Node,Section,Material,UeGCS,Options)
   computes the element stresses in the
   local coordinate system for the plane3 element.

   Node       Node definitions    [x y z] (4 * 3)
   Section    Section definition  [h]  (only used in plane stress)
   Material   Material definition [E nu rho]
   UeLCS      Displacements (6 * nSteps)
   Options    Element options            {Option1 Option2 ...}
   SeLCS      Element stresses in LCS in corner nodes IJKL
              (9 * nTimeSteps) [sxx syy sxy]

   See also ELEMSTRESS, SE_PLANE3.
```

selcs_plane4

```
SELCS_PLANE4   Compute the element stresses for a plane4 element.

   [SeLCS] = selcs_plane4(Node,Section,Material,UeGCS,Options)
   computes the element stresses in the
   local coordinate system for the plane4 element.

   Node       Node definitions    [x y z] (4 * 3)
   Section    Section definition  [h]  (only used in plane stress)
   Material   Material definition [E nu rho]
   UeLCS      Displacements (8 * nSteps)
   Options    Element options            {Option1 Option2 ...}
   SeLCS      Element stresses in LCS in corner nodes IJKL
              (12 * nTimeSteps) [sxx syy sxy] for plane stress / plane strain problems
              (16 * nTimeSteps) [sxx stheta syy sxy] for axisymmetric problems

   See also ELEMSTRESS, SE_PLANE4.
```

selcs_plane6

```
SELCS_PLANE6   Compute the element stresses for a plane6 element.

   [SeLCS] = selcs_plane6(Node,Section,Material,UeGCS,Options)
   computes the element stresses in the
   local coordinate system for the plane6 element.

   Node       Node definitions           [x y z] (6 * 3)
              Nodes should have the following order:
              3
              | \
              6  5
              |    \
              1--4--2
   Section    Section definition  [h]  (only used in plane stress)
   Material   Material definition [E nu rho]
   UeLCS      Displacements (6 * nSteps)
   Options    Element options           {Option1 Option2 ...}
   SeLCS      Element stresses in LCS in corner nodes IJKL
              (9 * nTimeSteps) [sxx syy sxy]

   See also ELEMSTRESS, SE_PLANE6.
```

selcs_plane8

```
SELCS_PLANE8   Compute the element stresses for a plane4 element.

   [SeLCS] = selcs_plane8(Node,Section,Material,UeGCS,Options)
   computes the element stresses in the
   local coordinate system for the plane8 element.

   Node       Node definitions    [x y z] (4 * 3)
   Section    Section definition  [h]  (only used in plane stress)
   Material   Material definition [E nu rho]
   UeLCS      Displacements (8 * nSteps)
   Options    Element options            {Option1 Option2 ...}
   SeLCS      Element stresses in LCS in corner nodes IJKL
              (12 * nTimeSteps) [sxx syy sxy]

   See also ELEMSTRESS, SE_PLANE8.
```

selcs_shell4

```
SELCS_SHELL4   Compute the element stresses for a shell4 element.

   [SeLCS] = selcs_shell4(Node,Section,Material,UeGCS,Options)
   computes the element stresses in the global and the
   local coordinate system for the shell4 element.

   Node       Node definitions           [x y z] (4 * 3)
   h          Shell thickness
   E          Young's modulus
   nu         Poisson coefficient
   rho        Mass density
   UeLCS      Displacements (24 * nTimeSteps)
   Options    Element options           {Option1 Option2 ...}
   SeLCS      Element stresses in LCS in corner nodes IJKL and
              at top/mid/bot of shell (72 * nTimeSteps)
                                       [sxx syy szz sxy syz sxz]

   See also ELEMSTRESS, SE_SHELL4.
```

selectdof

```
SELECTDOF   Select degrees of freedom.

       L=selectdof(DOF,seldof)
   [L,I]=selectdof(DOF,seldof)
       L=selectdof(DOF,seldof,'Ordering',ordering)
   creates the matrix to extract degrees of freedom from the global degrees of
   freedom by matrix multiplication.

   DOF         Degrees of freedom  (nDOF * 1)
   seldof      Selected DOF labels (kDOF * 1)
   L           Selection matrix (kDOF * nDOF)
   I           Index vector (kDOF * 1)
   ordering    'seldof','DOF' or 'sorted'                      Default: 'seldof'
               Ordering of L and I similar as seldof, DOF or sorted
```

selectnode

```
SELECTNODE   Select nodes by location.

   Nodesel=selectnode(Nodes,x,y,z)
   Nodesel=selectnode(Nodes,xmin,ymin,zmin,xmax,ymax,zmax)
   selects nodes by location.

   Nodes      Node definitions          [NodID x y z]
   Nodesel    Node definitions of the selected nodes
```

se_plane3

```
SE_PLANE3   Compute the element stresses for a plane3 element.

   [SeGCS,SeLCS,vLCS] = se_plane3(Node,Section,Material,UeGCS,Options,GCS)
   [SeGCS,SeLCS]      = se_plane3(Node,Section,Material,UeGCS,Options,GCS)
    SeGCS             = se_plane3(Node,Section,Material,UeGCS,Options,GCS)
   computes the element stresses in the global and the
   local coordinate system for the shell3 element.

   Node       Node definitions           [x y z] (3 * 3)
              Nodes should have the following order:
              3
              | \
              1--2
   Section    Section definition         [h]  (only used in plane stress)
   Material   Material definition        [E nu rho]
   UeGCS      Displacements (8 * nTimeSteps)
   Options    Element options            {Option1 Option2 ...}
   GCS        Global coordinate system in which stresses are returned
              'cart'|'cyl'
   SeGCS      Element stresses in GCS in corner nodes IJKL
              18 = 6 stress comp. * 3 nodes (18 * nTimeSteps)
                                        [sxx syy szz sxy syz sxz]
   SeLCS      Element stresses in LCS in corner nodes IJKL
              18 = 6 stress comp. * 3 nodes (18 * nTimeSteps)
                                        [sxx syy szz sxy syz sxz]
   vLCS       Unit vectors of LCS (1 * 9)

   See also ELEMSTRESS, SE_SHELL8.
```

se_plane4

```
SE_PLANE4   Compute the element stresses for a plane4 element.

   [SeGCS,SeLCS,vLCS] = se_plane4(Node,Section,Material,UeGCS,Options,GCS)
   [SeGCS,SeLCS]      = se_plane4(Node,Section,Material,UeGCS,Options,GCS)
    SeGCS             = se_plane4(Node,Section,Material,UeGCS,Options,GCS)
   computes the element stresses in the global and the
   local coordinate system for the shell4 element.

   Node       Node definitions           [x y z] (4 * 3)
              Nodes should have the following order:
                4---3
                |   |
                1---2
   Section    Section definition         [h]  (only used in plane stress)
   Material   Material definition        [E nu rho]
   UeGCS      Displacements (8 * nTimeSteps)
   Options    Element options            {Option1 Option2 ...}
   GCS        Global coordinate system in which stresses are returned
              'cart'|'cyl'
   SeGCS      Element stresses in GCS in corner nodes IJKL
              24 = 6 stress comp. * 4 nodes (24 * nTimeSteps)
                                        [sxx syy szz sxy syz sxz]
   SeLCS      Element stresses in LCS in corner nodes IJKL
              24 = 6 stress comp. * 4 nodes (24 * nTimeSteps)
                                        [sxx syy szz sxy syz sxz]
   vLCS       Unit vectors of LCS (1 * 9)

   See also ELEMSTRESS, SE_SHELL8.
```

se_plane6

```
SE_PLANE6   Compute the element stresses for a plane6 element.

   [SeGCS,SeLCS,vLCS] = se_plane6(Node,Section,Material,UeGCS,Options,GCS)
   [SeGCS,SeLCS]      = se_plane6(Node,Section,Material,UeGCS,Options,GCS)
    SeGCS             = se_plane6(Node,Section,Material,UeGCS,Options,GCS)
   computes the element stresses in the global and the
   local coordinate system for the plane6 element.

   Node       Node definitions           [x y z] (8 * 3)
              Nodes should have the following order:
              3
              | \
              6  5
              |    \
              1--4--2

   Section    Section definition         [h]  (only used in plane stress)
   Material   Material definition        [E nu rho]
   UeGCS      Displacements (8 * nTimeSteps)
   Options    Element options           {Option1 Option2 ...}
   GCS        Global coordinate system in which stresses are returned
              'cart'|'cyl'
   SeGCS      Element stresses in GCS in corner nodes IJKL
              48 = 6 stress comp. * 8 nodes (24 * nTimeSteps)
                                        [sxx syy szz sxy syz sxz]
   SeLCS      Element stresses in LCS in corner nodes IJKL
              48 = 6 stress comp. * 8 nodes (24 * nTimeSteps)
                                        [sxx syy szz sxy syz sxz]
   vLCS       Unit vectors of LCS (1 * 9)

   See also ELEMSTRESS, SE_PLANE8.
```

se_plane8

```
SE_PLANE8   Compute the element stresses for a plane8 element.

   [SeGCS,SeLCS,vLCS] = se_plane8(Node,Section,Material,UeGCS,Options,GCS)
   [SeGCS,SeLCS]      = se_plane8(Node,Section,Material,UeGCS,Options,GCS)
    SeGCS             = se_plane8(Node,Section,Material,UeGCS,Options,GCS)
   computes the element stresses in the global and the
   local coordinate system for the shell8 element.

   Node       Node definitions           [x y z] (8 * 3)
              Nodes should have the following order:
              4----7----3
              |         |
              8         6
              |         |
              1----5----2
   Section    Section definition         [h]  (only used in plane stress)
   Material   Material definition        [E nu rho]
   UeGCS      Displacements (8 * nTimeSteps)
   Options    Element options           {Option1 Option2 ...}
   GCS        Global coordinate system in which stresses are returned
              'cart'|'cyl'
   SeGCS      Element stresses in GCS in corner nodes IJKL
              48 = 6 stress comp. * 8 nodes (24 * nTimeSteps)
                                         [sxx syy szz sxy syz sxz]
   SeLCS      Element stresses in LCS in corner nodes IJKL
              48 = 6 stress comp. * 8 nodes (24 * nTimeSteps)
                                         [sxx syy szz sxy syz sxz]
   vLCS       Unit vectors of LCS (1 * 9)

   See also ELEMSTRESS, SE_SHELL8.
```

se_shell4

```
SE_SHELL4   Compute the element stresses for a shell4 element.

   [SeGCS,SeLCS,vLCS] = se_shell4(Node,Section,Material,UeGCS,Options,GCS)
   [SeGCS,SeLCS]      = se_shell4(Node,Section,Material,UeGCS,Options,GCS)
    SeGCS             = se_shell4(Node,Section,Material,UeGCS,Options,GCS)
   computes the element stresses in the global and the
   local coordinate system for the shell4 element.

   Node       Node definitions           [x y z] (4 * 3)
              Nodes should have the following order:
   Section    Section definition         [h]
   Material   Material definition        [E nu rho]
   UeGCS      Displacements (24 * nTimeSteps)
   Options    Element options           {Option1 Option2 ...}
   GCS        Global coordinate system in which stresses are returned
              'cart'|'cyl'|'sph'
   SeGCS      Element stresses in GCS in corner nodes IJKL and
              at top/mid/bot of shell (72 * nTimeSteps)
              72 = 6 stress comp. * 4 nodes * 3 locations
                                        [sxx syy szz sxy syz sxz]
   SeLCS      Element stresses in LCS in corner nodes IJKL and
              at top/mid/bot of shell (72 * nTimeSteps)
                                        [sxx syy szz sxy syz sxz]
   vLCS       Unit vectors of LCS (1 * 9)

   See also ELEMSTRESS, SE_SHELL8.
```

se_shell6

```
SE_SHELL6   Compute the element stresses for a shell6 element.

   [SeGCS,SeLCS,vLCS] = se_shell6(Node,Section,Material,UeGCS,Options,gcs)
   [SeGCS,SeLCS]      = se_shell6(Node,Section,Material,UeGCS,Options,gcs)
    SeGCS             = se_shell6(Node,Section,Material,UeGCS,Options,gcs)
   computes the element stresses in the global and the
   local coordinate system for the shell6 element.

   Node       Node definitions           [x y z] (6 * 3)
              Nodes should have the following order:
              3
              | \
              6  5
              |    \
              1--4--2

   Section    Section definition         [h] or [h1 h2 h3]
              (uniform thickness or defined in corner nodes(1,2,3))
   Material   Material definition        [E nu rho] or [Exx Eyy nuxy muxy muyz muzx theta rho]
   UeGCS      Displacements (36 * nTimeSteps)
   Options    Element options struct. Fields:
              -LCSType: determine the reference local element
                        coordinate system. Values:
                        'element' (default) or 'global'
              -MatType: 'isotropic' (default) or 'orthotropic'
              -Offset: nodal offset from shell midplane. Values:
                      'top', 'mid' (default), 'bot' or numerical value
   GCS        Global coordinate system in which stresses are returned
              'cart'|'cyl'|'sph'
   SeGCS      Element stresses in GCS in corner nodes IJKL and
              at top/mid/bot of shell (72 * nTimeSteps)
              72 = 6 stress comp. * 4 nodes * 3 locations
              (in a triangular element the fourth node is zero)
                                        [sxx syy szz sxy syz sxz]
   SeLCS      Element stresses in LCS in corner nodes IJKL and
              at top/mid/bot of shell (54 * nTimeSteps)
                                        [sxx syy szz sxy syz sxz]
   vLCS       Unit vectors of LCS (3 * 3)

   See also ELEMSTRESS, SE_SHELL4.
```

se_shell8

```
SE_SHELL8   Compute the element stresses for a shell8 element.

   [SeGCS,SeLCS,vLCS] = se_shell8(Node,Section,Material,UeGCS,Options,gcs)
   [SeGCS,SeLCS]      = se_shell8(Node,Section,Material,UeGCS,Options,gcs)
    SeGCS             = se_shell8(Node,Section,Material,UeGCS,Options,gcs)
   computes the element stresses in the global and the
   local coordinate system for the shell8 element.

   Node       Node definitions           [x y z] (8 * 3)
              Nodes should have the following order:
              4----7----3
              |         |
              8         6
              |         |
              1----5----2

   Section    Section definition         [h] or [h1 h2 h3 h4]
              (uniform thickness or defined in corner nodes(1,2,3,4))
   Material   Material definition        [E nu rho] or [Exx Eyy nuxy muxy muyz muzx theta rho]
   UeGCS      Displacements (48 * nTimeSteps)
   Options    Element options struct. Fields:
              -LCSType: determine the reference local element
                        coordinate system. Values:
                        'element' (default) or 'global'
              -MatType: 'isotropic' (default) or 'orthotropic'
              -Offset: nodal offset from shell midplane. Values:
                      'top', 'mid' (default), 'bot' or numerical value
   GCS        Global coordinate system in which stresses are returned
              'cart'|'cyl'|'sph'
   SeGCS      Element stresses in GCS in corner nodes IJKL and
              at top/mid/bot of shell (72 * nTimeSteps)
              72 = 6 stress comp. * 4 nodes * 3 locations
                                          [sxx syy szz sxy syz sxz]
   SeLCS      Element stresses in LCS in corner nodes IJKL and
              at top/mid/bot of shell (72 * nTimeSteps)
                                          [sxx syy szz sxy syz sxz]
   vLCS       Unit vectors of LCS (3 * 3)

   See also ELEMSTRESS, SE_SHELL4.
```

se_solid20

```
SE_SOLID20   Compute the element stresses for a solid20 element.

   [SeGCS,SeLCS,vLCS] = se_solid20(Node,Section,Material,UeGCS,Options,gcs)
   [SeGCS,SeLCS]      = se_solid20(Node,Section,Material,UeGCS,Options,gcs)
    SeGCS             = se_solid20(Node,Section,Material,UeGCS,Options,gcs)
   computes the element stresses in the global and the
   local coordinate system for the solid20 element.

   Node       Node definitions           [x y z] (20 * 3)
              Nodes should have the following order:
                   8---15----7
                  /|        /|
                16 |      14 |
                / 20      / 19
               /   |     /   |
              5---13----6    |
              |    4--11|----3
              |   /     |   /
             17 12     18  10
              | /       | /
              |/        |/
              1----9----2
```

se_solid8

```
SE_SOLID8   Compute the element stresses for a solid8 element.

   [SeGCS,SeLCS,vLCS] = se_solid8(Node,Section,Material,UeGCS,Options,gcs)
   [SeGCS,SeLCS]      = se_solid8(Node,Section,Material,UeGCS,Options,gcs)
    SeGCS             = se_solid8(Node,Section,Material,UeGCS,Options,gcs)
   computes the element stresses in the global and the
   local coordinate system for the solid8 element.

   Node       Node definitions           [x y z] (8 * 3)
              Nodes should have the following order:
                  8---------7
                 /|        /|
                / |       / |
               /  |      /  |
              5---------6   |
              |   4-----|---3
              |  /      |  /
              | /       | /
              |/        |/
              1---------2

   Section    Section definition         []
   Material   Material definition        [E nu rho]
   UeGCS      Displacements (48 * nTimeSteps)
   Options    Element options struct. Fields: []
   GCS        Global coordinate system in which stresses are returned
              'cart'|'cyl'|'sph'
   SeGCS      Element stresses in GCS in nodes (48 * nTimeSteps)
              48 = 6 stress comp. * 8 nodes
                                        [sxx syy szz sxy syz sxz]
   SeGCS      Element stresses in GCS in nodes (48 * nTimeSteps)
              48 = 6 stress comp. * 8 nodes
                                        [sxx syy szz sxy syz sxz]
   vLCS       Unit vectors of LCS (3 * 3)

   See also ELEMSTRESS, SE_SHELL4.
```

se_truss

```
e_truss is a function.
   [SeGCS, SeLCS, vLCS] = se_truss(Node, Section, Material, UeGCS, Options, gcs)
```

sh_qs4

```
SH_QS4     Shape functions for a quadrilateral serendipity element with 4
           nodes.

   [Ni,dN_dxi,dN_deta] = sh_qs4(xi,eta) returns the shape functions and
   its derivatives to the natural coordinates in the point (xi,eta).

   xi      Natural coordinate                  (scalar)
   eta     Natural coordinate                  (scalar)
   Ni      Shape functions in point (xi,eta)   (4 * 1)
   dN_dxi  Derivative of Ni to xi              (4 * 1)
   dN_deta Derivative of Ni to eta             (4 * 1)

   See also KELCS_SHELL4.
```

sh_qs8

```
SH_QS8     Shape functions for an 8 node quadrilateral serendipity element.

   [Ni,dN_dxi,dN_deta] = sh_qs8(xi,eta) returns the shape functions and
   its derivatives with respect to the natural coordinates in the point (xi,eta).

   xi      natural coordinate                 (scalar)
   eta     natural coordinate                 (scalar)
   Ni      shape functions in point (xi,eta)   (8 * 1)
   dN_dxi  derivative of Ni to xi             (8 * 1)
   dN_deta derivative of Ni to eta            (8 * 1)

   see also KE_SHELL8.
```

sh_t

```
SH_T     Shape functions for a triangular plate element.

   [Ni] = sh_t(L,b,c) returns the shape functions and
   its derivatives in point L.

   L       Area coordinates [L1,L2,L3]                       (3 * 1)
   b       Geometrical property of the triangle (see ke_dkt) (3 * 1)
   c       Geometrical property of the triangle              (3 * 1)
   Ni      Shape functions and derivatives                   (9 * 1)
           in point L

   These shape functions are used to determine the mass matrix of a
   triangular plate element.

   See also KE_DKT.
```

sh_t10

```
SH_T10    Shape functions for a 10 node triangular element.

   [Ni,dN_dxi,dN_deta] = sh_t10(xi,eta) returns the shape functions and
   its derivatives with respect to the natural coordinates in the point (xi,eta)

   xi       natural coordinate                 (scalar)
   eta      natural coordinate                 (scalar)
   Ni       shape functions in point (xi,eta)   (10 * 1)
   dN_dxi   derivative of Ni to xi              (10 * 1)
   dN_deta  derivative of Ni to eta             (10 * 1)

   The nodes have the following order:

   see also SH_T3, SH_T6, SH_T15.
```

sh_t15

```
SH_T15    Shape functions for a 15 node triangular element.

   [Ni,dN_dxi,dN_deta] = sh_t15(xi,eta) returns the shape functions and
   its derivatives with respect to the natural coordinates in the point (xi,eta)*

   xi      natural coordinate                  (scalar)
   eta     natural coordinate                  (scalar)
   Ni      shape functions in point (xi,eta)   (15 * 1)
   dN_dxi  derivative of Ni to xi              (15 * 1)
   dN_deta derivative of Ni to eta             (15 * 1)

   The nodes have the following order:

   see also SH_T3, SH_T6, SH_T15.
```

sh_t3

```
SH_T3     Shape functions for a 3 node triangular element.

   [Ni,dN_dxi,dN_deta] = sh_t3(xi,eta) returns the shape functions and
   its derivatives with respect to the natural coordinates in the point (xi,eta).

   xi      natural coordinate                 (scalar)
   eta     natural coordinate                 (scalar)
   Ni      shape functions in point (xi,eta)   (3 * 1)
   dN_dxi  derivative of Ni to xi             (3 * 1)
   dN_deta derivative of Ni to eta            (3 * 1)
```

sh_t6

```
SH_T6     Shape functions for an 6 node triangular element.

   [Ni,dN_dxi,dN_deta] = sh_t6(xi,eta) returns the shape functions and
   its derivatives with respect to the natural coordinates in the point (xi,eta).

   xi      natural coordinate                 (scalar)
   eta     natural coordinate                 (scalar)
   Ni      shape functions in point (xi,eta)   (6 * 1)
   dN_dxi  derivative of Ni to xi             (6 * 1)
   dN_deta derivative of Ni to eta            (6 * 1)

   see also KE_SHELL6.
```

sh_vs20

```
SH_VS20    Shape functions for a volume serendipity element with 20 nodes.

   [Ni,dNi_dxi,dNi_deta,dNi_dzeta] = sh_vs20(xi,eta,zeta)
   returns the shape functions and its derivatives to the natural coordinates
   in the point (xi,eta,zeta).

   xi       Natural coordinate                  (scalar)
   eta      Natural coordinate                  (scalar)
   zeta     Natural coordinate                  (scalar)
   Ni       Shape functions in point (xi,eta)   (20 * 1)
   dN_dxi   Derivative of Ni to xi              (20 * 1)
   dN_deta  Derivative of Ni to eta             (20 * 1)
   dN_dzeta Derivative of Ni to zeta            (20 * 1)

   See also KE_SOLID20.
```

sh_vs8

```
SH_VS8     Shape functions for a volume serendipity element with 8 nodes.

   [Ni,dNi_dxi,dNi_deta,dNi_dzeta] = sh_vs8(xi,eta,zeta)
   returns the shape functions and its derivatives to the natural coordinates
   in the point (xi,eta,zeta).

   xi        Natural coordinate                  (scalar)
   eta       Natural coordinate                  (scalar)
   zeta      Natural coordinate                  (scalar)
   Ni        Shape functions in point (xi,eta)   (8 * 1)
   dN_dxi    Derivative of Ni to xi              (8 * 1)
   dN_deta   Derivative of Ni to eta             (8 * 1)
   dN_dzeta  Derivative of Ni to zeta            (8 * 1)

   See also KE_SOLID8.
```

size_beam

```
SIZE_BEAM     Compute beam element size (length).

  S = SIZE_BEAM(Node) computes the element size (length) of a two node
  beam element.

  [S,dSdx] = SIZE_BEAM(Node,dNodedx) additionally computes the
  derivatives of the element size with respect to the design variables x.

  Node        Node definitions                [x y z] (3 * 3)
  dNodedx     Node definitions derivatives (SIZE(Node) * nVar)
  S           Element size
  dSdx        Element size derivatives

  See also ELEMSIZES, ELEMVOLUMES, SIZE_TRUSS.
```

size_plane6

```
size_plane6   Compute plane6 element size (area).

    s = size_plane6(NodeNum) computes the size (area) of a plane6 element.

    Node        Node definitions                    [x y z] (6 * 3)
    S           Element size
```

size_solid10

```
size_solid10   Compute solid10 element size (volume).

    S = size_solid10(NodeNum) computes the size of a solid10 element.

    Node        Node definitions                    [x y z] (10 * 3)
    S           Element size
```

size_solid4

```
 size_solid4   Compute solid4 element size (volume).

    s = size_solid4(NodeNum) computes the size (volume) of a solid4 element.

    Node        Node definitions                    [x y z] (4 * 3)
    S           Element size
```

size_truss

```
SIZE_TRUSS    Compute truss element size (length).

  S = SIZE_TRUSS(Node) computes the element size (length) of a two node
  truss element.

  [S,dSdx] = SIZE_TRUSS(Node,dNodedx) additionally computes the
  derivatives of the element size with respect to the design variables x.

  Node        Node definitions               [x y z] (3 * 3)
  dNodedx     Node definitions derivatives    (SIZE(Node) * nVar)
  S           Element size
  dSdx        Element size derivatives

  See also ELEMSIZES, ELEMVOLUMES, SIZE_BEAM.
```

tconstr

```
TCONSTR     Return matrices to apply constraint equations.

   [T,Q0,MasterDOF]=tconstr(Constr,DOF)
   returns matrices to apply constraint equations to the stiffness and mass
   matrix and the load vector: Kr=T.'*K*T, Mr=T.'*M*T and Fr=T.'*(F-K*Q0).
   The original displacement vector is computed using U=T*Ur+Q0.

   Constr     Constraint equation:
               Constant=CoefS*SlaveDOF+CoefM1*MasterDOF1+CoefM2*MasterDOF2+...
              [Constant CoefS SlaveDOF CoefM1 MasterDOF1 CoefM2 MasterDOF2 ...]
   DOF        Degrees of freedom  (nDOF * 1)
```

tloads_beam

```
TLOADS_BEAM   Equivalent nodal forces for a beam element in the GCS.

   F = tloads_beam(TLoad,Node,Section,Material)
   computes the equivalent nodal forces of a temperature load
   (in the global coordinate system).

   TLoad      Temperature loads        [dTm; dTy; dTz] (3 * 1)
   Node       Node definitions         [x y z] (2 * 3)
   Section    Section definitions      [A ky kz Ixx Iyy Izz yt yb zt zb]
   Material   Material definitions     [E nu rho alpha]
   F          Load vector              (12 * 1)

   See also ELEMTLOADS, TLOADS_TRUSS.
```

tloads_truss

```
TLOADS_TRUSS   Equivalent nodal forces for a truss element in the GCS.

   F = tloads_truss(TLoad,Node,Section,Material)
   computes the equivalent nodal forces of a temperature load
   (in the global coordinate system).

   TLoad      Temperature loads        [n1globalX; n1globalY; n1globalZ; ...]
                                                                   (6 * 1)
   Node       Node definition          [x y z] (2 * 3)
   Section    Section definition       [A ...]
   Material   Material definition      [E nu rho alpha]
   F          Load vector              (6 * 1)

   See also ELEMTLOADS, TLOADS_BEAM.
```

trans_beam

```
TRANS_BEAM   Transform coordinate system for a beam element.

   t = TRANS_BEAM(Node)
       computes the transformation matrix between the local and the global
       coordinate system for the beam element.
   [t,dtdx] = TRANS_BEAM(Node,dNodedx)
       additionally computes the derivatives of the transformation matrix
       with respect to the design variables x.

   Node       Node definitions             [x y z] (3 * 3)
   dNodedx    Node definitions derivatives         (SIZE(Node) * nVar)
   t          Transformation matrix                (3 * 3)
   dtdx       Transformation matrix derivatives    (3 * 3 * nVar)

   See also KE_BEAM, TRANS_TRUSS.
```

trans_shell2

```
TRANS_BEAM   Transform coordinate system for a beam element.

   t = trans_beam(Node)
   computes the transformation matrix between the local and the global
   coordinate system for the SHELL2 element.

   Node       Node definitions           [x y z] (3 * 3)
   t          Transformation matrix  (3 * 3)

   See also KE_BEAM, TRANS_TRUSS.
```

trans_shell4

```
TRANS_SHELL4   Transform coordinate system for a shell4 element.

   [t,Node_lc,W] = trans_shell4(Node)
   [t,Node_lc]   = trans_shell4(Node)
    t            = trans_shell4(Node)
   computes the transformation matrix between the local and the global
   coordinate system and the correction matrix for non-coplanar nodes
   for the shell4 element.

   Node       Node definitions           [x y z] (4 * 3)
   t          Transformation matrix  (3 * 3)
   Node_lc    Nodes in LCS               [x y z] (4 * 3)
   W          Correction matrix for warped elements
                                     (24 * 24)

   See also KE_BEAM, TRANS_TRUSS.
```

trans_shell8

```
TRANS_SHELL8   Transform coordinate system of a shell8 element.

   t   = trans_shell8(Node)
   t   = trans_shell8(Node,Options)
   computes the transformation matrix between the local and the global
   coordinate system for stress computations.

   Node       Node definitions           [x y z] (8 * 3)
   t          Transformation matrix  (3 * 3)
   Options    Element options struct. Fields:
              -LCSType: determine the reference local element
                        coordinate system. Values:
                        'element' (default) or 'global'

   See also SE_SHELL8, TRANS_TRUSS.
```

trans_solid20

```
TRANS_SOLID20   Transform coordinate system of a solid8 element.

   t   = trans_solid20(Node)
   t   = trans_solid20(Node,Options)
   computes the transformation matrix between the local and the global
   coordinate system for stress computations.

   Node       Node definitions           [x y z]   (20 * 3)
   t          Transformation matrix                (3 * 3)

   See also SE_SOLID20, TRANS_TRUSS.
```

trans_solid8

```
TRANS_SOLID8   Transform coordinate system of a solid8 element.

   t   = trans_solid8(Node)
   t   = trans_solid8(Node,Options)
   computes the transformation matrix between the local and the global
   coordinate system for stress computations.

   Node       Node definitions           [x y z]   (8 * 3)
   t          Transformation matrix                (3 * 3)

   See also SE_SOLID8, TRANS_TRUSS.
```

trans_truss

```
TRANS_TRUSS   Transform coordinate system for a truss element.

   t = TRANS_TRUSS(Node)
       computes the transformation matrix between the local and the global
       coordinate system for the truss element.
   [t,dtdx] = TRANS_TRUSS(Node,dNodedx)
       additionally computes the derivatives of the transformation matrix
       with respect to the design variables x.

   Node       Node definitions             [x y z] (2 * 3)
   dNodedx    Node definitions derivatives         (SIZE(Node) * nVar)
   t          Transformation matrix                (3 * 3)
   dtdx       Transformation matrix derivatives    (3 * 3 * nVar)

   See also KE_TRUSS, TRANS_BEAM.
```

udiagrgcs

```
UDIAGRGCS   Return displacement diagrams in GCS

   [UxdiagrGCS,UydiagrGCS,UzdiagrGCS] = UDIAGRGCS(Nodes,Elements,Types,DOF,U,DLoads,
                                                        Sections,Materials,Points)
   [UxdiagrGCS,UydiagrGCS,UzdiagrGCS] = UDIAGRGCS(Nodes,Elements,Types,DOF,U,DLoads,
                                                        Sections,Materials)
   [UxdiagrGCS,UydiagrGCS,UzdiagrGCS] = UDIAGRGCS(Nodes,Elements,Types,DOF,U,[],
                                                        Sections,Materials)
   [UxdiagrGCS,UydiagrGCS,UzdiagrGCS] = UDIAGRGCS(Nodes,Elements,Types,DOF,U)
       computes the displacements of the interpolation points after
       deformation in the global (algebraic) coordinate system. If DLoads,
       Sections and Materials are supplied, the displacements that occur
       due to distributed loads if all nodes are fixed, are superimposed.

   [UxdiagrGCS,UydiagrGCS,UzdiagrGCS,dUxdiagrGCSdx,dUydiagrGCSdx,dUzdiagrGCSdx]
           = UDIAGRGCS(Nodes,Elements,Types,DOF,U,DLoads,Sections,Materials,Points,
                                                  dNodesdx,dUdx,dDLoadsdx,dSectionsdx)
       additionally computes the derivatives of the displacement values
       with respect to the design variables x.

   Nodes      Node definitions          [NodID x y z]
   Elements   Element definitions       [EltID TypID SecID MatID n1 n2 ...]
   Types      Element type definitions  {TypID EltName Option1 ... }
   DOF        Degrees of freedom  (nDOF * 1)
   U          Displacements  (nDOF * 1)
   DLoads     Distributed loads         [EltID n1globalX n1globalY n1globalZ ...]
                   (use empty array [] when shear deformation (in beam element)
                   is considered but no DLoads are present)
   Sections   Section definitions       [SecID SecProp1 SecProp2 ...]
   Materials  Material definitions      [MatID MatProp1 MatProp2 ... ]
   Points     Points in the local coordinate system           (1 * nPoints)
   dNodesdx        Node definitions derivatives               (SIZE(Nodes) * nVar)
   dUdx            Displacements derivatives                  (SIZE(U) * nVar)
   dDLoadsdx       Distributed loads derivatives              (SIZE(DLoads) * nVar)
   UxdiagrGCS x-direction displacement values at the points    (nElem * nPoints * nLC)
   UydiagrGCS y-direction displacement values at the points    (nElem * nPoints * nLC)
   UzdiagrGCS z-direction displacement values at the points    (nElem * nPoints * nLC)
   dUxdiagrGCSdx   x-direction displacement values derivatives (nElem * nPoints * nLC * nVar)
   dUydiagrGCSdx   y-direction displacement values derivatives (nElem * nPoints * nLC * nVar)
   dUzdiagrGCSdx   z-direction displacement values derivatives (nElem * nPoints * nLC * nVar)

   See also PLOTDISP, DISP_TRUSS, DISP_BEAM.
```

unselectdof

```
UNSELECTDOF   Unselect degrees of freedom.

       L=unselectdof(DOF,seldof)
   [L,I]=unselectdof(DOF,seldof)
   creates the matrix to unselect degrees of freedom from the global degrees of
   freedom.

   DOF         Degrees of freedom  (nDOF * 1)
   seldof      Unselected dof labels (ndof * 1)
   L           Selection matrix ((nDOF-ndof) * nDOF)
   I           Index vector ((nDOF-ndof) * 1)

   See also SELECTDOF.
```

clearpage

volume_beam

```
 VOLUME_BEAM     Compute the volume of a beam element.

   V = VOLUME_BEAM(Node,Section) computes the volume of a two-node beam
   element.

   [V,dVdx] = VOLUME_BEAM(Node,Section,dNodedx,dSectiondx) computes the
   volume of a two node beam element, as well as the derivatives of the
   volume with respect to the design variables x.

   Node        Node definitions          [x y z] (3 * 3)
   Sections    Section definitions       [SecID SecProp1 SecProp2 ...]
   dNodedx     Node definitions derivatives    (SIZE(Node) * nVar)
   dSectionsdx Section definitions derivatives (SIZE(Section) * nVar)
   V           Element volume                  (1 * 1)
   dVdx        Element volume derivatives      (nVar * 1)

   See also ELEMVOLUMES, VOLUME_TRUSS, ELEMSIZES, SIZE_BEAM.
```

clearpage

volume_truss

```
 VOLUME_TRUSS     Compute the volume of a truss element.

   V = VOLUME_TRUSS(Node,Section) computes the volume of a two-node truss
   element.

   [V,dVdx] = VOLUME_TRUSS(Node,Section,dNodedx,dSectiondx) computes the
   volume of a two node truss element, as well as the derivatives of the
   volume with respect to the design variables x.

   Node        Node definitions          [x y z] (3 * 3)
```

```
Sections    Section definitions       [SecID SecProp1 SecProp2 ...]
dNodedx     Node definitions derivatives   (SIZE(Node) * nVar)
dSectionsdx Section definitions derivatives (SIZE(Section) * nVar)
V           Element volume                  (1 * 1)
dVdx        Element volume derivatives      (nVar * 1)

See also ELEMVOLUMES, VOLUME_BEAM, ELEMSIZES, SIZE_TRUSS.
```

vtrans_solid

```
VTRANS_SOLID   Transformation matrix for stress and strain components in matrix (Voigt) notation.

   [theta] = vtrans_solid(t)
   [theta] = vtrans_solid(t,vtype)
   computes the transformation matrix between the local and the global
   coordinate system for t stress or strain vector in matrix notation.

   t          Transformation matrix                (3 * 3)
   vtype      Vector type 'stress' (default) | 'strain'
   theta      Stress transformation matrix         (6 * 6)

   See also TRANS_SOLID8, TRANS_SOLID20.
```

wilson

```
WILSON   Direct time integration for dynamic systems - Wilson-theta method
   [u,v,a,t] = WILSON(M,C,K,dt,p,u1,v1,[alpha delta theta]) applies the
   Wilson-theta method for the calculation of the nodal displacements u,
   velocities v and accelerations a of the dynamic system with the system
   matrices M, C and K due to the excitation p.

   M    Mass matrix (nDof * nDof)
   C    Damping matrix (nDof * nDof)
   K    Stiffness matrix (nDof * nDof)
   dt   Time step of the integration scheme (1 * 1).  Should be small enough
        to ensure the stability and the precision of the integration scheme.
   p    Excitation (nDof * N).  p(:,k) corresponds to time point t(k).
   u1   Displacements at time point t(1) (nDof * 1).  Defaults to zero.
   v1   Velocities at time point t(1) (nDof * 1).  Defaults to zero.
   u    Displacements (nDof * N).  u(:,k) corresponds to time point t(k).
   t    Time axis (1 * N), defined as t = [0:N-1] * dt.
```

Bibliography

[1] K.J. Bathe. *Finite Element Procedures.* Prentice-Hall, Englewood Cliffs, NJ, second edition, 1996.

[2] A.K. Chopra. *Dynamics of structures: theory and applications to earthquake engineering.* Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

[3] R.D. Cook. *Finite element modelling for stress analysis.* John Wiley and Sons, 1995.

[4] R.D. Cook, D.S. Malkus, M.E. Plesha, and R.J. Witt. *Concepts and applications of finite element analysis.* John Wiley and Sons, fourth edition, 2002.

[5] J.S. Przemieniecki. *Theory of matrix structural analysis.* Dover Publications, New York, NY, 1985.

[6] O.C. Zienkiewicz and R.L. Taylor. *The finite element method, Volume 1: the basis.* Butterworth-Heinemann, Oxford, United Kingdom, fifth edition, 2000.

[7] O.C. Zienkiewicz and R.L. Taylor. *The finite element method, Volume 2: solid mechanics.* Butterworth-Heinemann, Oxford, United Kingdom, fifth edition, 2000.

[8] O.C. Zienkiewicz and R.L. Taylor. *The finite element method, Volume 3: fluid dynamics.* Butterworth-Heinemann, Oxford, United Kingdom, fifth edition, 2000.

[9] O.C. Zienkiewicz, R.L. Taylor, and J.Z. Zhu. *The finite element method: its basis and fundamentals.* Elsevier Butterworth-Heinemann, sixth edition, 2005.