Programmer's Manual

Mefisto 2.8

Contents

1 Introduction....3
2 Data Structures of Solid Components....4
2.1 Model Description....8
2.1.1 Structure nodes....8
2.1.2 Structure elements....9
2.1.3 Structure dofs....11
2.1.4 Structures nset and eset....13
2.1.5 Structure load....13
2.1.6 Material data....15
2.2 Analysis Data....16
2.2.1 Structure stiff....16
2.2.2 Structure mass....16
2.2.3 Structure damping....17
2.3 Results....17
2.3.1 Structure statresp....17
2.3.2 Structure modes....19
2.3.3 Structure freqresp....20
2.3.4 Structure transresp....25
2.3.5 Structure trim....27
2.3.6 Structure diverg....28
2.3.7 Structure flutter....28
2.4 Element Data....29
2.4.1 2-dimensional elements....30
2.4.2 3-dimensional elements....30
2.4.3 Element Results....33

2.5 Implementing New Elements....35
2.6 Implementing New Cross Sections....36
2.7 Implementing New Linear Constraints....36
3 Data Structures of Aerodynamic Components....37
3.1 Model Description....40
3.1.1 Structure nodes....40
3.1.2 Structure ls....41
3.1.3 Structure panels....42
3.1.4 Structure pancols....43
3.1.5 Structure controls....44
3.1.6 Structure config....44
3.1.7 Structure rbmp....46
3.2 Results....47
3.2.1 Structure statresp....47
3.2.2 Structure trim....47
3.2.3 Structure diverg....49
3.2.4 Structure flutter....49
3.2.5 Structure freqresp....49
3.2.6 Structure modes....52
4 Data Structures of Aeroelastic Components....54
4.1 Model Description....55
4.1.1 Structure splines....55
4.1.2 Structure load....56
4.2 Results....57
4.2.1 Structure flutter....57
4.2.2 Aerodynamic and structural results....59

1 Introduction

This document describes the data structures implemented in Mefisto. This information might be useful if you want to modify or extend Mefisto.

The theoretical background of the methods as well as of the element formulations can be found in the Mefisto Theory Manual.

The basic data units of Mefisto are so called components. In the simplest case an analysis model consists of only one component. Models consisting of more than one component currently occur in aeroelasticity where one component contains the data of the solid structure, one the aerodynamic data and one the aeroelastic data.

All data of a component are stored in a structure. The fields of this structure depend on the component type:

- The fields of a solid component are described in chapter 2.
- The fields of an aerodynamic component are described in chapter 3.
- The fields of an aeroelastic component are described in chapter 4.

Notation

variables typed in italic can take any name. The names of **`variables`** not typed in italic cannot be changed.

2 Data Structures of Solid Components

Overview:

```
component.type
         .subtype
         .nodes.nofnod
               .ncoor
               .maxcor(ncoor)
               .mincor(ncoor)
               .ids(:)
               .coor(nofnod, ncoor)
         .elements.nofelt
                  .noftyp
                  .types(noftyp).name
                                 .nelnod
                                 .ndofnod
                                 .mat
                                 .geom
                                 .gmshid
                                 .rtype
                                 .axes
                                 .nofelt
                                 .ixelt1
                  .index(nofelt, 3)
                  .elem(nofelt).id
                               .nodes(nelnod)
                               .coor(nelnod, ncoor)
                               .ects(:)
                               .geom
                               .mat
         .dofs.mxdofpnt
              .ndofg
              .ndofl
              .ndofp
              .ndofc
              .ndofd
              .ndofa
              .dofl(ndofl)
              .dofp(ndofp)
              .dofc(ndofc)
              .dofd(ndofd)
              .dofa(ndofa)
              .C(ndofg, ndofg)
         .nset
         .eset
         .load.nofldc
```

```
                .inflc(nofldc)
                .point(nofldc).type
                              .nofnod
                              .id(nofnod)
                              .d(nofnod, mxdofpnt)
                              .res(mxdofpnt)
                .motion(nofldc).type
                               .nofnod
                               .id(nofnod)
                               .d(nofnod, mxdofpnt)
                .inertia(nofldc).d(ncoor)
                .f(ndofg, nofldc)
                .u(ndofg, nofldc)
                .v(ndofg, nofldc)
                .a(ndofg, nofldc)
                .ar(mxdofpnt, nofldc)
        .stiff.K
               .ndofr
               .ndofe
               .dofr(ndofr)
               .dofe(ndofe)
        .mass.type
              .M
        .damping.type
                .data
        .statresp.nofldc
                 .disp(ndofg, nofldc)
                 .es(nofldc)
                 .wr(nofldc)
                 .reac(nofldc).type
                              .nofnod
                              .id(nofnod)
                              .d(nofnod, mxdofpnt)
                              .res(mxdofpnt)
                 .icsl(nofldc).type
                              .nofnod
                              .id(nofnod)
                              .d(nofnod, mxdofpnt)
                              .res(mxdofpnt)
                 .elres.subtype
                       .nofelt
                       .noftyp
                       .types(noftyp)
                       .index(nofelt, 3)
                       .nofcol
                       .elem(nofelt).id
                                    .nofpnt
```

```
                                    .coor
                                    .rtype
                                    .results
        .modes.nofmod
              .Kmodal(nofmod, nofmod)
              .omega(nofmod)
              .freq(nofmod)
              .disp(ndofg, nofmod)
              .reac(nofmod).type
                           .nofnod
                           .id(nofnod)
                           .d(nofnod, mxdofpnt)
                           .res(mxdofpnt)
              .icsl(nofmod).type
                           .nofnod
                           .id(nofnod)
                           .d(nofnod, mxdofpnt)
                           .res(mxdofpnt)
        .freqresp.nfa
                  .resp{nfa}.method
                            .type
                            .nfreq
                            .freq(nfreq)
                            .kred(nfreq)
                            .es(nfreq)
                            .wr(nfreq)
                            .Q(:, nfreq)
                            .mct(ndofg)
                            .base(ndofg)
                            .Ue(ndofg, nfreq)
                            .nback
                            .freqback(nback)
                            .U(ndofg, nback)
                            .elres.subtype
                                  .nofelt
                                  .noftyp
                                  .types(noftyp)
                                  .index(nofelt, 3)
                                  .nofcol
                                  .elem(nofelt).id
                                               .nofpnt
                                               .coor
                                               .rtype
                                               .results
        .transresp.nta
                   .resp{nta}.method
                             .ntime
```

```
                                .tsteps(ntime)
                                .nback
                                .tback(nback)
                                .q(:, ntime)
                                .qd(:, ntime)
                                .qdd(:, ntime)
                                .base(ndofg, ntime)
                                .f(:, ntime)
                                .fd(:, ntime)
                                .fdd(:, ntime)
                                .u(ndofg, ntime)
                                .v(ndofg, ntime)
                                .a(ndofg, ntime)
        .trim.nconf
             .disp(ndofg, nconf)
             .qr(6, nconf)
        .diverg.ndiv
               .qdyn(ndiv)
               .disp(ndofg, ndiv)
        .flutter.nshape
                .mode(nshape)
                .point(nshape)
                .v(nshape)
                .f(nshape)
                .kred(nshape)
                .disp(ndofg, nshape)
```

type	**String**	The model type defines the type of the mechanical field. The model type of a solid component is **"solid"**.
subtype	**String**	The subtype further specifies the model type: **"2d"** 2-dimensional solid **"3d"** 3-dimensional solid
nodes	**Structure**	Structure with the nodal point data (see page 8)
elements	**Structure**	Structure with the element data (see page 9)
dofs	**Structure**	Structure with the degree of freedom data (see page 11)
nset	**Structure**	Structure with the nodal point sets (see page 13)
eset	**Structure**	Structure with the element sets (see

		page 13)
load	**Structure**	Structure with the load data (see page 13)
stiff	**Structure**	Structure with the stiffness matrix (see page 16)
mass	**Structure**	Structure with the mass matrix (see page 16)
damping	**Structure**	Structure with the damping data (see page 17)
statresp	**Structure**	Structure with the results of static analyses (see page 17)
modes	**Structure**	Structure with the results of a normal modes analysis (see page 19)
freqresp	**Structure**	Structure with the results of frequency response analyses (see page 20)
transresp	**Structure**	Structure with the results of transient response analyses (see page 25)
trim	**Structure**	Structure with the results of an aeroelastic trim analysis (see page 27)
diverg	**Structure**	Structure with the results of a divergence analysis (see page 28)
flutter	**Structure**	Structure with the results of a flutter analysis (see page 28)

2.1 Model Description

2.1.1 Structure nodes

Created by: **mfs_new_nodes**

Fields:

```
nodes.nofnod
     .ncoor
     .maxcor(ncoor)
     .mincor(ncoor)
     .ids(nofnod)
     .coor(nofnod, ncoor)
```

nofnod	**Integer**	Number of nodal points
ncoor	**Integer**	Number of coordinates per nodal point (2 or 3)
maxcor(ncoor)	**Real**	Maxima of the coordinates
mincor(ncoor)	**Real**	Minima of the coordinates
ids(nofnod)	**Integer**	List of nodal point identifiers in ascending order
coor(nofnod, ncoor)	**Real**	Nodal point coordinates

2.1.2 Structure elements

Created by: **mfs_new_elements**

Fields:

```
elements.nofelt
        .noftyp
        .types(noftyp).name
                      .nelnod
                      .ndofnod
                      .mat
                      .geom
                      .gmshid
                      .rtype
                      .axes
                      .nofelt
                      .ixelt1
        .index(nofelt, 3)
        .elem(nofelt).id
                     .nodes(nelnod)
                     .coor(nelnod, ncoor)
                     .ects(:)
                     .geom
                     .mat
```

nofelt	**Integer**	Number of elements
noftyp	**Integer**	Number of element types
types(noftyp)	**Structure**	Structure array with the element type data (see page 10), lexically sorted according to element names
index(nofelt, 3)	**Integer**	Element index:

		(1) Element identifier (2) Index in field **elem** (3) Index of type in array **types** (sorted according to ascending element identifiers)
elem(nofelt)	**Structure**	Structure array with element data (see page 11)

Fields of the structure **types**

name	**String**	Element name
nelnod	**Integer**	Number of element nodal points
ndofnod	**Integer**	Number of degrees of freedom per nodal point
mat	**Integer**	indicates if material data are needed (0 or 1)
geom	**Integer**	indicates if geometrical data are needed (0 or 1)
gmshid	**Integer**	Gmsh geometrical type of element
rtype	**Integer**	Result type: 0 no results 1 beam results 2 rod results 3 plane stress results 4 shell results
axes	**Integer**	Axes type: 0 none 2 beam 3 triangle 4 quadrangle
nofelt	**Integer**	Number of elements of this type
ixelt1	**Integer**	Index of first element of this type in structure array **elem**

Fields of the structure **elem**

id	**Integer**	Element identifier
nodes(nelnod)	**Integer**	Indices of element nodal points in array **nodes.ids**
coor(nelnod, ncoor)	**Real**	Coordinates of the element nodal points
ects(:)	**Integer**	Element connectivity
geom	**Structure**	Structure with geometrical data
mat	**Structure**	Structure with material data

Remarks

1. The structure array **elem** is sorted such that elements of the same type are in consecutive order.
2. The element connectivity **ects** contains the indices of the degrees of freedom of the element within the set of global degrees of freedom.
3. The fields of structure **geom** depend on the element type. They are described in chapter 2.4.
4. The fields of structure **mat** depend on the material type. They are described in chapter 2.1.6.

2.1.3 Structure dofs

Created by: **mfs_new_dofs**

Fields:

```
dofs.mxdofpnt
    .ndofg
    .ndofl
    .ndofp
    .ndofc
    .ndofd
    .ndofa
    .dofl(ndofl)
    .dofp(ndofp)
    .dofc(ndofc)
    .dofd(ndofd)
    .dofa(ndofa)
    .C(ndofg, ndofg)
```

mxdofpnt	**Integer**	Maximum number of degrees of freedom per nodal point
ndofg	**Integer**	Number of global degrees of freedom
ndofl	**Integer**	Number of local degrees of freedom
ndofp	**Integer**	Number of prescribed degrees of freedom
ndofc	**Integer**	Number of coupled degrees of freedom
ndofd	**Integer**	Number of dependent degrees of freedom
ndofa	**Integer**	Number of autonomous degrees of freedom
dofl(ndofl)	**Integer**	Indices of the local degrees of freedom within the set of global degrees of freedom
dofp(ndofp)	**Integer**	Indices of the prescribed degrees of freedom within the set of global degrees of freedom
dofc(ndofc)	**Integer**	Indices of the coupled degrees of freedom within the set of global degrees of freedom
dofd(ndofd)	**Integer**	Indices of the dependent degrees of freedom within the set of global degrees of freedom
dofa(ndofa)	**Integer**	Indices of the autonomous degrees of freedom within the set of global degrees of freedom
C(ndofg, ndofg)	**Real**	Constraint matrix

Remarks

1. The set of global degrees of freedom contains **mxdofpnt** degrees of freedom for each nodal point.
2. Prescribed degrees of freedom are those degrees of freedom with prescribed displacements.
3. Local degrees of freedom are those degrees of freedom with unknown displacements.

4. Coupled degrees of freedom are those degrees of freedom shared with another substructure.
5. Dependent degrees of freedom are those degrees of freedom whose displacements are determined by linear constraints.
6. Autonomous degrees of freedom are independent degrees of freedom involved in linear constraints.
7. The current version of Mefisto does not yet support substructures, i.e. there are no coupled degrees of freedom.

2.1.4 Structures nset and eset

Created by: **`mfs_new_nset, mfs_new_eset`**

Fields:

- The names of the fields correspond to the names of the sets.
- The fields of structure **`nset`** contain the indices in array **`nodes.ids`** of the nodal points in the sets.
- The fields of structure **`eset`** contain the indices in array **`elements.index`** of the elements in the sets.

2.1.5 Structure load

Created by: **`mfs_new_load`**

Fields:

```
load.nofldc
    .inflc(nofldc)
    .point(nofldc).type
                  .nofnod
                  .id(nofnod)
                  .d(nofnod, mxdofpnt)
                  .res
    .motion(nofldc).type
                   .nofnod
                   .id(nofnod)
                   .d(nofnod, mxdofpnt)
    .inertia(nofldc).type
                    .d(ncoor)
    .f(ndofg, nofldc)
    .u(ndofg, nofldc)
    .v(ndofg, nofldc)
    .a(ndofg, nofldc)
    .ar(ncoor, nofldc)
```

nofldc	**Integer**	Number of load cases
inflc(nofldc)	**Integer**	Load case information: Bit 1: Concentrated forces or moments Bit 2: Prescribed displacements Bit 3: Prescribed velocities Bit 4: Prescribed accelerations Bit 5: Inertia loads
point(nofldc)	**Structure**	Structure array with data of concentrated forces and moments (see page 14)
motion(nofldc)	**Structure**	Structure array with data of prescribed motion (see page 15)
inertia(nofldc)	**Structure**	Structure array with data of inertia loads (see page 15)
f(ndofg, nofldc)	**Real**	Load matrix
u(ndofg, nofldc)	**Real**	Matrix with prescribed displacements
v(ndofg, nofldc)	**Real**	Matrix with prescribed velocities
a(ndofg, nofldc)	**Real**	Matrix with prescribed accelerations
ar(ncoor, nofldc)	**Real**	Matrix with rigid body accelerations

Fields of the structure **point**

type	**Integer**	Type: 0 = concentrated loads
nofnod	**Integer**	Number of nodal points with loads
id(nofnod)	**Integer**	Indices of the nodal points with loads (in ascending order)
d(nofnod, mxdofpnt)	**Real**	Load matrices at the nodal points
res(mxdofpnt)	**Real**	Resultants of loads with respect to origin

Fields of the structure **motion**

type	**Integer**	Type: 1 = displacement 2 = velocity 3 = acceleration
nofnod	**Integer**	Number of nodal points with loads
id(nofnod)	**Integer**	Indices of the nodal points with loads
d(nofnod, mxdofpnt)	**Real**	Load matrices at the nodal points

Fields of structure **inertia**

type	**Integer**	Type: 4 = inertia loads
d(ncoor)	**Real**	Acceleration vector

Remarks

1. Only fields for which data are present are created.
2. Velocities and accelerations are used in a dynamic analysis only.
3. Prescribed displacements, velocities and accelerations cannot be mixed within one load case.

2.1.6 Material data

The material data depend on the material type.

Linear isotropic material

```
mat.type
   .E
   .ny
   .rho
```

type	**Integer**	Material type: 1
E	**Real**	Young's modulus
ny	**Real**	Poisson's ratio
rho	**Real**	Mass density

Which fields are needed depends on the element type and the analysis to be performed.

2.2 Analysis Data

2.2.1 Structure stiff

Created by: **mfs_stiff**

Modified by: **mfs_freevib**

Fields:

```
stiff.K(ndofg, ndofg)
     .ndofr
     .ndofe
     .dofr(ndofr)
     .dofe(ndofe)
```

K(ndofg,ndofg)	**Real**	Stiffness matrix
ndofr	**Integer**	Number of rigid body degrees of freedom
ndofe	**Integer**	Number of elastic degrees of freedom
dofr(ndofr)	**Integer**	Indices of the rigid body degrees of freedom within the set of local degrees of freedom
dofe(ndofe)	**Integer**	Indices of the elastic degrees of freedom within the set of local degrees of freedom

Fields **ndofr**, **ndofe**, **dofr** and **dofe** are added by function **mfs_freevib**. The fields **dofr** and **dofe** are only present if the structure has rigid body degrees of freedom.

2.2.2 Structure mass

Created by: **mfs_mass**

Fields:

```
mass.type
    .M(ndofg, ndofg)
```

type	**Integer**	Type: 1 = lumped mass matrix 2 = consistent mass matrix

M(ndofg, ndofg)	**Real**	Mass matrix

2.2.3 Structure damping

Created by: **mfs_damping**

Fields:

```
damping.type
       .data
```

type	**String**	Damping type
data	**Real**	Damping data

The damping data depend on the damping type.

Rayleigh damping

type	**"Rayleigh"**
data(1)	α_K
data(2)	α_M

The damping matrix is: $[D]=\alpha_K[K]+\alpha_M[M]$

Modal damping ratios

type	**"ratios"**
data(n)	D_n

The last damping ratio defined is also used for all higher normal modes.

2.3 Results

2.3.1 Structure statresp

Created by: **mfs_statrespx, mfs_statrespc**

Modified by: **mfs_resultsx**

Fields:

```
statresp.nofldc
        .disp(ndofg, nofldc)
        .es(nofldc)
```

```
        .wr(nofldc)
        .reac(nofldc).type
                     .nofnod
                     .id(nofnod)
                     .d(nofnod, mxdofpnt)
                     .res(mxdofpnt)
        .icsl(nofldc).type
                     .nofnod
                     .id(nofnod)
                     .d(nofnod, mxdofpnt)
                     .res(mxdofpnt)
        .elres.subtype
              .nofelt
              .noftyp
              .types(noftyp)
              .index(nofelt, 3)
              .nofcol
              .elem(nofelt).id
                           .nofpnt
                           .coor(nofpnt, ncoor)
                           .rtype
                           .results
```

nofldc	**Integer**	Number of load cases
disp(ndofg, nofldc)	**Real**	Displacement matrix
es(nofldc)	**Real**	Strain energy
wr(nofldc)	**Real**	Work of residual load
reac(nofldc)	**Structure**	Structure array with reaction loads (see page 18)
icsl(nofldc)	**Structure**	Structure array with internal constraint loads (see page 19)
elres	**Structure**	Structure with element results (see page 19)

Fields of the structure **reac**

type	**Integer**	Type: 4
nofnod	**Integer**	Number of nodal points with reaction loads
id(nofnod)	**Integer**	Indices of the nodal points with reaction loads (in ascending order)

d(nofnod, mxdofpnt)	**Real**	Reaction load matrices at the nodal points
res(mxdofpnt)	**Real**	Resultants of reaction loads with respect to origin

Fields of the structure **icsl**

type	**Integer**	Type: 4
nofnod	**Integer**	Number of nodal points with reaction loads
id(nofnod)	**Integer**	Indices of the nodal points with reaction loads (in ascending order)
d(nofnod, mxdofpnt)	**Real**	Internal constraint load matrices at the nodal points
res(mxdofpnt)	**Real**	Resultants of internal constraint loads with respect to origin

Fields of the structure **elres**

subtype	**String**	Model subtype (see page 7)
nofelt	**Integer**	Number of elements
noftyp	**Integer**	Number of element types
types(noftyp)	**Structure**	Structure array with the element type data (see page 10)
index(nofelt, 3)	**Integer**	Element index (see page 9)
nofcol	**Integer**	Number of load cases
elem(nofelt)	**Structure**	Structure array with element results (see section 2.4.3)

2.3.2 Structure modes

Created by: **mfs_freevib**

Fields:

```
modes.nofmod
     .Kmodal(nofmod, nofmod)
     .omega(nofmod)
     .freq(nofmod)
```

```
     .disp(ndofg, nofmod)
     .reac(nofmod).type
                  .nofnod
                  .id(nofnod)
                  .d(nofnod, mxdofpnt)
                  .res(mxdofpnt)
     .icsl(nofmod).type
                  .nofnod
                  .id(nofnod)
                  .d(nofnod, mxdofpnt)
                  .res(mxdofpnt)
```

nofmod	**Integer**	Number of normal modes
Kmodal(nofmod, nofmod)	**Real**	Modal stiffness matrix
omega(nofmod)	**Real**	Circular eigenfrequencies
freq(nofmod)	**Real**	Eigenfrequencies
disp(ndofg, nofmod)	**Real**	Normal modes matrix
reac(nofmod)	**Structure**	Structure array with reaction loads (see page 18)
icsl(nofmod)	**Structure**	Structure array with internal constraint loads (see page 19)

Remarks

1. Normal modes are mass normalized.
2. Circular eigenfrequencies and eigenfrequencies are stored as column matrices.

2.3.3 Structure freqresp

Created by: **mfs_freqrespx**

Modified by: **mfs_backx**

Fields:

```
freqresp.nfa
        .resp{nfa}
```

nfa	**Integer**	Number of analyses
resp{nfa}	**Cell**	Cell array with responses

For each analysis, the cell array **resp** contains a structure the fields of which depend on the analysis method.

Direct frequency response analysis

method	**Integer**	Method: 1
type	**Integer**	Load type: 1 Force or moment 2 Prescribed displacement 3 Prescribed velocity 4 Prescribed acceleration 11 Aeroelastic results
nfreq	**Integer**	Number of excitation frequencies
freq(nfreq)	**Real**	List of excitation frequencies
kred(nfreq)	**Real**	List of reduced frequencies
es(nfreq)	**Real**	Strain energy
wr(nfreq)	**Real**	Work of residual load
nback	**Integer**	same as **nfreq**
freqback(nback)	**Real**	same as **freq**
U(ndofg, nfreq)	**Complex**	Displacement response matrix
elres	**Structure**	Structure with element results (see page 24)

nback and **freqback** are included for compatibility with modal frequency response results.

Modal frequency response analysis

method	**Integer**	Method: 2
type	**Integer**	Load type 1 Force or moment 2 Prescribed displacement 3 Prescribed velocity 4 Prescribed acceleration 11 Aeroelastic results

nfreq	**Integer**	Number of excitation frequencies
freq(nfreq)	**Real**	List of excitation frequencies
kred(nfreq)	**Real**	List of reduced frequencies
es(nfreq)	**Real**	Strain energy
Q(:, nfreq)	**Complex**	Response of modal coordinates
base(ndofg)	**Real**	Base motion
nback	**Integer**	Number of excitation frequencies of which physical displacements are available from back transformation
freqback(nback)	**Real**	Excitation frequencies of which physical displacements are available from back transformation
U(ndofg, nback)	**Complex**	Physical displacements from back-transformation
elres	**Structure**	Structure with element results (see page 24)

Enhanced modal frequency response analysis

method	**Integer**	Method: 3
type	**Integer**	Load type: 1 Force or moment 2 Prescribed displacement 3 Prescribed velocity 4 Prescribed acceleration
nfreq	**Integer**	Number of excitation frequencies
freq(nfreq)	**Real**	List of excitation frequencies
es(nfreq)	**Real**	Strain energy
Q(:, nfreq)	**Complex**	Response of modal coordinates
mct(ndofg)	**Real**	Modal truncation correction matrix
base(ndofg)	**Real**	Base motion
nback	**Integer**	Number of excitation frequencies of which physical displacements are available from back-transformation

freqback(nback)	**Real**	Excitation frequencies of which physical displacements are available from back-transformation
U(ndofg, nback)	**Complex**	Physical displacements from back-transformation
elres	**Structure**	Structure with element results (see page 24)

The base motion is computed in case of enforced motion. Depending on the type of enforced motion, one of the following matrices is stored:

- Prescribed displacements: $[\text{base}]=\left[U_G^B\right]=\begin{bmatrix}-[K_{LL}]^{-1}[K_{LP}][U_P]\\ [U_P]\end{bmatrix}$
- Prescribed velocities: $[\text{base}]=\left[V_G^B\right]=\begin{bmatrix}-[K_{LL}]^{-1}[K_{LP}][V_P]\\ [V_P]\end{bmatrix}$
- Prescribed accelerations: $[\text{base}]=\left[A_G^B\right]=\begin{bmatrix}-[K_{LL}]^{-1}[K_{LP}][A_P]\\ [A_P]\end{bmatrix}$

The modal truncation correction matrix reads

$$[\text{mct}]=\begin{bmatrix}[K_{LL}]^{-1}\left([Y_L]-[M_{LL}][X_L][X_L]^T[Y_L]\right)\\ [0]\end{bmatrix}$$

where

$[Y_L]=[L_L]$ (load matrix)

for load type 1 and

$[Y_L]=-[M_{LG}][\text{base}]$

in case of enforced motion (load types 2 to 4).

Force summation method

method	**Integer**	Method: 4
type	**Integer**	Load type: 11 Aeroelastic results
nfreq	**Integer**	Number of excitation frequencies
freq(nfreq)	**Real**	List of excitation frequencies
kred(nfreq)	**Real**	List of reduced frequencies

Q(:, nfreq)	**Complex**	Response of modal coordinates
Ue(ndofg, nfreq)	**Complex**	Elastic displacements with respect to mean axes system
nback	**Integer**	Number of excitation frequencies of which physical displacements are available from back-transformation
freqback(nback)	**Real**	Excitation frequencies of which physical displacements are available from back-transformation
U(ndofg, nback)	**Complex**	Physical displacements from back-transformation
elres	**Structure**	Structure with element results (see page 24)

Fields of the structure **elres**

subtype	**String**	Model subtype (see page 7)
nofelt	**Integer**	Number of elements
noftyp	**Integer**	Number of element types
types(noftyp)	**Structure**	Structure array with the element type data (see page 10)
index(nofelt, 3)	**Integer**	Element index (see page 9)
nofcol	**Integer**	Number of excitation frequencies
elem(nofelt)	**Structure**	Structure array with element results (see section 2.4.3)

Remarks

1. The force summation method is used in aeroelasticity.
2. Reduced frequencies are only available with load type 11 (aeroelastic results).
3. Strain energies are not available for load type 11.
4. Element results are computed for those excitation frequencies for which the back-transformation has been performed.
5. Beam, rod and shell results are computed at the midpoint of the element.

2.3.4 Structure transresp

Created by: **mfs_transrespx**

Modified by: **mfs_backx**

Fields:

```
transresp.nta
         .resp{nta}
```

nta	**Integer**	Number of analyses
resp{nta}	**Cell**	Cell array with responses

For each analysis, the cell array **resp** contains a structure the fields of which depend on the analysis method.

Direct transient response analysis

method	**Integer**	Method: 1
ntime	**Integer**	Number of time steps
tsteps(ntime)	**Real**	List of time steps
nback	**Integer**	same as **ntime**
tback(nback)	**Real**	same as **tsteps**
u(ndofg, ntime)	**Real**	Matrix with displacements
v(ndofg, ntime)	**Real**	Matrix with velocities
a(ndofg, ntime)	**Real**	Matrix with accelerations

nback and **freqback** are included for compatibility with modal transient response results.

Modal transient response analysis

method	**Integer**	Method: 2
ntime	**Integer**	Number of time steps
tsteps(ntime)	**Real**	List of time steps
q(:, ntime)	**Real**	Matrix with modal coordinates
qd(:, ntime)	**Real**	Matrix with modal velocities
qdd(:, ntime)	**Real**	Matrix with modal accelerations
base(ndofg, :)	**Real**	Base motion

f(:, ntime)	**Real**	Functions defining time dependency of prescribed motion
fd(:, ntime)	**Real**	First derivatives of functions **f**
fdd(:, ntime)	**Real**	Second derivatives of functions **f**
nback	**Integer**	Number of time steps of which physical displacements are available from back-transformation
tback(nback)	**Real**	Time steps at which physical displacements are available from back-transformation
u(ndofg, nback)	**Real**	Matrix with displacements
v(ndofg, nback)	**Real**	Matrix with velocities

Enhanced modal transient response analysis

method	**Integer**	Method: 3
ntime	**Integer**	Number of time steps
tsteps(ntime)	**Real**	List of time steps
q(:, ntime)	**Real**	Matrix with modal coordinates
qd(:, ntime)	**Real**	Matrix with modal velocities
qdd(:, ntime)	**Real**	Matrix with modal accelerations
base(ndofg, :)	**Real**	Base motion
f(:, ntime)	**Real**	Functions defining time dependency of prescribed motion
fd(:, ntime)	**Real**	First derivatives of functions **f**
fdd(:, ntime)	**Real**	Second derivatives of functions **f**
Xs(:, :)	**Real**	Static mode shapes
nback	**Integer**	Number of time steps of which physical displacements are available from back-transformation
tback(nback)	**Real**	Time steps at which physical displacements are available from back-transformation
u(ndofg, nback)	**Real**	Matrix with displacements
v(ndofg, nback)	**Real**	Matrix with velocities

Force summation method

method	**Integer**	Method: 4
ntime	**Integer**	Number of time steps
tsteps(ntime)	**Real**	List of time steps
q(:, ntime)	**Real**	Matrix with modal coordinates
qd(:, ntime)	**Real**	Matrix with modal velocities
qdd(:, ntime)	**Real**	Matrix with modal accelerations
ue(ndofg, ntime)	**Real**	Elastic displacements with respect to mean axes system
ve(ndofg, ntime)	**Real**	Elastic velocities with respect to mean axes system
ae(ndofg, ntime)	**Real**	Elastic accelerations with respect to mean axes system
nback	**Integer**	Number of time steps of which physical displacements are available from back-transformation
tback(nback)	**Real**	Time steps at which physical displacements are available from back-transformation
u(ndofg, nback)	**Real**	Matrix with displacements
v(ndofg, nback)	**Real**	Matrix with velocities

Remarks

1. The base motion is computed in case of enforced motion.
2. Physical displacements and velocities are computed by **mfs_backx**.
3. The force summation method is used in aeroelasticity.

2.3.5 Structure trim

Created by: **mfs_flextrim**

Modified by: **mfs_resultsx**

Fields:

```
trim.nconf
    .disp(ndofg, nconf)
```

```
    .qr(6, nconf)
    .elres.subtype
          .nofelt
          .noftyp
          .types(noftyp)
          .index(nofelt, 3)
          .nofcol
          .elem(nofelt).id
                       .nofpnt
                       .coor(nofpnt, ncoor)
                       .rtype
                       .results
```

nconf	**Integer**	Number of configurations
disp(ndofg, nconf)	**Real**	Displacement matrix
qr(6, nconf)	**Real**	Rigid body motion parameters
elres	**Structure**	Structure with element results (see page 19)

Rigid body motion parameters describe the translation and rotation of the mean axes system in case of an unrestrained analysis.

2.3.6 Structure diverg

Created by: **mfs_diverg**

Fields:

```
diverg.ndiv
      .qdyn(ndiv)
      .disp(ndofg, ndiv)
```

ndiv	**Integer**	Number of divergence cases
qdyn(ndiv)	**Real**	Dynamic pressure at divergence
disp(ndofg, ndiv)	**Real**	Displacement matrix

2.3.7 Structure flutter

Created by: **mfs_backc**

Fields:

```
flutter.nshape
       .mode(nshape)
       .point(nshape)
```

```
        .v(nshape)
        .f(nshape)
        .kred(nshape)
        .disp(ndofg, nshape)
```

nshape	**Integer**	Number of flutter shapes
mode(nshape)	**Integer**	Flutter mode numbers
point(nshape)	**Integer**	Flutter point numbers
v(nshape)	**Real**	Flutter velocities
f(nshape)	**Real**	Flutter frequencies
kred(nshape)	**Real**	Reduced frequencies
disp(ndofg, nshape)	**Complex**	Displacement matrix

2.4 Element Data

List of Variables:

nelnod	Number of element nodes		
ndofnod	Number of degrees of freedom per node		
mat	indicates if material data are needed (1) or not (0)		
geom	geometrical data		
rtype	Response type:	0	no results
		1	beam results
		2	rod results
		3	plane stress
		4	shell results
axes	Axis type:	0	no axes
		2	beam axes
		3	triangle
		4	quadrangle
gmshid	Element type identifier in Gmsh		

2.4.1 2-dimensional elements

Name	**nelnod**	**ndofnod**	**mat**	**geom**	**rtype**	**axes**	**gmshid**
"r2"	2	2	1	**A**	2	0	1
"b2"	2	3	1	**A**	1	0	1
				I			
"t3"	3	2	1	**t**	3	0	2
"t6"	6	2	1	**t**	3	0	9
"q4"	4	2	1	**t**	3	0	3
"q8"	8	2	1	**t**	3	0	16
"q9"	9	2	1	**t**	3	0	10
"t3r"	3	3	1	**t**	3	0	2
"q4r"	4	3	1	**t**	3	0	3
"q4e"	4	2	1	**t**	3	0	3
"m1"	1	2	0	**m**	0	0	15
"m2"	1	3	0	**m**	0	0	15
				J			
"g2"	2	0	0	**0**	0	0	1

Fields of structure **geom**

A	Cross section area
I	Area moment of inertia $I_z = \int_A y^2 \, dA$
t	Thickness
m	Mass
J	Mass moment of inertia $J_z = \int_K (x^2 + y^2) \, dm$

2.4.2 3-dimensional elements

Name	**nelnod**	**ndofnod**	**mat**	**geom**	**rtype**	**axes**	**gmshid**
"r2"	2	3	1	**A**	2	0	1
"b2"	2	6	1	**A**	1	2	1

Name	nelnod	ndofnod	mat	geom	rtype	axes	gmshid
				Iy Iz Iyz IT v(3) CE(2) P(2)			
"t3"	3	3	1	t wtol	3	3	2
"q4"	4	3	1	t wtol	3	4	3
"t3r"	3	6	1	t wtol	3	3	2
"q4r"	4	6	1	t wtol	3	4	3
"s3"	3	6		t zo ka l2trat wtol	4	3	2
"s4"	4	6	1	t zo ka l2trat wtol	4	4	3
"s3r"	3	6		t zo l2trat wtol	4	3	2
"s4r"	4	6	1	t	4	4	3

Name	**nelnod**	**ndofnod**	**mat**	**geom**	**rtype**	**axes**	**gmshid**
				zo			
				l2trat			
				wtol			
"h8"	8	3	1		5	0	5
"h20"	20	3	1		5	0	17
"h27"	27	3	1		5	0	12
"p6"	6	3	1		5	0	6
"p15"	15	3	1		5	0	18
"p18"	18	3	1		5	0	13
"m1"	1	3	0	**m**	0	0	15
"m2"	1	3	0	**m**	0	0	15
				Jx			
				Jy			
				Jz			
				Jxy			
				Jxz			
				Jyz			
"g2"	2	0	0	**0**	0	0	1

Fields of structure **geom**

A Cross section area

Iy Area moment of inertia $I_y = \int_A z^2 \, dA$

Iz Area moment of inertia $I_z = \int_A y^2 \, dA$

Iyz Moment of deviation $I_{yz} = -\int_A y \, z \, dA$

IT Torsional constant

v(3) Vector in the $x_E z_E$-plain of the element

CE(2) Coordinates of the shear centre in the element coordinate system

P(2) Coordinates of the nodal points in the element coordinate system

t	Thickness
wtol	Warping tolerance (not used for triangular elements)
zo	Offset
ka	Factor for artificial stiffness
l2trat	Length to thickness ratio
m	Mass
Jx	Mass moment of inertia about *x*-axis $J_x = \int_K (y^2 + z^2)\, dm$
Jy	Mass moment of inertia about *y*-axis $J_y = \int_K (x^2 + z^2)\, dm$
Jz	Mass moment of inertia about *y*-axis $J_z = \int_K (x^2 + y^2)\, dm$
Jxy	Mass moment of deviation $J_{xy} = -\int_K x\, y\, dm$
Jxz	Mass moment of deviation $J_{xz} = -\int_K x\, z\, dm$
Jyz	Mass moment of deviation $J_{yz} = -\int_K y\, z\, dm$

Remarks

1. The warping tolerance controls if the warping correction is applied.
2. The length to thickness ratio controls the algorithm to adjust the shear modulus for the transverse shear in order to ensure convergence to the Kirchhoff theory (see the Theory Manual for details).

2.4.3 Element Results

Element results are stored in structure array **elem** which is a field of structure **elres**. Each element of the structure array **elem** contains the results of one element.

Structure **elem** has the following fields:

id	**Integer**	Element identifier
nofpnt	**Integer**	Number of result points
coor(nofpnt, ncoor)	**Real**	Coordinates of result points
rtype	**Integer**	Result type:

		0 no results
		1 beam results
		2 rod results
		3 plane stress results
		4 shell results
		5 general stress results
results	**Structure**	Structure with element results

The fields of structure **results** depend on the result type and the dimension of the problem:

rtype	Dim.	Field	Contents
0	2/3		No results
1	2	**rslt(3,nofcol)**	Stress resultants: N, Q_y, M_z
	3	**rslt(6,nofcol)**	Stress resultants: N, Q_y, Q_z, M_x, M_y, M_z
2	2/3	**sig(nofcol)**	Stress
		eps(nofcol)	Strain
		rslt(nofcol)	Normal force
3	2/3	**sig(3,nofcol, nofpnt)**	Stresses σ_x, σ_y, τ_{xy}
	2/3	**eps(3,nofcol, nofpnt)**	Strains ε_x, ε_y, γ_{xy}
	3	**TE(3, 3)**	Transformation matrix from global to element coordinate system
4	3	**sig(3,nofcol, 2)**	Stresses σ_x, σ_y, τ_{xy} on the upper and lower side of the element
		eps(3,nofcol, 2)	Strains ε_x, ε_y, γ_{xy} on the upper and lower side of the element
		rslt(8,nofcol)	Stress resultants N_x, N_y, N_{xy}, M_x, M_y, M_{xy}, Q_x, Q_y
		TE(3, 3)	Transformation matrix from global to element coordinate system
5	3	**sig(6,nofcol, nofpnt)**	Stresses σ_x, σ_y, σ_z, τ_{xy}, τ_{yz}, τ_{xz}
		eps(6,nofcol,	Strains ε_x, ε_y, ε_z, γ_{xy}, γ_{yz}, γ_{xz}

rtype	Dim.	Field	Contents
		nofpnt)	

Remarks

1. Beam, rod and shell results are computed at the midpoint of the element.
2. Results from a frequency response analysis are complex.
3. **nofcol** is the number of load cases, the number of excitation frequencies or the number of time steps, depending on the type of analysis.

2.5 Implementing New Elements

To implement a new element, only the functions related to the new element need be coded. There is no need to change the existing code.

All functions specific to one element are located in a directory the name of which equals the name of the element. In case of a 2-dimensional element, this directory is a subdirectory of `src/solid/elem/2d`, and in case of a 3-dimensional element, of `src/solid/elem/3d`.

Currently, the following functions may be coded:

mfs_defelt	defines the characteristic data of the element
mfs_chkelt	performs basic element checks
mfs_ke	computes the element stiffness matrix
mfs_mce	computes the consistent mass matrix
mfs_mle	computes the lumped mass matrix
mfs_pltelt	plots the elements
mfs_prtelt	prints the element results
mfs_reselt	computes the element results of a static response
mfs_reseltf	computes the element results of a frequency response

Function **mfs_defelt** is always required. All other functions need only be coded if they are needed by the element.

Templates of these functions, with a description of the input and output arguments, can be found in directory `templates`.

2.6 Implementing New Cross Sections

The functions for calculating the cross section properties can be found in directory `src/solid/sections`. The names of the functions read

`mfs_`*`sectionname`*`_section`

where ***`sectionname`*** is the name of the cross section (argument **`type`** of function **`mfs_beamsection`**).

To implement a new cross section, it is only necessary to add the corresponding function to this directory. The file **`mfs_template_section.m`** contains a template that explains what has to be coded.

2.7 Implementing New Linear Constraints

To implement a new linear constraint only one function, called **`mfs_cs`**, need be coded. The function must be stored in a directory whose name is the same as the name of the constraint. The directory is a subdirectory of `src/solid/lincon/2d` for 2-dimensional structures and of `src/solid/lincon/3d` for 3-dimensional structures. Subdirectory `template` contains a template of this function, explaining the input and output arguments and what has to be coded.

3 Data Structures of Aerodynamic Components

Overview:

```
component.type
         .subtype
         .cref
         .nodes.nofnod
               .ncoor
               .maxcor(ncoor)
               .mincor(ncoor)
               .ids(:)
               .coor(nofnod, ncoor)
         .ls.nofls
            .ids(nofls)
            .surfs(nofls).nx
                         .ny
                         .P(2, 3)
                         .c(2)
                         .a(2)
                         .camber{2}
                         .nvec(3)
                         .tx(:)
                         .ty(:)
                         .p1
                         .pend
                         .gid
         .panels.nofpan
                .corner(4, nofpan)
                .A(3, nofpan)
                .B(3, nofpan)
                .C(3, nofpan)
                .nvec(3, nofpan)
                .area(nofpan)
                .width(nofpan)
                .slope(nofpan)
         .pancols.npcol
                 .ng
                 .panids{npcol}
                 .gcx(2, ng)
                 .tepcor(3, 2, npcol)
         .symy
         .controls.ncntrl
                  .cntnam.nofls
                         .P1(nofls, 3)
                         .P2(nofls, 3)
                         .pids(nofls, 2)
```

```
                    .factors(nofls)
        .config.nconf
               .ncntrl
               .nlinc(nconf)
               .names{nconf}
               .qdyn(nconf)
               .ax(nconf)
               .ay(nconf)
               .az(nconf)
               .racce(nconf)
               .pacce(nconf)
               .yacce(nconf)
               .alpha(nconf)
               .beta(nconf)
               .pitch(nconf)
               .yaw(nconf)
               .roll(nconf)
               .cntls.cntnam(nconf)
               .tpdef.tpnam
               .lincon{nconf}
        .rbmp.m
             .JS(3, 3)
             .cm
        .statresp.nconf
                 .G(nofpan, nconf)
                 .panres.nofpan
                        .P(3, nofpan)
                        .f(3, nofpan, nconf)
                        .p(nofpan, nconf)
                 .disp(:, nconf)
        .trim.nconf
             .ntp
             .tpnames{ntp}
             .tpval(ntp, nconf)
             .isangle(ntp)
             .G(:, nconf)
             .panres.nofpan
                    .P(3, nofpan)
                    .f(3, nofpan, nconf)
                    .p(nofpan, nconf)
             .disp(:, nconf)
             .qr(6, nconf)
        .diverg.ndiv
               .qdyn(ndiv)
               .disp(:, ndiv)
        .flutter.nshape
                .mode(nshape)
```

```
                .point(nshape)
                .v(nshape)
                .f(nshape)
                .kred(nshape)
                .disp(ndofg, nshape)
        .freqresp.nfa
                 .resp{nfa}.method
                           .type
                           .nfreq
                           .kred(nfreq)
                           .freq(nfreq)
                           .G(nofpan, nfreq)
                           .panres.nofpan
                                  .P(3, nofpan)
                                  .f(3, :, nfreq)
                                  .p(:, nfreq)
                           .nback
                           .freqback(nback)
                           .U(:, nfreq)
                           .Q(:, nfreq)
                           .Ue(:, nfreq)
        .modes.nofmod
              .rigmod
              .omega(nofmod)
              .freq(nofmod)
              .disp(:, nofmod)
```

type	**String**	The model type defines the type of the mechanical field. The model type of an aerodynamic component is **"aero"**.
subtype	**String**	The subtype further specifies the model type: **"vlm"** Vortex-Lattice-Method
cref	**Real**	Reference length
nodes	**Structure**	Structure with the nodal point data (see page 40)
ls	**Structure**	Structure with lifting surface data (see page 41)
panels	**Structure**	Structure with panel data (see page 42)
pancols	**Structure**	Structure with data of panel columns (see page 43)

symy	**Real**	*y*-coordinate of plane of symmetry (if present)
controls	**Structure**	Structure with control surface data (see page 44)
config	**Structure**	Structure with configuration data (see page 44)
rbmp	**Structure**	Structure with rigid body mass properties (see page 46)
statresp	**Structure**	Structure with results of static analyses (see page 47)
trim	**Structure**	Structure with results of trim analyses (see page 47)
diverg	**Structure**	Structure with results of a divergence analysis (see page 49)
flutter	**Structure**	Structure with results of a flutter analysis (see page 49)
freqresp	**Structure**	Structure with results of frequency response analyses (see page 49)
modes	**Structure**	Structure with normal modes of the solid (see page 52)

3.1 Model Description

3.1.1 Structure nodes

Created by: **mfs_new_ls**

Fields:

```
nodes.nofnod
     .ncoor
     .maxcor(ncoor)
     .mincor(ncoor)
     .ids(nofnod)
     .coor(nofnod, ncoor)
```

nofnod	**Integer**	Number of nodal points
ncoor	**Integer**	Number of coordinates per nodal point (always 3)

maxcor(ncoor)	**Real**	Maxima of the coordinates
mincor(ncoor)	**Real**	Minima of the coordinates
ids(nofnod)	**Integer**	List of nodal point identifiers in ascending order
coor(nofnod, ncoor)	**Real**	Nodal point coordinates

3.1.2 Structure ls

Created by: **mfs_new_ls**

Modified by: **mfs_new_pancols**

Fields:

```
ls.nofls
  .ids(nofls)
  .surfs(nofls).nx
               .ny
               .P(2, 3)
               .c(2)
               .a(2)
               .camber{2}
               .nvec(3)
               .tx(:)
               .ty(:)
               .p1
               .pend
               .gid
```

nofls	**Integer**	Number of lifting surfaces
ids(nofls)	**Integer**	Identifiers of lifting surfaces in ascending order
surfs(nofls)	**Structure**	Lifting surface data (see page 41)

Fields of structure **surfs**

nx	**Integer**	Number of panels in *x*-direction
ny	**Integer**	Number of panels in *y*-direction
P(2, 3)	**Integer**	Coordinates of points P1 and P2 defining the leading edge of the lifting surface
c(2)	**Real**	Chord length at P1 and P2

a(2)	**Real**	Angle of attack at P1 and P2 (in radians)
camber{2}	**Cell Array**	Piecewise polynomial structures defining the camber at P1 and P2
nvec(3)	**Real**	Normal vector
tx(nx + 1)	**Real**	Parametric positions of panel nodes in *x*-direction
ty(ny + 1)	**Real**	Parametric positions of panel nodes in *y*-direction
p1	**Integer**	Index of first panel of this lifting surface in structure **panels**
pend	**Integer**	Index of last panel of this lifting surface in structure **panels**
gid	**Integer**	Panel column group identifier

Remarks

1. A panel column group contains all panel columns that share the same trailing edge.
2. Field **gid** is inserted by function **mfs_new_pancols**.

3.1.3 Structure panels

Created by: **mfs_new_ls**

Fields:

```
panels.nofpan
      .corner(4, nofpan)
      .A(3, nofpan)
      .B(3, nofpan)
      .C(3, nofpan)
      .nvec(3, nofpan)
      .area(nofpan)
      .width(nofpan)
      .slope(nofpan)
```

nofpan	**Integer**	Number of panels
corner(4, nofpan)	**Integer**	Indices of panel corner nodes (see Figure 3.1-1) in structure **nodes** (see page 40)

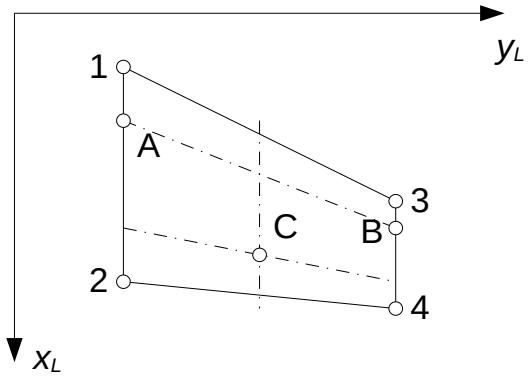


Figure 3.1-1: Panel Geometry

A(3, nofpan)	**Real**	Coordinates of vortex points A
B(3, nofpan)	**Real**	Coordinates of vortex points B
C(3, nofpan)	**Real**	Coordinates of control points C
nvec(3, nofpan)	**Real**	Panel normal vectors
area(nofpan)	**Real**	Panel areas
width(nofpan)	**Real**	Panel widths
slope(nofpan)	**Real**	Panel slopes computed from camber and angle of incidence

3.1.4 Structure pancols

Created by: **mfs_new_pancols**

Fields:

```
pancols.npcol
       .ng
       .panids{npcol}
       .gcx(2, ng)
       .tepcor(3, 2, npcol)
```

npcol	**Integer**	Number of panel columns
ng	**Integer**	Number of panel column groups
panids{npcol}	**Cell array**	Lists of panels in each column
gcx(2, ng)	**Integer**	Indices of first and last panel columns in panel column group

tepcor(3, 2, npcol)	**Real**	Coordinates of trailing edge points 2 and 4 (cf. Figure 3.1-1) of the last panel in each column

Remarks

1. A panel column group contains all panel columns that share the same trailing edge.

3.1.5 Structure controls

Created by: **mfs_new_controls**

Fields:

```
controls.ncntrl
        .cntnam.nofls
               .P1(nofls, 3)
               .P2(nofls, 3)
               .pids(nofls, 2)
               .factors(nofls)
```

ncntrl	**Integer**	Number of control surfaces
cntnam	**Structure**	Control surface data (see page 44); the field name equals the name of the control surface as defined in the input data

Fields of structure ***cntnam***

nofls	**Integer**	Number of lifting surfaces
P1(nofls, 3)	**Real**	Coordinates of leading edge point P1
P2(nofls, 3)	**Real**	Coordinates of leading edge point P2
pids(nofls, 2)	**Integer**	Index of first and last panel of each of the lifting surfaces of the control surface
factors(nofls)	**Real**	Factors for lifting surfaces

3.1.6 Structure config

Created by: **mfs_new_config**

Fields:

```
config.nconf
      .ncntrl
      .nlinc(nconf)
      .names{nconf}
      .qdyn(nconf)
      .ax(nconf)
      .ay(nconf)
      .az(nconf)
      .racce(nconf)
      .pacce(nconf)
      .yacce(nconf)
      .alpha(nconf)
      .beta(nconf)
      .pitch(nconf)
      .yaw(nconf)
      .roll(nconf)
      .cntls.cntnam(nconf)
      .tpdef.tpnam
      .lincon{nconf}
```

nconf	**Integer**	Number of configurations
ncntrl	**Integer**	Number of control surfaces
nlinc(nconf)	**Integer**	Number of linear constraints of each configuration
names{nconf}	**Cell Array**	Configuration names
qdyn(nconf)	**Real**	Dynamic pressure
ax(nconf)	**Real**	Linear acceleration in *x*-direction
ay(nconf)	**Real**	Linear acceleration in *y*-direction
az(nconf)	**Real**	Linear acceleration in *z*-direction
racce(nconf)	**Real**	Roll acceleration
pacce(nconf)	**Real**	Pitch acceleration
yacce(nconf)	**Real**	Yaw acceleration
alpha(nconf)	**Real**	Angle of attack (in radians)
beta(nconf)	**Real**	Sideslip angle (in radians)
pitch(nconf)	**Real**	Pitch rate divided by velocity
yaw(nconf)	**Real**	Yaw rate divided by velocity
roll(nconf)	**Real**	Roll rate divided by velocity
cntls	**Structure**	Control surface angles (see page 46)
tpdef	**Structure**	Information on which trim parameters

		are defined (see page 46)
lincon{nconf}	**Cell Array**	Linear constraints (see page 46)

Fields of structure **cntls**

***cntnam*(nconf)**	**Real**	Control surface angles in radians; the field names equal the names of the control surfaces

Fields of the structure **tpdef**

tpnam	**Integer**	The *n*-th bit is 1 if the trim parameter with name ***tpnam*** is defined for the *n*-th configuration

Linear constraints

For each configuration, cell array **lincon** contains a cell array of structures defining the constraints. The constraints are linear equations of the form

$$\sum_n c_n t_n = r \quad .$$

Each structure defines one linear constraint. The field names correspond to the names of the trim parameters t_n. The values of the fields define the coefficients c_n of the linear equation. The value r of the right-hand side is defined in field **rhs**.

3.1.7 Structure rbmp

Created by: **mfs_new_rbmp**

Fields:

```
rbmp.m
    .JS(3, 3)
    .cm(3)
```

m	**Real**	Mass
JS(3, 3)	**Real**	Inertia tensor with respect to centre of mass
cm(3)	**Real**	Coordinates of centre of mass

3.2 Results

3.2.1 Structure statresp

Created by: **mfs_statrespx, mfs_statrespc**

Modified by: **mfs_resultsx**

Fields:

```
statresp.nconf
        .G(nofpan, nconf)
        .panres.nofpan
               .P(3, nofpan)
               .f(3, nofpan, nconf)
               .p(nofpan, nconf)
        .disp(:, nconf)
```

nconf	**Integer**	Number of configurations
G(nofpan, nconf)	**Real**	Vortex strengths
panres	**Structure**	Panel results (see page 47)
disp(:, nconf)	**Real**	Displacements of aerodynamic nodes

Fields of structure **panres**

nofpan	**Integer**	Number of panels
P(3, nofpan)	**Real**	Coordinates of point in the middle between points A and B
f(3, nofpan, nconf)	**Real**	Panel force vectors
p(nofpan, nconf)	**Real**	Panel pressure

Field **panres** is added by function **mfs_resultsx**.

Remarks

1. Displacements are only available in an aeroelastic analysis.

3.2.2 Structure trim

Created by: **mfs_rigtrim, mfs_flextrim**

Modified by: **mfs_resultsx**

Fields:

```
trim.type
    .nconf
    .ntp
    .tpnames{ntp}
    .tpval(ntp, nconf)
    .isangle(ntp)
    .G(:, nconf)
    .panres.nofpan
           .P(3, nofpan)
           .f(3, nofpan, nconf)
           .p(nofpan, nconf)
    .disp(:, nconf)
    .qr(6, nconf)
```

type	**String**	Trim analysis type (see page 48)
nconf	**Integer**	Number of configurations
ntp	**Integer**	Number of trim parameters
tpnames{ntp}	**Cell Array**	Names of trim parameters
tpval(ntp, nconf)	**Real**	Trim parameter values
isangle(ntp)	**Integer**	Array indicating if trim parameters are angles (0 or 1)
G(nofpan, nconf)	**Real**	Vortex strengths
panres	**Structure**	Panel results (see page 47)
disp(:, nconf)	**Real**	Displacements of aerodynamic nodes
qr(6, nconf)	**Real**	Rigid body parameters

Trim analysis types

"rigid"	Trim analysis of rigid aircraft
"restrained"	Restrained trim analysis of flexible aircraft
"unrestrained"	Unrestrained trim analysis of flexible aircraft

Remarks

1. Field **panres** is added by function **mfs_resultsx**.
2. Displacements are only available in an aeroelastic trim analysis.
3. Rigid body parameters are only available in an unrestrained aeroelastic trim analysis.

3.2.3 Structure diverg

Created by: **mfs_diverg**

Fields:Created by: **mfs_diverg**

Fields:

```
diverg.ndiv
      .qdyn(ndiv)
      .disp(:, ndiv)
```

ndiv	**Integer**	Number of divergence cases
qdyn(ndiv)	**Real**	Dynamic pressure at divergence
disp(:, ndiv)	**Real**	Displacement matrix

3.2.4 Structure flutter

Created by: **mfs_backc**

Fields:

```
flutter.nshape
       .mode(nshape)
       .point(nshape)
       .v(nshape)
       .f(nshape)
       .kred(nshape)
       .disp(ndofg, nshape)
```

nshape	**Integer**	Number of flutter shapes
mode(nshape)	**Integer**	Flutter mode numbers
point(nshape)	**Integer**	Flutter point numbers
v(nshape)	**Real**	Flutter velocities
f(nshape)	**Real**	Flutter frequencies
kred(nshape)	**Real**	Reduced frequencies
disp(ndofg, nshape)	**Complex**	Displacement matrix

3.2.5 Structure freqresp

Created by: **mfs_freqrespx, mfs_freqrespc**

Modified by: **mfs_resultsx**

Fields:

freqresp.nfa
.resp{nfa}

nfa	**Integer**	Number of analyses
resp{nfa}	**Cell**	Cell array with responses

For each analysis, the cell array **resp** contains a structure the fields of which depend on the analysis method.

Direct frequency response analysis

method	**Integer**	Method: 1
type	**Integer**	Load type: 1 Motion 11 Aeroelastic results
nfreq	**Integer**	Number of excitation frequencies
freq(nfreq)	**Real**	List of excitation frequencies
kred(nfreq)	**Real**	List of reduced frequencies
G(:, nfreq)	**Complex**	Vortex strengths
nback	**Integer**	same as **nfreq**
freqback(nback)	**Real**	same as **freq**
U(ndofg, nfreq)	**Complex**	Displacement response matrix
panres	**Structure**	Structure with panel results (see page 52)

nback and **freqback** are included for compatibility with modal frequency response results.

Modal frequency response analysis

method	**Integer**	Method: 2
type	**Integer**	Load type 11 Aeroelastic results
nfreq	**Integer**	Number of excitation frequencies
freq(nfreq)	**Real**	List of excitation frequencies
kred(nfreq)	**Real**	List of reduced frequencies
G(:, nfreq)	**Complex**	Vortex strengths

Q(:, nfreq)	**Complex**	Response of modal coordinates
nback	**Integer**	Number of excitation frequencies of which physical displacements are available from back transformation
freqback(nback)	**Real**	Excitation frequencies of which physical displacements are available from back transformation
U(ndofg, nback)	**Complex**	Physical displacements from back transformation
panres	**Structure**	Structure with panel results (see page 52)

Force summation method

method	**Integer**	Method: 4
type	**Integer**	Load type: 11 Aeroelastic results
nfreq	**Integer**	Number of excitation frequencies
freq(nfreq)	**Integer**	List of excitation frequencies
kred(nfreq)	**Integer**	List of reduced frequencies
G(:, nfreq)	**Complex**	Vortex strengths
Q(:, nfreq)	**Complex**	Response of modal coordinates
Ue(:, nfreq)	**Complex**	Elastic deformation
nback	**Integer**	Number of excitation frequencies of which physical displacements are available from back transformation
freqback(nback)	**Real**	Excitation frequencies of which physical displacements are available from back transformation
U(ndofg, nback)	**Complex**	Physical displacements from back transformation
panres	**Structure**	Structure with panel results (see page 52)

Fields of structure **panres**

nofpan	**Integer**	Number of panels
P(3, nofpan)	**Real**	Coordinates of point in the middle between points A and B (cf. Figure 3.1-1)
f(3, nofpan, nfreq)	**Complex**	Panel force vectors
p(nofpan, nfreq)	**Complex**	Panel pressure

Field **panres** is added by function **mfs_resultsx**.

Remarks

1. Results from a modal frequency response analysis or from the force summation method are only available in an aeroelastic analysis.
2. Displacements are only available in an aeroelastic analysis.
3. Frequencies are only available in an aeroelastic analysis.

3.2.6 Structure modes

Created by: **mfs_trf_modes**, **mfs_freqrespc**

Fields:

```
modes.nofmod
     .rigmod
     .omega(nofmod)
     .freq(nofmod)
     .disp(:, nofmod)
```

nofmod	**Integer**	Number of normal modes
rigmod	**Integer**	Number of rigid body modes
omega(nofmod)	**Real**	Circular eigenfrequencies
freq(nofmod)	**Real**	Eigenfrequencies
disp(:, nofmod)	**Real**	Normal modes matrix

Remarks

1. These results are only available in an aeroelastic frequency response analysis.

2. Circular eigenfrequencies and eigenfrequencies are stored as column matrices.

4 Data Structures of Aeroelastic Components

Overview:

```
component.type
         .solid
         .aero
         .splines.nofspl
                 .ids(nofspl)
                 .types{nofspl}
                 .data{nofspl}
                 .Shg(nh, ng)
                 .Svh(nofpan, nh)
                 .Snh(3*nofnod, nh)
                 .D1h(nofpan, nh)
                 .D2h(nofpan, nh)
         .load.nofldc
              .infldc(nofldc)
              .gust(nofldc).qdyn
                           .v
                           .wg
                           .x0
              .mnvr.qdyn(nofldc)
                   .v(nofldc)
                   .D1K(nofpan, ncntrl)
                   .D2K(nofpan, ncntrl)
                   .UK(ncntrl, nofldc)
         .flutter.method
                 .nofmod
                 .modno(nofmod)
                 .nofpnt
                 .nsel
                 .kred(nofmod, nofpnt) | kred(nofpnt)
                 .v(nofmod, nofpnt) | v(nofpnt)
                 .g(nofmod, nofpnt)
                 .a(nofmod, nofpnt)
                 .f(nofmod, nofpnt)
                 .msel(nsel)
                 .XX(nsel, nofmod, nofpnt)
```

type	**String**	The model type defines the type of the mechanical field. The model type of an aeroelastic component is **"aeroelastic"**.
solid	**Structure**	Structure with data of the solid component (see chapter 2)

aero	**Structure**	Structure with data of the aerodynamic component (see chapter 3)
splines	**Structure**	Structure with spline data (see page 55)
load	**Structure**	Structure with load data (see page 56)
flutter	**Structure**	Results of flutter analysis (see page 57)

4.1 Model Description

4.1.1 Structure splines

Created by: **mfs_new_splines**

Modified by: **mfs_splinesx**

Fields:

```
splines.nofspl
       .ids(nofspl)
       .types{nofspl}
       .data{nofspl}
       .Shg(nh, ng)
       .Svh(nofpan, nh)
       .Snh(3*nofnod, nh)
       .D1h(nofpan, nh)
       .D2h(nofpan, nh)
```

nofspl	**Integer**	Number of splines
ids(nofspl)	**Integer**	Spline identifiers in ascending sort
types{nofspl}	**String**	Spline types: **"tb"** Torsion-bending-spline
data{nofspl}	**Cell array**	Spline data (see page 56)
Shg(nh, ng)	**Real**	Matrix relating spline coefficients to displacements of solid component
Svh(nofpan, nh)	**Real**	Matrix relating displacements at vortex points to spline coefficients
Snh(3*nofnod, nh)	**Real**	Matrix relating displacements at panel corner nodes to spline coefficients

D1h(nofpan, nh)	**Real**	Matrix relating normal wash at control points to spline coefficients
D2h(nofpan, nh)	**Real**	Matrix relating displacements at the control points to the spline coefficients

where

nh	Number of spline coefficients
ng	Number of global degrees of freedom of the solid component
nofpan	Number of panels
nofnod	Number of aerodynamic nodes in structure **nodes** (see page 40)

The matrices are added by functions depending on the spline type.

Spline data of torsion-bending splines

Spline type: **"tb"**

Function: **mfs_new_spline_tb**

For each spline, the cell array contains a structure with the following fields:

T(3, 3)	**Real**	Transformation matrix to the spline coordinate system
ixs(:)	**Integer**	Indices of the solid nodes involved in the spline
scoor(2, :)	**Real**	Coordinates of solid nodes in spline coordinate system
nbreaks	**Integer**	Number of spline breaks
breaks(nbreaks)	**Real**	Spline breaks
pid(:)	**Integer**	Indices of panels in structure **panel**

4.1.2 Structure load

Created by: **mfs_new_load**

Fields:

```
load.nofldc
    .infldc(nofldc)
    .gust(nofldc).qdyn
```

```
                .v
                .wg
                .x0
    mnvr.qdyn(nofldc)
        .v(nofldc)
        .D1K(nofpan, ncntrl)
        .D2K(nofpan, ncntrl)
        .UK(ncntrl, nofldc)
```

nofldc	**Integer**	Number of load cases
infldc(nofldc)	**Integer**	Load case information: Bit 1: Gust Bit 2: Manoeuvre
gust(nofldc)	**Structure**	Structure array with gust data
mnvr	**Structure**	Structure with manoeuvre data

Fields of structure **gust**

qdyn	**Real**	Dynamic pressure
v	**Real**	Flight velocity
wg	**Real**	Gust velocity
x0	**Real**	*x*-coordinate of reference point

Fields of structure **mnvr**

qdyn(nofldc)	**Real**	Dynamic pressure
v(nofldc)	**Real**	Flight velocity
D1K(nofpan, ncntrl)	**Real**	Downwash matrix $\left[D_K^1\right]$
D2K(nofpan, ncntrl)	**Real**	Downwash matrix $\left[D_K^2\right]$
UK(ncntrl, noflcd)	**Complex**	Matrix $\left[U^K\right]$ with complex controller amplitudes

4.2 Results

4.2.1 Structure flutter

Created by: **mfs_flutter_k, mfs_flutter_pk**

Fields:

```
flutter.method
       .nofmod
       .modno(nofmod)
       .nofpnt
       .nsel
       .msel(nsel)
       .kred(nofmod, nofpnt) | kred(nofval)
       .v(nofmod, nofpnt) | v(nofpnt)
       .g(nofmod, nofpnt)
       .a(nofmod, nofpnt)
       .f(nofmod, nofpnt)
       .XX(nsel, nofmod, nofpnt)
```

method	**String**	Flutter method: **"k"** k-method **"pk"** pk-method
nofmod	**Integer**	Number of flutter modes
modno(nofmod)	**Integer**	Array with flutter mode numbers
nofpnt	**Integer**	Number of flutter points: k-method: Number of reduced frequencies pk-method: Number of velocities
nsel	**Integer**	Number of structural normal modes used in the modal reduction
msel(nsel)	**Integer**	Indices of normal modes used in modal reduction
kred(nofmod, nofpnt)	**Real**	Array with reduced frequencies (pk-method)
kred(nofpnt)	**Real**	Array with reduced frequencies (k-method)
v(nofmod, nofpnt)	**Real**	Array with velocities (k-method)
v(nofpnt)	**Real**	Array with velocities (pk-method)
g(nofmod, nofpnt)	**Real**	Array with loss factors (k-method)
a(nofmod, nofpnt)	**Real**	Array with real parts of eigenvalues (pk-method)
f(nofmod, nofpnt)	**Real**	Array with frequencies

XX(nsel, nofmod, nofpnt)	**Complex**	Array with flutter mode shapes with respect to modal basis

4.2.2 Aerodynamic and structural results

The aerodynamic results of an aeroelastic analysis are stored in the aerodynamic component and the structural results in the solid component.

	Solid component	Aerodynamic component
statresp	see section 2.3.1	see section 3.2.1
freqresp	see section 2.3.3	see section 3.2.5
trim	see section 2.3.5	see section 3.2.2
diverg	see section 2.3.6	see section 3.2.3
flutter	see section 2.3.7	see section 3.2.4