

User Manual

Mefisto 2.7

Volume 3: Aeroelasticity

Contents

1	Introduction.....	2
2	Model Description.....	3
2.1	Description of the Input Data.....	3
2.1.2	Splines.....	4
2.1.3	Loads.....	5
3	Functions.....	7
3.1	Analysis.....	7
3.2	Output.....	13
3.3	Utility functions.....	14
4	List of Functions.....	17

1 Introduction

This volume of the manual describes how to use Mefisto to solve problems in aeroelasticity. If you are new to Mefisto, please read first the Getting Started Manual.

Currently, the following features are supported:

Solid Mechanics:	Finite Element Method
Aerodynamics:	Vortex-Lattice Method
Splines:	Torsion-bending splines
Analysis:	Static aeroelastic analysis Restrained and unrestrained trim analysis Flutter analysis Frequency response analysis

To perform an aeroelastic analysis, you need be familiar with solid mechanics and aerodynamics. Volume 1 of the Mefisto User Manual describes how to solve problems in solid mechanics, and Volume 2 is on aerodynamics.

2 Model Description

The model is described by a GNU Octave structure that is input to function **mfs_new**.

In the following, *variables* written in italics can have any name or value. The names of **variables** not written in italics must not be changed.

In an aeroelastic analysis three different components are needed, namely a solid component, an aerodynamic component and an aeroelastic component.

The data needed to define a solid component are described in Volume 1 of the Mefisto User Manual and those needed to define an aerodynamic component are described in Volume 2. This manual describes only the data needed to define the aeroelastic component.

2.1 Description of the Input Data

The model is defined by the following GNU Octave structure which is defined by the user:

```
model.type
    .solid
    .aero
    .splines(:).id
                .type
                .lsid
                .nodes
                .data
    .loads.gust(:).wg
                .v
                .qdyn
                .x0
                .lc
    .manoeuvre(:).qdyn
                .v
                .cntname(2)
                .lc
```

type	String	For an aeroelastic component, the model type is " aeroelastic ".
solid	Structure	Structure with the data of the solid component (see User Manual, Vol. 1)
aero	Structure	Structure with the data of the aerodynamic component (see User Manual,

Vol. 2)

splines (:)	Structure	Structure array with the spline data (see page 4)
loads	Structure	Structure with the load data (see page 5)

Remarks

1. The solid component must contain the following data:
 - a) the model description as created by function **mfs_new**
 - b) the stiffness matrix
 - c) the mass matrix (in case of a trim analysis, a flutter analysis or a frequency response analysis)
 - d) the normal modes (in case of a flutter analysis or a modal frequency response analysis)
2. The aerodynamic component must contain the following data:
 - a) the model description as created by function **mfs_new**

2.1.2 Splines

```
splines (:).id
           .type
           .lsid(:)
           .nodes(:)
           .data
```

id	Integer	Spline identifier
type	String	Spline type (see page 5)
lsid(:)	Integer	Identifiers of lifting surfaces
nodes(:)	Integer	List of solid nodes involved in spline (optional; default: all solid nodes)
nodes	String	Name of a nodal point set defining the nodes involved in the spline (optional)
data	Structure	Spline data (see page 5)

Remarks

1. Spline identifiers are arbitrary positive integer numbers. They have to be unique.
2. The spline data depend on the spline type.
3. Lifting surfaces connected to the same spline must have parallel normal vectors.
4. The nodal points involved in the spline can be defined either by a list of nodal point identifiers or by the name of a nodal point set. The definition is optional. By default, all nodal points of the solid are involved in the spline.
5. Different splines should only be used if the normal vectors of the lifting surfaces are not parallel or if the lifting surfaces are connected to different sets of solid nodes.

Spline types and spline data

Currently, the following splines are available:

"tb" Torsion-Bending Spline

The fields of structure **data** depend on the spline type:

Type	Fields		
"tb"	nbreaks	Integer	Number of spline breaks
	breaks (:)	Real	Array with spline breaks

Remarks

1. Either the number of spline breaks or the breaks have to be defined.

2.1.3 Loads

The structure contains one structure array for each load type.

```
loads.gust(:).wg
               .v
               .qdyn
               .x0
               .lc
      .manoeuvre(:).qdyn
                      .v
                      .cntname(2)
                      .lc
```

gust (:)	Structure	Gust loads
manoeuvre (:)	Structure	Manoeuvre loads

Gust loads

wg	Real	Gust velocity
v	Real	Flight velocity
qdyn	Real	Dynamic pressure
x0	Real	x-coordinate of origin of aircraft system (optional; default: 0)
lc	Integer	Load case number (optional; default: 1)

Manoeuvre loads

qdyn	Real	Dynamic pressure
v	Real	Flight velocity
cntname (2)	Real	Amplitude and phase of the controller angle (both in degrees)
lc	Integer	Load case number (optional; default: 1)

Remarks

1. Gusts and manoeuvres cannot be used in the same load case.
2. Load case numbers are positive integer numbers. There should be no gaps between the load case numbers.
3. **cntname** is the name of a controller.
4. The phase of the controller angle is optional. The default is 0°.

3 Functions

In GNU Octave, a list of all functions can be obtained by typing

```
help mefisto
```

or just

```
mefisto
```

Information on a specific function is obtained by typing

```
help function
```

where *function* has to be replaced by the name of the specific function.

3.1 Analysis

```
cmp = mfs_new(fid, model, opts)
```

fid	File Handle	File handle of the output file
model	Structure	Structure with the model description (see Section 2.1)
opts	Structure	Structure with options (optional)
cmp	Structure	Structure with the component data (0 if errors occurred)

The function generates a component from the model description. Error messages and information on the component generated are written to the output file.

List of options

Name	Value	Default	Description
ptol	Real	1e-4	Tolerance to test if vectors are parallel

Remarks

1. Vectors are parallel if the condition $1 - \cos(\alpha) < \mathbf{ptol}$ is satisfied where α is the angle between the vectors.

```
cmp = mfs_splines(cmp)
```

cmp	Structure	Structure with data of aeroelastic component
------------	------------------	--

The function computes the spline matrices.

```
[cmps, cmpa] = mfs_statresp(cmp, opts)
```

cmp	Structure	Structure with data of aeroelastic component
opts	Structure	Structure with options (optional)
cmps	Structure	Structure with data of solid component
cmpa	Structure	Structure with data of aerodynamic component

The function computes the primary results of a steady aeroelastic analysis.

The results are stored in the solid and the aerodynamic component:

cmps	Displacements, reaction loads
cmpa	Vortex strengths, displacements

List of options

Name	Value	Default	Description
method	String	"direct"	Solution method: "direct" direct solution (default) "bicg" biconjugate gradient method
tol	Real	1e-6	Tolerance for the biconjugate gradient method
maxit	Integer	20	Maximum number of iterations for the biconjugate gradient method

Remarks

1. The direct solution method uses the Frobenius-Schur-Woodbury identity, thus avoiding the factorization of a large dense matrix. It is almost as efficient as the iterative biconjugate gradient method.

```
[cmps, cmpa, nfound] = mfs_diverg(cmp, nvec)
```

cmp	Structure	Structure with data of aeroelastic component
nvec	Integer	Number of vectors to compute
cmps	Structure	Structure with data of solid component
cmpa	Structure	Structure with data of aerodynamic component
nfound	Integer	Number of positive real eigenvalues found (optional)

The function computes **nvec** eigenvectors of the divergence eigenvalue problem. For each positive eigenvalue, the dynamic pressure and the deformation are computed.

The results are stored in the solid and the aerodynamic component:

cmps Dynamic pressure at divergence, displacements
cmpa Dynamic pressure at divergence, displacements

```
[cmps, cmpa] = mfs_trim(cmp, method)
```

cmp **Structure** Structure with data of aeroelastic component
method **String** Method (optional):
 "restrained" restrained trim analysis
 "unrestrained" unrestrained trim analysis (default)
cmps **Structure** Structure with data of solid component
cmpa **Structure** Structure with data of aerodynamic component

The function performs an aeroelastic trim analysis of a flexible aircraft.

The results are stored in the solid and the aerodynamic component:

cmps Displacements
cmpa Trim parameters, vortex strengths, displacements

```
cmp = mfs_flutter(cmp, kred, rho, "k", opts)
cmp = mfs_flutter(cmp, v, rho, "pk", opts)
cmp = mfs_flutter(cmp, v, rho, "pk", pairs)
cmp = mfs_flutter(cmp, v, rho, "pk", pairs, opts)
```

cmp **String** Structure with data of aeroelastic component
kred(:) **Real** Array with reduced frequencies
v(:) **Real** Array with velocities
rho **Real** Mass density of the air
pairs(:) **Integer** Array with flutter mode pair numbers (optional)
opts **Structure** Structure with options (optional)

The function performs a flutter analysis. The following results are computed and stored in the aeroelastic component:

- Reduced frequencies
- Velocities
- Loss factors (k-method) or real part of eigenvalues (pk-method)

- Frequencies
- Modal coordinates of flutter mode shapes

In the pk-method, modes appear as complex conjugate pairs. If the array **pairs** is defined, iterations on the reduced frequency are performed only for the pairs defined.

List of options

Name	Value	Default	Description
msg	File Handle	0	File handle of message file (if 0, no messages are printed)
nx	Integer	10	Number of intervals per reference chord length in regular wake grid
m1	Integer	4	Factor to divide minimum control point distance from trailing edge to get size of first interval of graded wake grid
lenw	Real	20	Length of wake behind last control point in multiples of reference chord
mtol	Real	100	Tolerance to identify normal modes that do not participate in flutter (in multiples of eps)
ktol	Real	1e-4	Tolerance for reduced frequencies
mxiter	Integer	10	Maximum number of iterations on reduced frequency

Remarks

1. **ktol** and **mxiter** are used in the pk-method only.

Examples

```
rho = 1.21e-12;           % Mass density of air
kred = 0.5 : 0.5 : 5.0;   % Reduced frequencies
wing = mfs_flutter(wing, kred, rho, "k");
```

performs a flutter analysis using the k-method. The eigenvalue problem is solved for all reduced frequencies in the list.

```
rho = 1.21e-12;           % Mass density of air
v = 1000 * (20 : 10 : 70); % Velocities (in mm/s)
pairs = 1 : 5;            % Flutter mode pairs
wing = mfs_flutter(wing, v, rho, "pk", pairs);
```

performs a flutter analysis using the pk-method. Iterations on the reduced frequency are performed for the first 5 mode pairs only.

```
[cmps, cmpa] = mfs_back(cmp, class, item, arg1, arg2, ...)
```

cmp	Structure	Structure with data of aeroelastic component
class	String	Class
item	String	Item
arg1, arg2, ...		Additional input arguments
cmps	Structure	Structure with data of solid component
cmpa	Structure	Structure with data of aerodynamic component

The function performs the backtransformation from modal to physical coordinates. The results are stored in the solid and aerodynamic component.

List of items

class = "flutter": Flutter analysis

Item	Description
"disp"	Flutter shape
arg1 = modpnt(:, 2)	Pairs of flutter mode numbers and flutter point numbers

Example

```
modpnt = [3, 10]; % Flutter mode, flutter point
[wings, winga] = mfs_back(wing, "flutter", "disp", modpnt);
```

performs the backtransformation of flutter point 10 of the 3rd flutter mode.

```
cmp = mfs_freqresp(cmp, f, parameter, value, ...)
cmp = mfs_freqresp(cmp, f, opts, parameter, value, ...)
```

cmp	Structure	Structure with data of aeroelastic component
f	Real	List of excitation frequencies
opts	Structure	Structure with options (optional)

The function performs a frequency response analysis. The results are stored in the solid and aerodynamic subcomponents of the aeroelastic component. For postprocessing, the solid and aerodynamic subcomponents can be ex-

tracted from the aeroelastic component (see function **mfs_extract**).

Parameters

Name	Value	Default	Description
"method"	String	"fsm"	Method: "fsm" Force summation method "modal" Modal frequency response "direct" Direct frequency response
"nband"	Integer	0	Number of additional excitation frequencies per halfpower bandwidth
"loadcase"	Integer	1	Load case number

List of options

Name	Value	Default	Description
msg	File Handle	0	File handle of message file (if 0, no messages are printed)
nx	Integer	10	Number of intervals per reference chord length in regular wake grid
m1	Integer	4	Factor to divide minimum control point distance from trailing edge to get size of first interval of graded wake grid
lenw	Real	20	Length of wake behind last control point in multiples of reference chord
ktol	Real	1e-4	Zero threshold for reduced frequencies

Examples

```
glider = mfs_freqresp(glider, f, "loadcase", 2);
```

performs a modal frequency response analysis with the force summation method using the loads defined in load case 2

```
glider = mfs_freqresp(glider, f, "method", "direct")
```

performs a direct frequency response analysis using the loads defined in load case 1

```
cplx = mfs_extract(cmp, subcmp)
```

cmp	Structure	Structure with data of aeroelastic component
subcmp	String	Name of component to extract: "solid" Solid component "aero" Aerodynamic component
cmpx	Structure	Structure with data of extracted component

The function extracts the selected component from the aeroelastic component.

3.2 Output

```
mfs_print(fid, cmp, class, item1, ...)
```

fid	File Handle	File handle of the output file
cmp	Structure	Structure with data of aeroelastic component
class	String	Class
item1, ...	String	Items

The function writes the selected result items to the output file.

List of items

class = "flutter": Results of a flutter analysis

Item	Description
"curves"	For each flutter mode shape, the following data are output in tabular form: k-method: reduced frequency, velocity, loss factor and frequency pk-method: velocity, reduced frequency, real part of eigenvalue, frequency

```
[out1, out2, ... ] =  
  mfs_getresp(cmp, class, item, arg1, arg2, ...)
```

cmp	Structure	Structure with the component data
class	String	Response class
item	String	Response item
arg1, arg2, ...		Additional input arguments
out1, out2, ...		Output arguments

The function returns the selected responses. The responses must have been computed before they can be requested.

The meaning of the additional input arguments and of the output arguments depends on the class and the item.

List of items

class = "flutter": Results of a flutter analysis

Item	Description
"curves"	Flutter curves
arg1 = modno(:)	Flutter mode numbers (optional; default is all flutter modes)
out1 = v(:, :)	Velocities: k-method: columns correspond to flutter modes
out2 = ag(:, :)	pk-method: one column only k-method: loss factors pk-method: real parts of eigenvalues
out3 = k(:, :)	columns correspond to flutter modes Reduced frequencies: k-method: one column only pk-method: columns correspond to flutter modes
out4 = f(:, :)	Frequencies: columns correspond to flutter modes

3.3 Utility functions

```
cmpt = mfs_transfer(cmp, cmpi, class, item)
```

cmp	String	Structure with data of aeroelastic component
cmpi	Structure	Structure with data of input component
class	String	Class of data to be transferred
item	String	Item to be transferred

cmpt **Structure** Structure with data of component the data are transferred to

The function transfers data from the solid to the aerodynamic component or vice versa. This function may be useful to test the splines.

Transfer from the aerodynamic to the solid component

class = "statresp": Steady aerodynamic results

Item	Description
"loads"	Structural loads from aerodynamic results

class = "trim": Trim analysis results

Item	Description
"loads"	Structural loads from aerodynamic results

Transfer from the solid to the aerodynamic component

class = "statresp": Static results

Item	Description
"disp"	Displacements

class = "modes": Normal modes

Item	Description
"disp"	Displacements

mfs_merge(fnin1, fnin2, fnout, format)

fnin1	String	Name of first input file
fnin2	String	Name of second input file
fnout	String	Name of output file
format	String	File format of output file

The function can be used to merge two files into one file. The following file formats are supported:

- **"msh":** Gmsh MSH ASCII file Version 4.1
- **"msh41":** Gmsh MSH ASCII file Version 4.1
- **"msh22":** Gmsh MSH ASCII file Version 2.2

The function can be used to merge the solid and the aerodynamic mesh and to merge the displacements of the solid and the aerodynamic mesh. This may be useful when checking the splines.

Remarks

1. If a Gmsh Version 4.1 output file is requested, both input files must be Gmsh Version 4.1 files.
2. If a Gmsh Version 2.2 output file is requested, the input files may be Gmsh Version 2.2 or Gmsh Version 4.1 files. The versions can also be mixed.

4 List of Functions

mfs_back.....	11	mfs_new.....	7
mfs_diverg.....	8	mfs_print.....	13
mfs_extract.....	12	mfs_splines.....	7
mfs_flutter.....	9	mfs_statresp.....	8
mfs_freqresp.....	11	mfs_transfer.....	14
mfs_getresp.....	13	mfs_trim.....	9
mfs_merge.....	15		