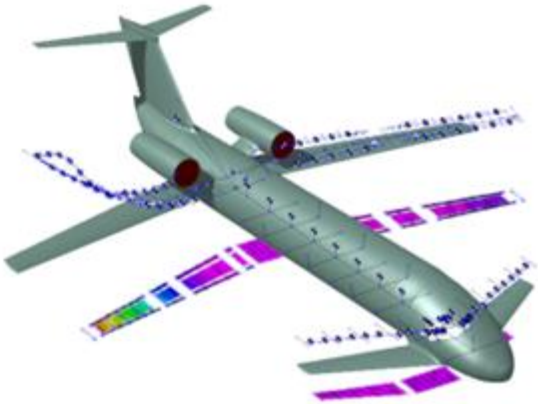
Aeroelastic analysis using NeoCASS and NeoRESP



18 July 2018

NeoCASS 2.2.82

Aeroelastic analysis in NeoCASS/NeoRESP

In NeoCASS are included capabilities for aeroelastic analysis:

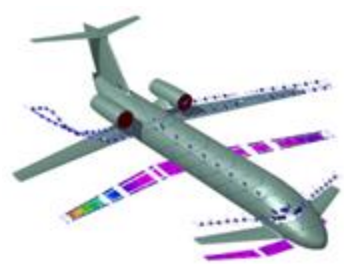
- Static aeroelasticity (trim and loads)
- Normal modes
- Flutter analysis

NeoRESP is an additional module used for dynamic response analysis:

- Gust response;
- Control surface deflection response;
- Response to external loads;
- State-space realization of the aeroelastic system.

It features

- Frequency domain solution;
- Time domain solution;
- Mode acceleration for load recovery.



Tutorial overview

This tutorial will introduce some of the capabilities of the code, following the steps:

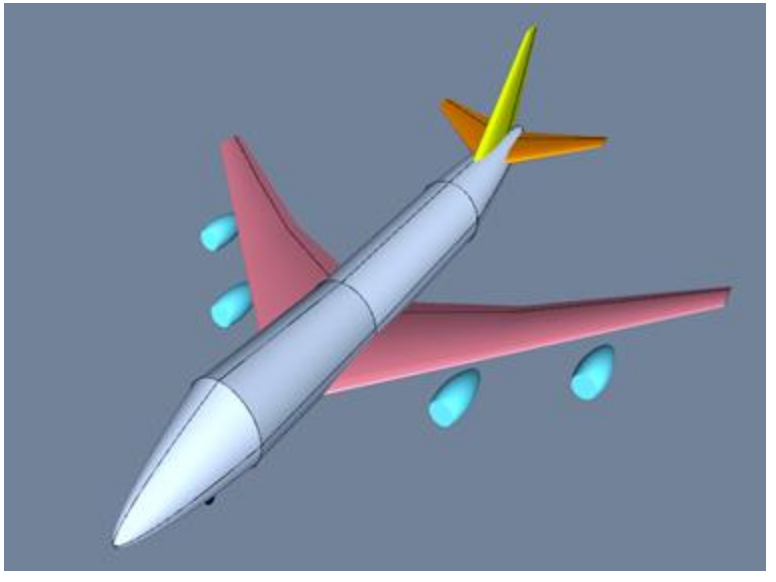
- Generation of the aircraft model using GUESS
- Linear trim analysis
- Normal mode computation
- Flutter analysis
- Gust response analysis
- Control surface deflection time response
- State-space model generation

All the procedures will be presented on a model inspired to the B747-100 aircraft. The model data, and the scripts that can be used to reproduce the results presented here can be found in the directory

```
<NeoCASS_base>/Examples/B747-100/
```

Model generation (1)

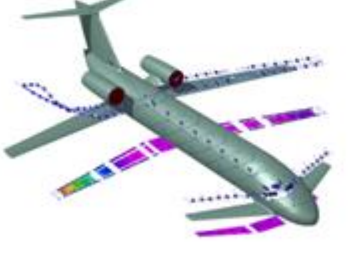
The aircraft model in Smartcad format is generated starting from an .xml file with the geometric properties and using GUESS.



The files used in this operation are

- `B747-100.xml` : geometry input file which can be defined/edited using AcBuilder
- `B747-100_CAS25.inc` : maneuver definition for structural sizing

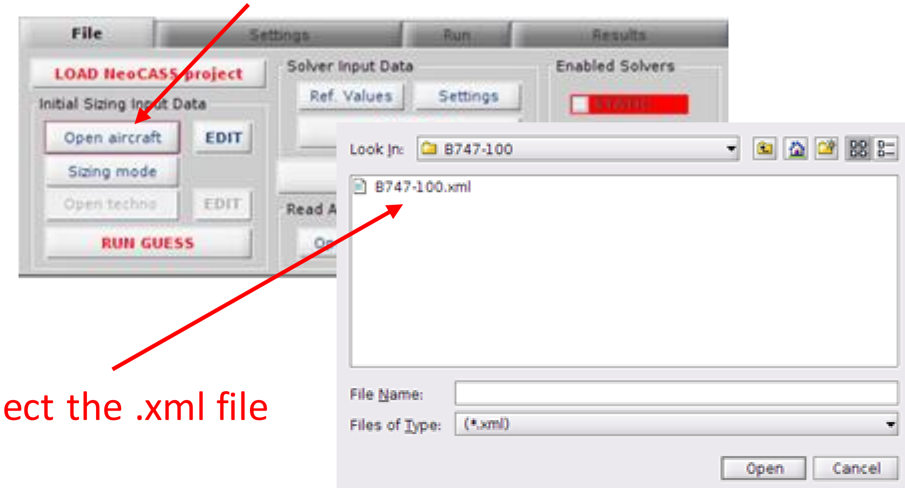
The operation can be performed either using the NeoCASS GUI (`NeoCASS` command) or programmatically (the commands can be found in the script `exampleMain_guess.m`).



Model generation (2)

GUI

Select Open aircraft button



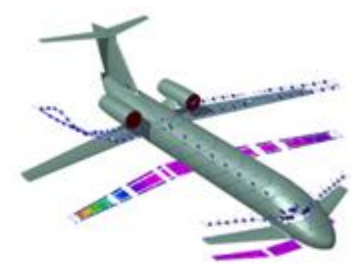
Select the .xml file

Select Sizing mode button

Command line

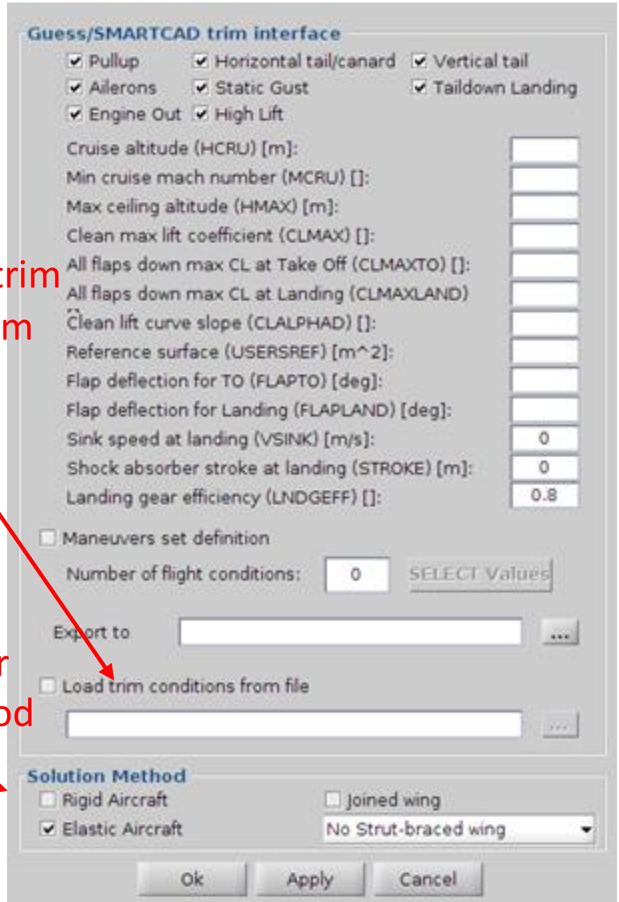
Define the .xml files

```
filename_geo = 'B747-100.xml';
filename_tech = 'B747-100.xml';
```



Model generation (3)

GUI



Select "Load trim conditions from file" option

Leave default parameters for solution method

Select the file B747-100_CAS25.inc

Command line

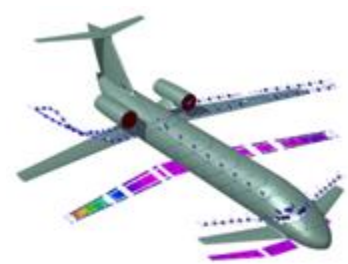
Define the .inc file for maneuver definition

```
filename_trim = 'B747-100_CAS25.inc';
```

Define the solution method

```
model.Joined_wing = false;
model.Strut_wing = false;
model.EnvPoints = [];
model.guessstd = false;
```

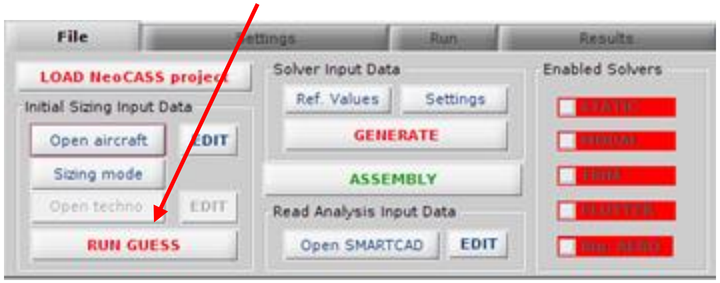
True : Rigid aircraft
False : elastic aircraft



Model generation (4)

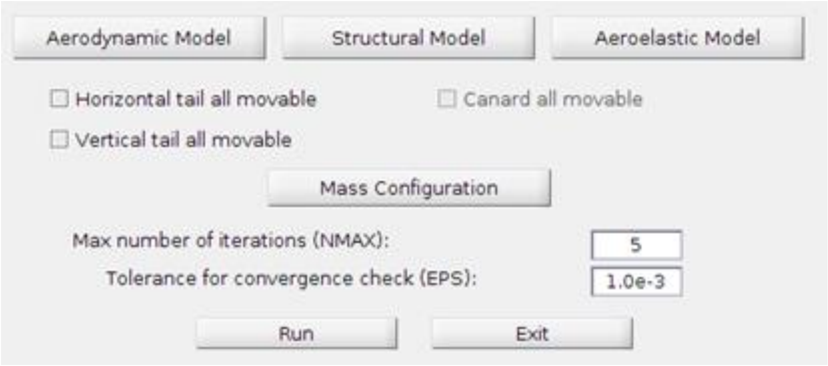
GUI

Select RUN GUESS button



Specify the name of the generated files (e.g. B747-100_v00.inc)

The ChEcK GUI will appear



Command line

Determine whether to use or not the ChEcK GUI (`true`: the ChEcK GUI will appear)

```
sizinginBatch = false;
```

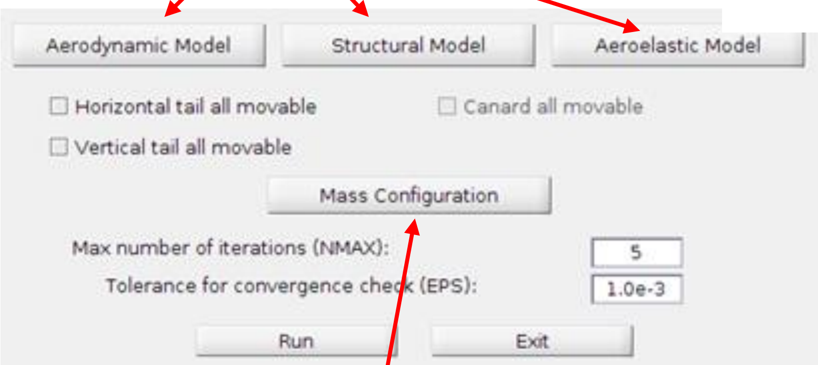
Define the name of the generated model file

```
filename_stick = 'B747-100_v00.inc';
```

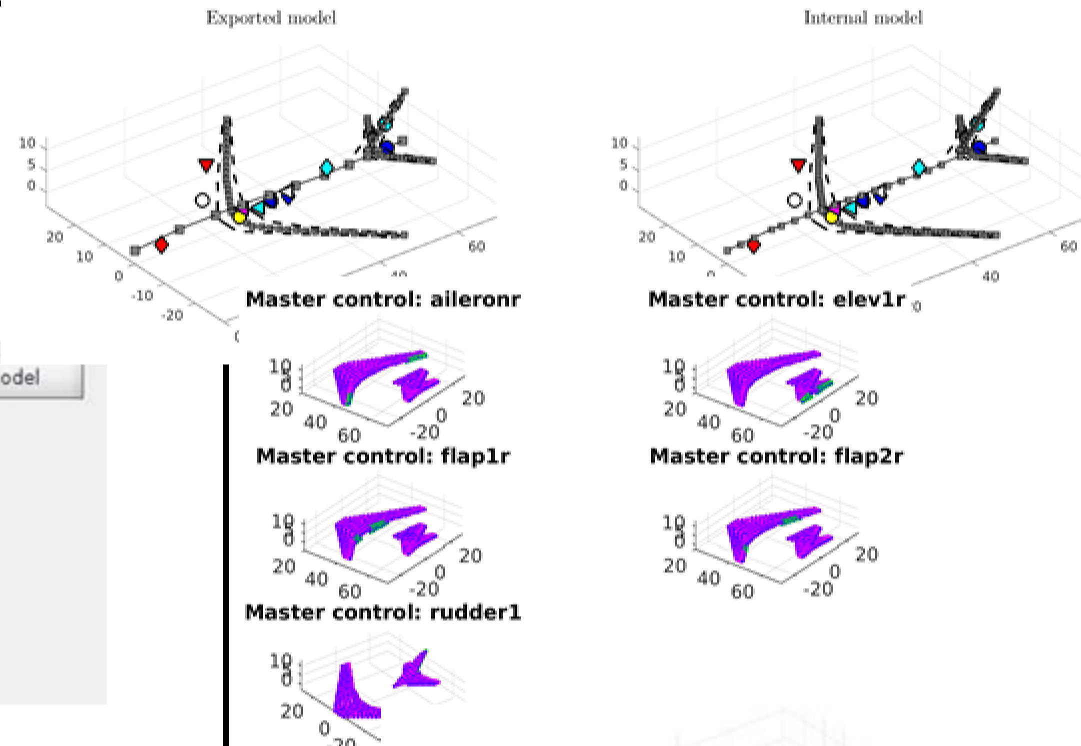
Model generation (5)

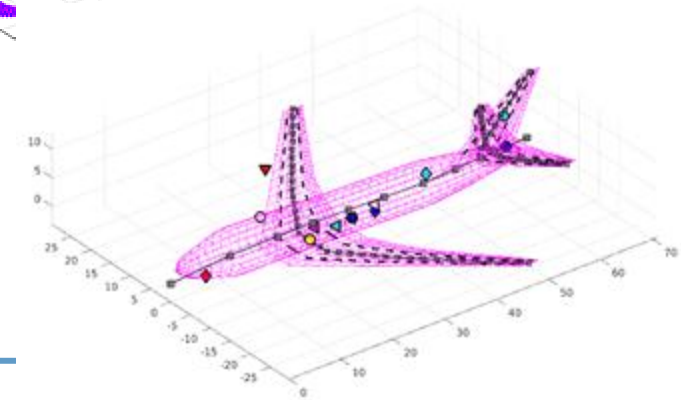
GUI

These buttons allows the visualization of the model



Run Mass Configuration





Model generation (6)

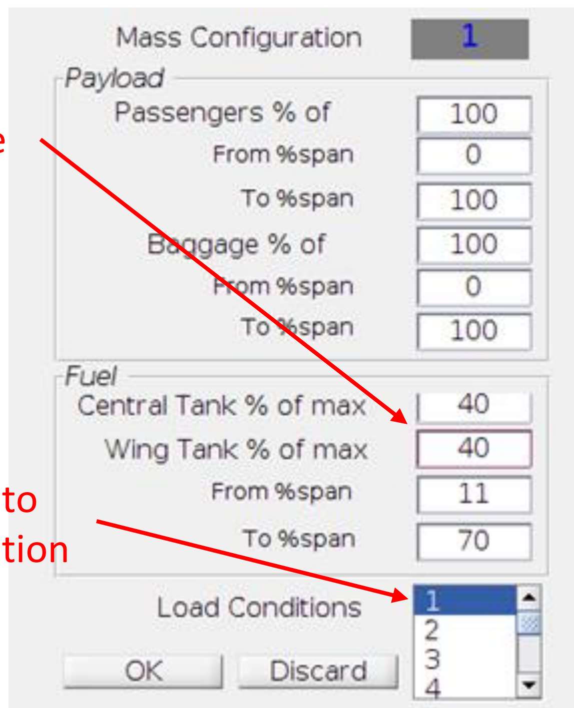
GUI

Specify 1 in the numer of configurations defined

Press ''Select values''

Define fuel quantity (set 40% of maximum value in wing and cental tank)

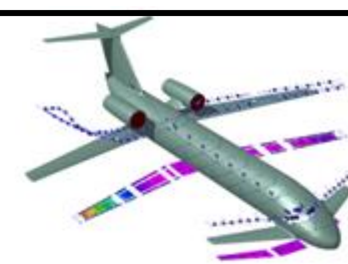
Select all the trim conditions related to the mass configuration (select all)



Command line

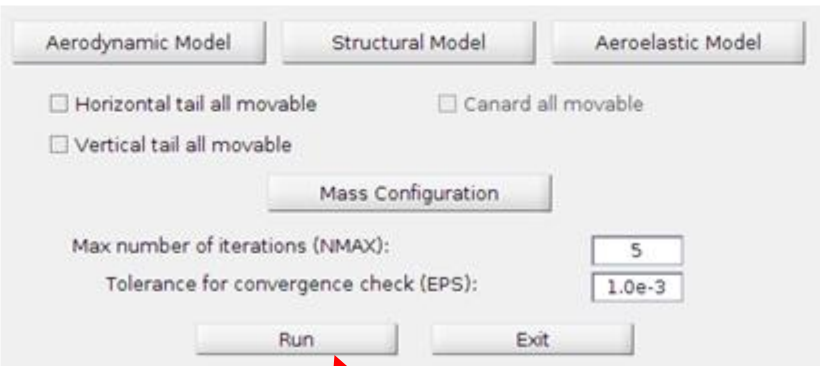
Define mass configurations (all fields are arrays for multiple mass configurations)

```
MassConf.Pass = 1;
MassConf.Baggage = 1;
MassConf.Cfuel = 0.4;
MassConf.Wfuel = 0.4;
MassConf.WfuelStart = 0;
MassConf.WfuelArrive = []; % default from xml
MassConf.WfuelStart = []; % default from xml
MassConf.BagArrive = 1;
MassConf.BagStart = 0;
MassConf.PaxArrive = 1;
MassConf.PaxStart = 0;
MassConf.Load = []; % all maneuvers
```



Model generation (7)

GUI



Run GUESS

Command line

Set options (otherwise defaults values are used)

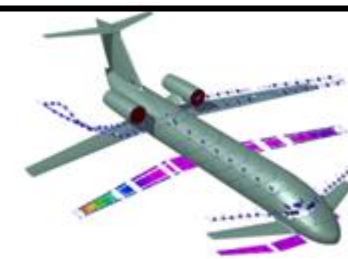
```
global NMAX EPS
NMAX = 5;
EPS = 0.001;
```

Initialize data structures

```
init_gui_params('neocass_gui_param.mat');
init_guess_model;
```

Run GUESS

```
guess_model = guess(filename_geo,
filename_tech, filename_stick, filename_trim,
model, inBatch, MassConf);
```



Model generation (7)

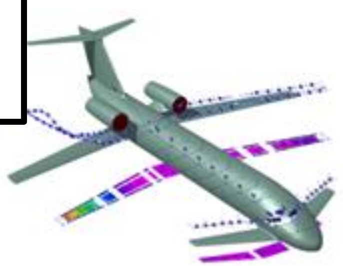
Once the optimization is done, these files are generated

- `B747-100_v00.inc` : model data in Smartcad format

```
$
$ Summary for total structural masses (CT included except for tailbooms):
$ Wing: 78415.9373 Kg
$ Htail: 4472.9978 Kg
$ Vtail: 3164.1663 Kg
$ Fuselage: 27537.1356 Kg
$ Canard: 0 Kg
$ Tailbooms: 0 Kg
$
$ Summary for secondary masses along fuselage (as PBAR density):
$ Systems: 44985.7794 Kg
$ Interior:  22000 Kg
$ Pilots: 255 Kg
$ Crew: 750 Kg
$ Paint: 317.7345 Kg
```

- `B747-100_v00CONM_CONF1.inc` : masses for the defined configuration

```
$ Mass Configuration 1
$ Baggage: 100% (4099.64 Kg)
$ Passengers: 100% (41004.988 Kg)
$ Wing Fuel: 40% (46000 Kg)
$ Central Fuel: 40% (18400 Kg)
```



Import the model in Matlab (1)

The smartcad model can be imported in matlab, it is convenient to define a complete aeroelastic solver in order to enable all the portions of the code.

Here a static aeroelastic solution is used:

```
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
$ DATA ADDED AFTER GUESS SIZING
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
$ ADD PAYLOAD
INCLUDE B747-100_v00CONM_CONF1.inc
$-------2-------3-------4-------5-------6-------7-------8-------9-------10$
SET TRIM SOLVER
SOL 144
AEROS           0       10.483  59.64   551     0       0
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
$ ADD MANEUVERS
INCLUDE B747-100_CAS25.inc
$
$ INCLUDE MODEL
INCLUDE B747-100_v00.inc
$
```

Import the model in Matlab (2)

The load_nastran_model command can be used to generate the beam_model data structure

```
global beam_model
filename_sma = 'inputMain_trim.dat';
beam_model = load_nastran_model(filename_sma);
plotNeoModel(beam_model);
```

Import the model in Matlab (3)

```
>> beam_model
beam_model =
          Info: [1x1 struct]
         Param: [1x1 struct]
         Coord: [1x1 struct]
          Node: [1x1 struct]
           Mat: [1x1 struct]
           Bar: [1x1 struct]
          PBar: [1x1 struct]
          Beam: [1x1 struct]
         PBeam: [1x1 struct]
             F: [1x1 struct]
             M: [1x1 struct]
         F_FLW: [1x1 struct]
          ConM: [1x1 struct]
            WB: [1x1 struct]
           SPC: [1x1 struct]
          Aero: [1x1 struct]
         Optim: [1x1 struct]
         Celas: [1x1 struct]
          RBE2: [1x1 struct]
          Gust: [1x1 struct]
           SET: [1x1 struct]
       Surfdef: [1x1 struct]
      Dextload: [1x1 struct]
          Damp: [1x1 struct]
        DesOPT: [1x1 struct]
           Res: []
```

The beam_model structure contains all the data defining

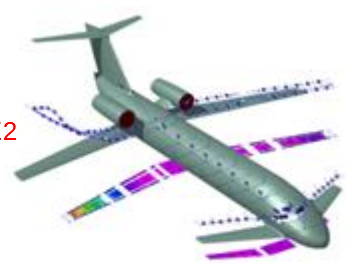
- The structural model
- The aerodynamic model
- The solver and the solver parameters
- (after running a NeoCASS solver) the solver results

Import the model in Matlab (4)

```
>> beam_model.Bar
          ID: [1x82 double]        Bar element IDs
         PID: [1x82 double]
        Conn: [82x3 double]        Connectivity: position in Node database of the 3 bar nodes
      Orient: [82x3 double]
     OffsetT: [1x82 double]
      Offset: [82x9 double]
      Colloc: [2x3x82 double]      Collocation points: coordinates of the points for load recoovery
           R: [4-D double]         Local orientation matrix (5 points: node1, colloc1, node2, colloc2, node3)
           D: [4-D double]
           M: [4-D double]
       barg0: [1x82 double]

>> beam_model.Node
        ID: [1x512 double]         Node ID list
        CS: [1x512 double]
     Coord: [512x3 double]         Node coordinates
        CD: [1x512 double]
     Index: [1x512 int32]
         R: [3x3x512 double]       Node orientation matrix
       DOF: [512x6 int32]          Table indicating for each nodal DOF the position in the global DOF vector
      Aero: [1x1 struct]           Aeronodes associated to each structural node
      DOF2: [512x6 int32]          As Node.DOF, but also RBE2 are considered in the definition of the global DOF
```

`beam_model.Node.DOF` : a zero value means that the corresponding DOF is constrained by a RBE0 or a SPC

`beam_model.Node.DOF2` : a zero value means that the corresponding DOF is constrained by a RBE0, an SPC or a RBE2



Import the model in Matlab (5)

```
>> beam_model.Aero
              ID: [200 250 201 251 202 252 204 254 400 450 401 451 402 452 301 302 300]
              CP: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
              IS: [1x17 struct]
             INT: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
          AESURF: [1x1 struct]
             geo: [1x1 struct]
           state: [1x1 struct]
             ref: [1x1 struct]
         lattice: []
    lattice_defo: []
     lattice_vlm: [1x1 struct]
     lattice_dlm: []
          Interp: [1x1 struct]
             Set: [1x1 struct]
            Trim: [1x1 struct]
            body: [1x1 struct]
```

Interpolation SETs

Geometry of lifting surfaces
(with control surface definition)

Reference dimensions

Aerodynamic mesh for
VLM computation

Spline definition

Trim cases

Aerodynamic mesh for
interference bodies

Export to NASTRAN model

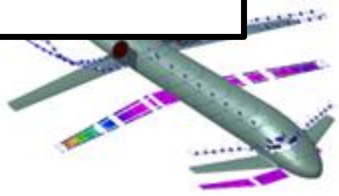
It is possible to export the model in NASTRAN format

- Only the model is exported, not the solver parameters
- It is necessary to define a complete aeroelastic solution input file for exporting the model

```
>> neo2nastran(inputFileName, outputFileName)
```

For example, using the input file with the static aeroelastic load definition:

```
filename_sma = 'inputMain_trim.dat';

neo2nastran(filename_sma, 'B747-100_v00_Nastran.bdf')
```



Aeroelastic static analysis (1)

NeoCASS can perform static trim and maneuver analysis

- Solution performed in mean axes
- Vortex Lattice Panel method used to compute aerodynamic forces

$$\begin{bmatrix} M_{rr} & M_{rd} \\ M_{dr} & M_{dd} \end{bmatrix} \begin{bmatrix} \ddot{u}_r \\ \ddot{u}_d \end{bmatrix} + \begin{bmatrix} \bar{K}_{rr} & \bar{K}_{rd} \\ \bar{K}_{dr} & \bar{K}_{dd} \end{bmatrix} \begin{bmatrix} u_r \\ u_d \end{bmatrix} = \begin{bmatrix} f_d \\ f_r \end{bmatrix} + q_\infty \begin{bmatrix} K_d^a \\ K_r^a \end{bmatrix} x_a$$

  - u_r : rigid motion
  - u_d : structural deformations
  - M: mass matrix
  - K: Stiffness matrix (structural and aerodynamic)
  - f : external forces
  - x_a : aircraft attitude (α, β, p, q, r)

- Computes:
  - Aerodynamic coefficients (corrected)
  - Trim solution
  - Structural deformations
  - Aerodynamic loads
  - Internal forces

Aeroelastic static analysis (2)

- The stiffness matrix already includes aerodynamic contribution

$$\bar{\boldsymbol{K}}_{zz} = \boldsymbol{K}_{zz} - q_\infty \boldsymbol{K}^a_{zz}$$

- Inertial forces from deformable motion neglected, rigid displacement assumed null (fixed axes/mean axes)

$$\begin{bmatrix} \boldsymbol{M}_{dt} \\ \boldsymbol{M}_{tt} \end{bmatrix} \ddot{\boldsymbol{u}}_t + \begin{bmatrix} \bar{\boldsymbol{K}}_{dd} & \bar{\boldsymbol{K}}_{dt} \\ \bar{\boldsymbol{K}}_{td} & \bar{\boldsymbol{K}}_{tt} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_d \\ \boldsymbol{u}_t \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{F}}_d \\ \bar{\boldsymbol{F}}_t \end{bmatrix} + q_\infty \begin{bmatrix} \boldsymbol{K}^a_{du} \\ \boldsymbol{K}^a_{tu} \end{bmatrix} \boldsymbol{v}_b + q_\infty \begin{bmatrix} \boldsymbol{K}^a_{dc} \\ \boldsymbol{K}^a_{tc} \end{bmatrix} \boldsymbol{\delta}_c$$

The trim equations including aeroelastic effects are then solved as

$$\tilde{\boldsymbol{M}}_{tt} \ddot{\boldsymbol{u}}_t = \tilde{\boldsymbol{F}}^{ext}_t + \tilde{\boldsymbol{F}}^a_{0t} + q_\infty \tilde{\boldsymbol{K}}^a_{tu} \boldsymbol{v}_b + q_\infty \tilde{\boldsymbol{K}}^a_{tc} \boldsymbol{\delta}_c$$

Aeroelastic static analysis (3)

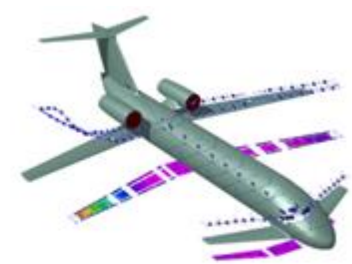
$$\tilde{\boldsymbol{M}}_{tt}\ddot{\boldsymbol{u}}_t = \tilde{\boldsymbol{F}}_t^{ext} + \tilde{\boldsymbol{F}}_{0t}^a + q_\infty \tilde{\boldsymbol{K}}_{tu}^a \boldsymbol{v}_b + q_\infty \tilde{\boldsymbol{K}}_{tc}^a \boldsymbol{\delta}_c$$

The equation components are defined by the user through the TRIM and SUPORT cards

- TRIM card defines values for

Flight condition	Mach, q_∞
URDD1, URDD2, ...	$\ddot{\boldsymbol{u}}_t$
ALPHA, BETA, ROLL, PITCH, YAW	$\boldsymbol{v}_b$
Control surfaces	$\boldsymbol{\delta}_c$

- The SUPORT card is used to define the rigid motion



Aeroelastic static analysis example (1)

Definition of the Smartcad input file (`inputMain_trim.inc`)

```
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
$ DATA ADDED AFTER GUESS SIZING
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
$ ADD PAYLOAD
INCLUDE B747-100_v00CONM_CONF1.inc
$-------2-------3-------4-------5-------6-------7-------8-------9-------10$
SET TRIM SOLVER
SOL 144
AEROS           0       10.483  59.64   551     0       0
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
$ ADD MANEUVERS
INCLUDE B747-100_CAS25.inc
$
$ INCLUDE MODEL
INCLUDE B747-100_v00.inc
$
```

Trim condition definition (`B747-100_CAS25.inc`)

```
TRIM=   1
TRIM    1       1       0.5232360.0     SIDES   0.0     ROLL    0.0
        PITCH   0.0     YAW     0.0     URDD2   0.0     THRUST  0.0
        URDD4   0.0     URDD5   0.0     URDD6   0.0     CLIMB   0.0
        BANK    0.0     HEAD    0.0     URDD3   24.525  aileronr0
        flap1r  0       flap2r  0       rudder1 0
```

Aeroelastic static analysis example (2)

Performs a trim analysis using the configuration with ID 1:

```
global beam_model
filename_sma = 'inputMain_trim.dat';
beam_model = load_nastran_model(filename_sma);
solve_free_lin_trim('INDEX', 1);
plotLinearDispl(beam_model, beam_model.Res, 1, 1);
```

Solution available as

- `beam_model.Res` data structure
- text file (`inputMain_trim_man_1.txt`)

Aeroelastic static analysis example (3)

beam_model.Res data structure

```
>> beam_model.Res
ans =
        SOL: 'Static linear unrestrained trim'
         FM: [1x1 struct]
         CS: [1x1 struct]
      state: [1x1 struct]
        Bar: [1x1 struct]
       Beam: [1x1 struct]
     NDispl: [512x6 double]
        NRd: [3x3x512 double]
       Aero: [1x1 struct]
     Struct: [1x1 struct]
     CPaero: [1x1 struct]
       Trim: [1x1 struct]
         WB: [1x1 struct]
```

Internal forces on bars (described later)

```
     CForces: [2x6x82 double]
    CStrains: [2x6x82 double]
   CStresses: [1x1 struct]
         CSM: [1x1 struct]
           R: [4-D double]
      Colloc: [2x3x82 double]
```

Node displacement

Aerodynamic forces/coefficients

```
   RStab_Der: [1x1 struct]
  RIntercept: [1x1 struct]
   DStab_Der: [1x1 struct]
  DIntercept: [1x1 struct]
   RTrim_sol: [1x1 struct]
   DTrim_sol: [1x1 struct]
         Qaa: [988x988 double]
         Kax: [988x10 double]
      QaaDOF: [1008x1008 double]
      KaxDOF: [1008x10 double]
         Fa0: [988x1 double]
      Fa0DOF: [1008x1 double]
           H: [5x10 double]
          H0: [5x1 double]
    DIVERG_Q: 0
      Fa0tot: [1008x2 double]
         XYZ: [458x3 double]
    F0_RTrim: [458x3 double]
    F0_DTrim: [458x3 double]
  RStability: [1x1 struct]
  DStability: [1x1 struct]
```

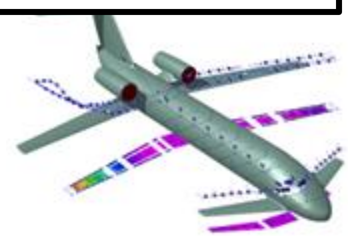
Structural matrices M and K (described later)

Aerodynamic forces/coefficients on aerodynamic mesh

```
    State: [1374x5 double]
  Control: [1374x5 double]
     Defo: [1374x1008 double]
       F0: [458x3 double]
```

Trim condition definition

Mass distribution of the aircraft



Aeroelastic static analysis (4)

- text file (`inputMain_trim_man_1.txt`)

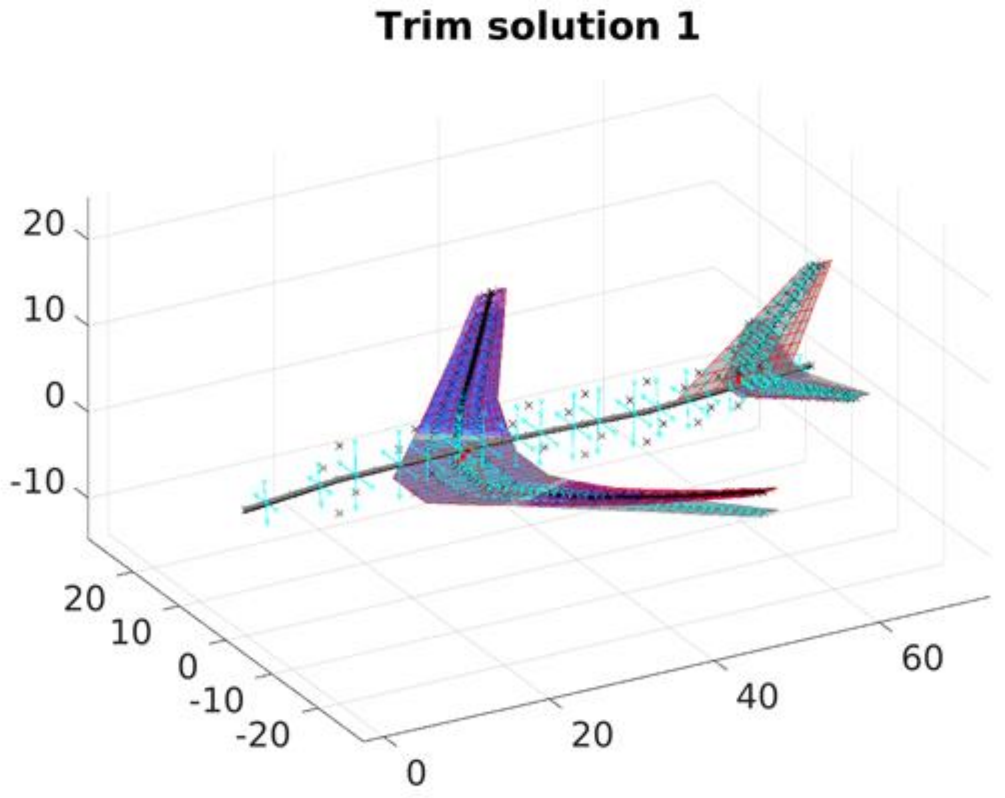
```
ELASTIC TRIM RESULTS
 - X acc:       -6.95218e-13 [m/s^2].
 - Y acc:       0 [m/s^2].
 - Z acc:       24.525 [m/s^2].
 - P-DOT:       0 [rad/s^2].
 - Q-DOT:       0 [rad/s^2].
 - R-DOT:       0 [rad/s^2].
 - Alpha:       8.46657 [deg].
 - Sideslip:    0 [deg].
 - Roll rate:   0 [-] (p*BREF/(2VREF)).
 - Pitch rate: 0 [-] (q*CREF/(2VREF)).
 - Yaw rate:    0 [-] (r*BREF/(2VREF)).
 - Control flap1r:    0 [deg].
 - Control flap2r:    0 [deg].
 - Control aileronr:    0 [deg].
 - Control elev1r:    12.3955 [deg].
 - Control rudder1:    0 [deg].
```

```
AERODYNAMIC DERIVATIVES
ALPHA
 NAME              RIGID          ELASTIC          RATIO E/R
Cy/alpha          -0.00000        0.00000          -0.43068
Cz/alpha          3.97805         3.62291          0.91072
Cl/alpha          0.00000         -0.00000         -204.05294
Cm/alpha          -0.34431        -0.24801         0.72031
Cn/alpha          -0.00000        0.00000          -0.11343
BETA
 NAME              RIGID          ELASTIC           RATIO E/R
Cy/beta           0.43077         0.40741          0.94579
Cz/beta           -0.00000        -0.00000         0.85057
Cl/beta           -0.13204        -0.12380         0.93760
Cm/beta           -0.00000        -0.00000         0.95816
Cn/beta           0.07728         0.06671          0.86320
```

Aeroelastic static analysis (5)

Display the deformed aircraft

```
plotLinearDispl(beam_model, beam_model.Res, 1, 1);
```



Aeroelastic static analysis - recover nodal load (1)

The internal forces on bars are defined in beam_model.Res.Bar struct

```
>> beam_model.Res.Bar
ans =
       CForces: [2x6x82 double]
      CStrains: [2x6x82 double]
     CStresses: [1x1 struct]
           CSM: [1x1 struct]
             R: [4-D double]
        Colloc: [2x3x82 double]
```

Internal forces stored as

Loads on the structural nodes, for each bar, on the evaluation points.

beam_model.Res.Bar.CForces   [2x6xnumberOfBars]

Coordinates of the points for load recovery

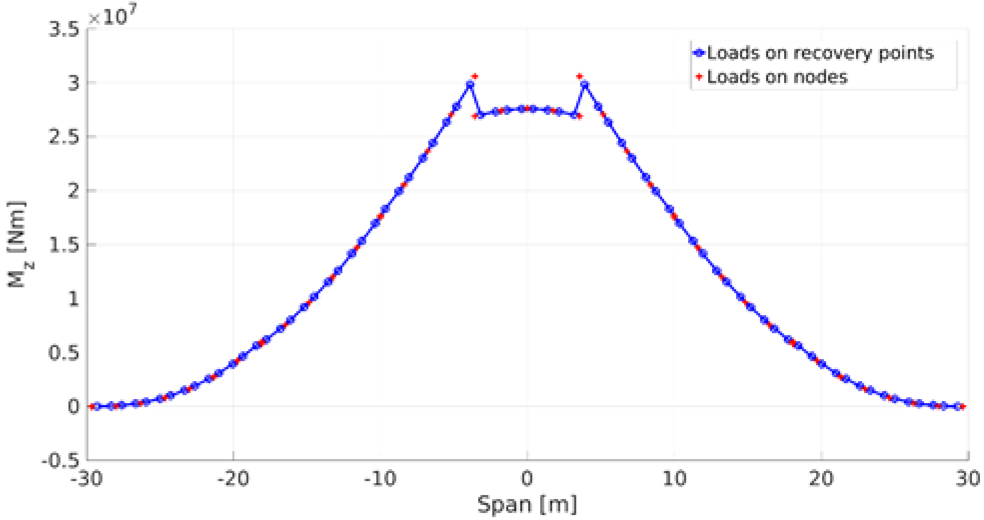
beam_model.Bar.Colloc [2 x 3 x numberOfBars]

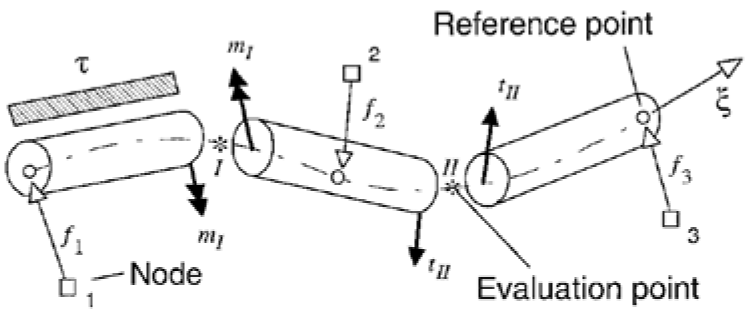
Example of load recovery procedure:

```
% Get position of bar in dataset from element ID
barID = 2001;
position = beam_model.Bar.ID==barID;

% Forces on the first recovery point of the bar (in the bar reference system)
barForces = beam_model.Res.Bar.Cforces(1,:,position);

% Rotate forces in the basic reference system
R = beam_model.Bar.R(:,:,2,position);
barForces_basic = [R*barForces(1:3)'; R*barForces(4:6)'];

% Get the coordinates of the evaluation point
pointCoord = beam_model.Bar.Colloc(1,:,position);
```

Aeroelastic static analysis - recover nodal load (2)

The internal forces on bars are defined in the load evaluation points. It is possible to obtain them in the beam nodes





The load transfer can be performed using the getForcesOnNodes function

```
[NForces, nodeCoords] = getForcesOnNodes(beam_model, beam_model.Res.Bar.CForces);
```

NForces [2 x 6 x numberOfBars] : loads on the structural nodes, for each bar.
NodeCoords [2 x 3 x numberOfBars] : coordinates of the nodes used for load recovery.

Recover structural matrices

Mass and stiffness matrices can be recovered from `beam_model.Res.Struct`

```
>> beam_model.Res.Struct
        M:  [988x988 double]
        K:  [988x988 double]
        F:  [988x1 double]
        D:  [982x6 double]
     ldof:  [1x982 double]
     rdof:  [25 26 27 28 29 30]
```

The matrices contain only the free DOFs

- Without considering the aeronodes
- Including the mid-bar nodes
- Without DOFs constrained by RBE2 and SPC

The corrispondence DOF – position in matrices can be obtained as

`beam_model.Node.DOF2(iNode,iComponent)`

This gives the position of the component `iComponent` of node `iNode`, if it is zero the component is constrained (`RBE0, RBE2, SPC`)

Divergence analysis example (1)

Divergence dynamic pressure computed by solving an eigenvalue problem

$$\det\left(\tilde{\boldsymbol{K}}_{dd} - \lambda \tilde{\boldsymbol{K}}^{a}_{dd}\right) = 0$$

- The lowest real, positive eigenvalue will provide the divergence dynamic pressure.
- The computation is performed using matrices in mean axes.

Definition of the Smartcad input file (`inputMain_diverg.inc`)

```
$
PARAM    DIVERG  1
$
$
PARAM    CM0FUS  -0.01
PARAM    CMAFUS  0.03

[...]
```

Divergence analysis example (2)

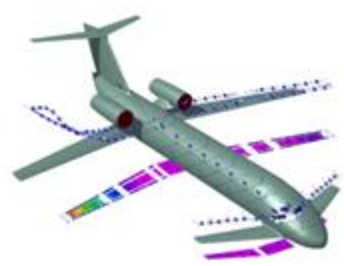
Screen output

```
[...]

 - Control flap1r:   0 [deg].
 - Control flap2r:   0 [deg].
 - Control aileronr:   0 [deg].
 - Control elev1r:   7.43586 [deg].
 - Control rudder1:   0 [deg].
done.
 Solving for unrestrained aeroelastic divergence...
 - Divergence dynamic pressure: 5.084410e+06 [Pa].
done.
 - Updating vlm model in Aero.lattice_defo...done.
 [...]
```

Results Structure

```
>> beam_model.Res

ans =

  struct with fields:

           SOL: 'Static linear unrestrained
trim'
            FM: [1x1 struct]
            CS: [1x1 struct]
         state: [1x1 struct]
           Bar: [1x1 struct]
          Beam: [1x1 struct]
        NDispl: [512x6x2 double]
           NRd: [3x3x512x2 double]
          Aero: [1x1 struct]
        Struct: [1x1 struct]
        CPaero: [1x1 struct]
          Trim: [1x1 struct]
            WB: [1x1 struct]
```

Deformation associated to divergence mode



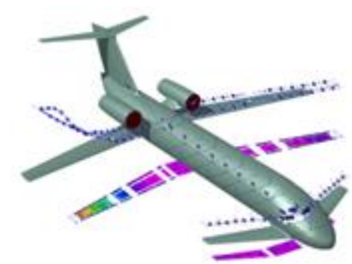
Specification of fuseleage moment coefficients (1)

It is possible to directly provide the fuselage $C_{m\alpha}$ coefficients

- The provided value is not used in the aerodynamic or structural computations
- The provided value is only used in the computation of the displayed static margin

Definition of the Smartcad input file (`inputMain_diverg.inc`)

```
[...]
PARAM   CMAFUS  0.03
[...]
```



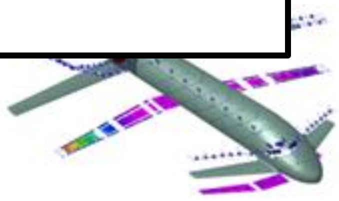
Specification of fuselage moment coefficients (2)

Results file without Cma definition (`inputMain_man_1.txt`)

```
STABILITY

 NAME                                        RIGID                ELASTIC
 Center of gravity                XG/CREF    3.20550
 Neutral point at fixed stick XN/CREF        3.15124              3.10956
 Control point at fixed stick XC/CREF        6.10821              5.77632
 Static margin         (XN-XCG)/CREF         -0.05427             -0.09594
 Static margin index (XN-XCG)/(XC-XN)        -0.01835             -0.03598
```

Results file with Cma definition (`inputMain_diverg_1.txt`)

```
STABILITY

 NAME                                        RIGID                ELASTIC
 Center of gravity                XG/CREF    3.20550
 Neutral point at fixed stick XN/CREF        3.34434              3.32028
 Control point at fixed stick XC/CREF        6.10821              5.77632
 Static margin         (XN-XCG)/CREF         0.13884              0.11477
 Static margin index (XN-XCG)/(XC-XN)        0.05023              0.04673
```



Modification of aerodynamic forces with external data

Linear aerodynamic forces:

Aerodynamic forces on structural DOFs

Structural DOFs

$$\frac{\boldsymbol{F}_z^a}{q_\infty} = \boldsymbol{F}_0^a + \boldsymbol{K}_u^a \boldsymbol{v}_b + \boldsymbol{K}_c^a \boldsymbol{\delta}_c + \boldsymbol{K}_z^a \boldsymbol{u}_z$$

By default all the components are obtained using the VLM method.

There is the possibility to:

- Provide directly F0a
- Provide directly columns of Kca and Kua
- Provide scaling factors for Kca and Kua

Modification of aerodynamic forces with external data

Definition of steady forces

The following files are used in this portion of the tutorial:

- inputMain_trimExtType2.dat : Smartcad input file
- exampleMain_trimExtType2.m : Matlab script

Definition of the Smartcad input file (inputMaing_trimExtType2.dat)

- The TRIMEXT entry can be used to introduce additional aerodynamic components, with the aerodynamic loads
- Several entries can be defined specifying each component (wing, fuselage etc...()
- A beam spline is used to connect the aerodynamic mesh to the structural mesh

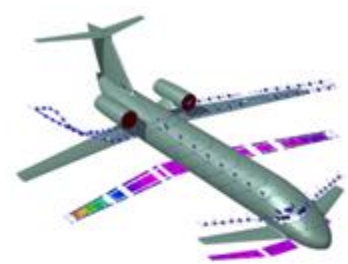
```
$
$-------2-------3-------4-------5-------6-------7-------8-------9---
TRIMEXT 100     './CFDresults/wing-right_base.bdf'
        1       0       0       210
        201     202     204
[...]
```

File with the mesh and load definition (see next slide)

ID of SET entry for spline definition

Position and reference system

CAEROs that will be substituted by the imported elements



Modification of aerodynamic forces with external data

Definition of the Imported mesh file (`CFDresults/wing-right_base.bdf`)

The mesh file contains

- the imported nodes
- the elements
- the pressure coefficients on each element

```
$
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
GRID    1100001 0       48.4992 29.5977 1.21137 0
[...]
CTRIA3  1100001 0 1100001 1100002 1100003
[...]
PLOAD4  1000    1109614 -1.089
[...]
```

Modification of aerodynamic forces with external data

Definition of force derivatives (substitution of columns)

- Forces are obtained by finite differences, defining the value for perturbed ALPHA, BETA, … and subtracting the value at the reference condition
- This correction is enabled by the PARAM DER_TYPE=2

Definition of the Smartcad input file (inputMaing_trimExtType2.dat)

```
PARAM   DER_TYPE       2
$-------2-------3-------4-------5-------6-------7-------8-------
TRIMEXT 100     './CFDresults/wing-right_base.bdf'
        1       0       0       1100    210
        201           202           204
$-------2-------3-------4-------5-------6-------7-------8-------
DEREXT  1100
        ALPHA   0.0349  './CFDresults/wing-right_alpha.bdf'
$
[...]
```

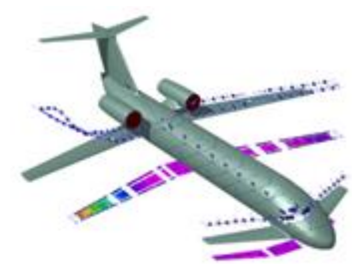
ID of DEREXT card defining the CFD results in perturbed condition

Several lines like this can be repeated for each DEREXT entry

Perturbed variable

Value of perturbed variable

File with results in perturbed conditions



Modification of aerodynamic forces with external data

Modification of force derivatives (scaling of columns)

- Forces are obtained by scaling a whole column of matrices
- This correction is enabled by the PARAM DER_TYPE=1

Definition of the Smartcad input file (inputMain_trimExtType1.dat)

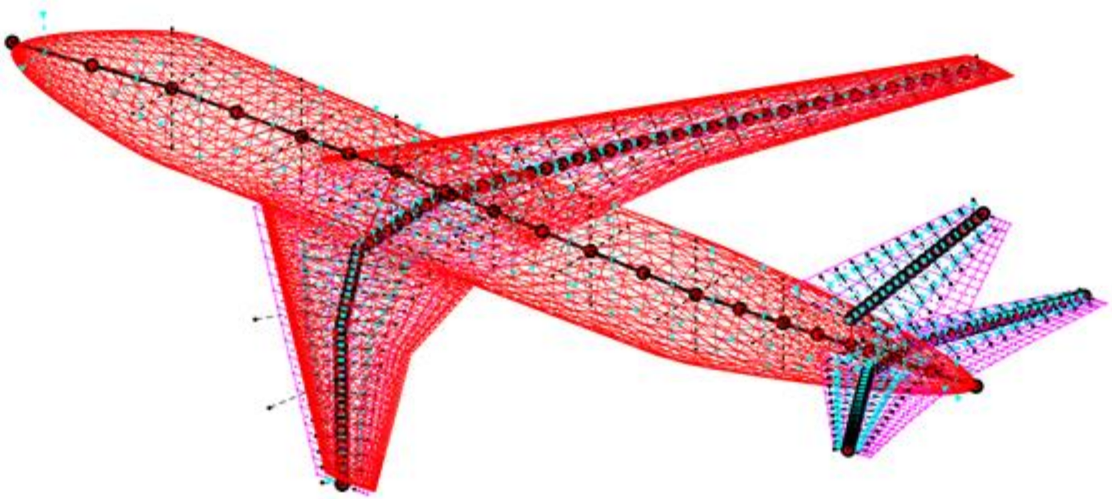
```
PARAM   DER_TYPE        1
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
TRIMEXT 100     './CFDresults/wing-right_base.bdf'
        1       0       0       210
        201     202     204
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
PARAM   DER_FILE './CFDresults/correctionFactors.bdf'
$
[...]
```

File with scaling factors

```
alpha 0.90
elev1r 0.8
```

Modification of aerodynamic forces with external data

- Also a partial substitution of aerodynamic surfaces can be performed.
- The VLM computation will always be performed considering all the CAEROs to get the correct aerodynamic interference.
- The forces generated by the substituted CAEROs will be discarded



Modal / flutter analysis (1)

The aeroelastic module in NeoCASS can perform

- Modal analysis on the structure defined on the Smartcad input file
- Import modal basis obtained from an external solver (not shown in this tutorial)
- Compute unsteady aerodynamic generalized forces in frequency domain based on the Doublet Lattice Method.
- Compute the roots of the aeroelastic system in a velocity range, using a continuation method.

The following files are used in this portion of the tutorial:

- `inputMain_flutter.dat` : Smartcad input file
- `exampleMain_flutter.m` : Matlab script

Modal / flutter analysis (2)

Aeroelastic system equations in frequency domain:

Aerodynamic matrices from DLM

Modal displacement

$$\left[\mathbf{M}s^2 + \mathbf{C}s + \mathbf{K} - q_\infty(v)\mathbf{H}_{\mathbf{am}}(p(s), M_\infty(v))\right]\mathbf{q} = 0$$

Laplace variable

Modal (structural) mass, damping and stiffness

Reduced frequency

Compact form: $\mathbf{F}(s,v)\mathbf{q} = 0$

The flutter eigenvalue and mode are found by solving a system of ODE on the flight speed

$$\begin{bmatrix} \frac{\partial \mathbf{F}(s,v)}{\partial s} & \mathbf{F}(s,v) \\ 0 & \mathbf{q}^H \end{bmatrix} \begin{bmatrix} \frac{ds}{dv} \\ \frac{d\mathbf{q}}{dv} \end{bmatrix} = \begin{bmatrix} -\frac{\partial \mathbf{F}(s,v)}{\partial v} \\ 0 \end{bmatrix}$$

Modal / flutter analysis example (1)

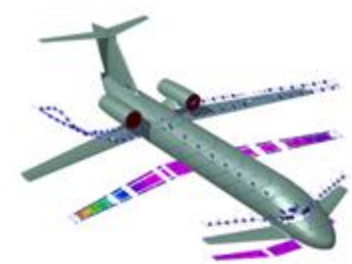
Smartcad input file (inputMain_flutter.dat)

```
[...]
SOL 145
$ Eigenvalue solver parameters
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
EIGR    1               0       300             20              MASS
$ Doublet lattice solver parameters
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
AERO    551     300     10.483  1.225   0       0       2       100     59.64
$ Doublet lattice aerodynamic transfer matrix points
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
MKAERO1 0.50
        0.0005  0.001   0.002   0.005   0.01    0.05    0.1     0.2
MKAERO1 0.50
        0.5     1       1.5     2
$ Selected modes defining the modal basis
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
MSELECT        1       2       3       4
               9      10      11      12
              17      18      19      20
$ Selected modes for tracking in V-g plot
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
FMODES         7       8       9      10      11      12      13      14
              15      16      17      18      19      20
UMODES         1       2       3       4       5       6       7       8
               9      10      11      12      13      14      15      16
              17      18      19      20
```

Compute the first 20 modes with frequency lower than 300Hz

Define aerodynamic data, including reference surface (551 m^2) and reference span (59.64m)

Modes used to compute aerodynamic matrices

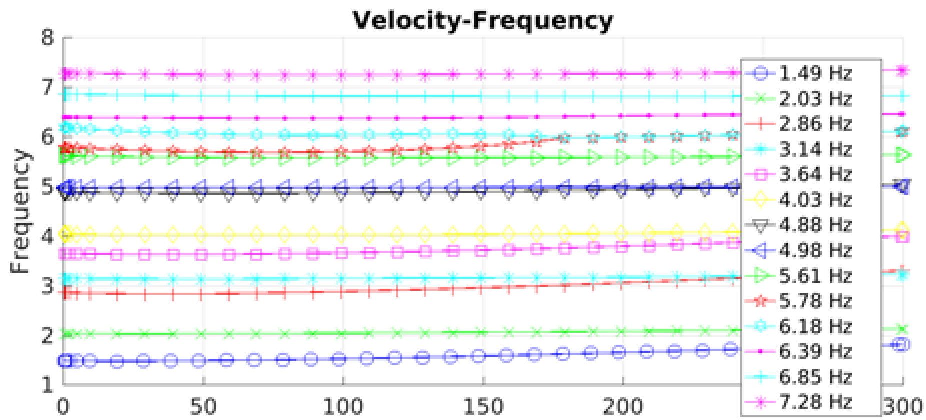


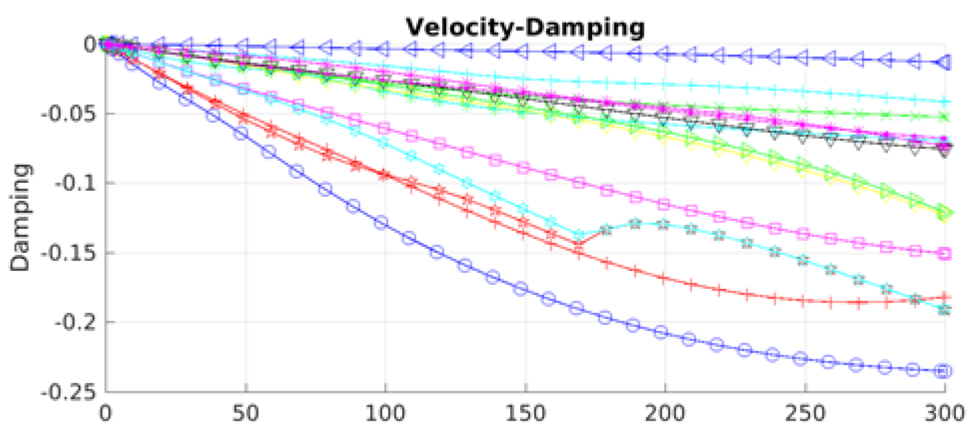
Modal / flutter analysis example (2)

Matlab script `exampleMain_flutter.m`

```
global beam_model

filename_sma = 'inputMain_flutter.dat';

beam_model = load_nastran_model(filename_sma);

solve_linflutt();
```

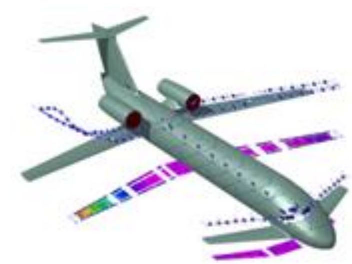
Modal / flutter analysis example (3)





Results

```
global fl_model

fl_model
         Res: [1x1 struct]
       param: [1x1 struct]
         ref: [1x1 struct]
      struct: [1x1 struct]
      interp: [1x1 struct]
        aero: [1x1 struct]
```



Recover flutter results

```
global fl_model
>> fl_model.aero
ans =
      Qhh: [20x20x24 double]
     Mach: 0.5000
    Klist: [5.0000e-04 1.0000e-03 0.0020 0.0050 0.0100 0.0500 0.1000 0.2000
0.5000 1 1.5000 2]
      rho: 1.2250

>> fl_model.Res.data
ans =
           k: {1x14 cell}
    Velocity: {1x14 cell}
           g: {1x14 cell}
        Freq: {1x14 cell}
       RealE: {1x14 cell}
       ImagE: {1x14 cell}
      g_Vder: {1x14 cell}
      F_Vder: {1x14 cell}
```

Condensed aerodynamic matrix

Velocities corresponding to the computed roots

Real/Imaginary part of the aeroelastic roots

Recover modal data

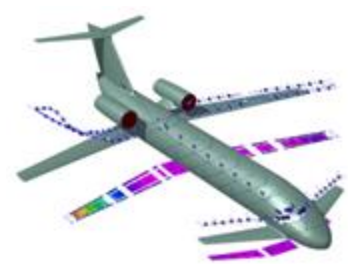
```
>> beam_model.Res

ans =

       SOL: 'Linear flutter'
    NDispl: [512x6x20 double]
       NRd: [4-D double]
       Bar: [1x1 struct]
      Beam: [1x1 struct]
       Mmm: [20x20 double]
       Kmm: [20x20 double]
         M: [988x988 double]
         K: [988x988 double]
     Omega: [20x1 double]
         V: [988x20 double]
        ID: [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
```

Nodal displacements on all structural nodes (modal matrix on all nodes)

Modal mass and stiffness matrices

Complete mass and stiffness matrices (defined as in the trim analysis)

Modal matrix on the free DOFs only



Dynamic response

NeoRESP is the module that can be used for dynamic response analysis, three types of analysis can be performed

- Gust response analysis
- Control surface deflection
- External forcing

The effects of the three types of forcing terms can be combined.

The computation is performed in frequency domain on a modally reduced system, with unsteady aerodynamic obtained using the DLM method

Dynamic response – input definition

In order to run NeoRESP is necessary to set

- The forcing term;
- The flight condition;
- The output of the simulation (displacements, loads, stresses …)

In addition, also the data concerning the modal computation and the DLM aerodynamic matrix need to be set.

All the data can be defined in the input Smartcad file.

Simulation Parameters – forcing term

Gust input:

Amplitude | Duration | Delay | Direction (vertical or horizontal gust)

```
GUST      1          17.07    0.12    0.0     3
          sin(pi*(x+25)/50)
          (1-cos(2*pi/0.1200*t))*0.5
```

Spanwise variation of the gust speed

Time evolution of the gust speed

Control surface deflection:

```
$         ID        label      Amplit    TMAX     Delay
SURFDEF   1         elev1r     1.0       0.5      0.0
          sin(2*pi/0.5*t)
```

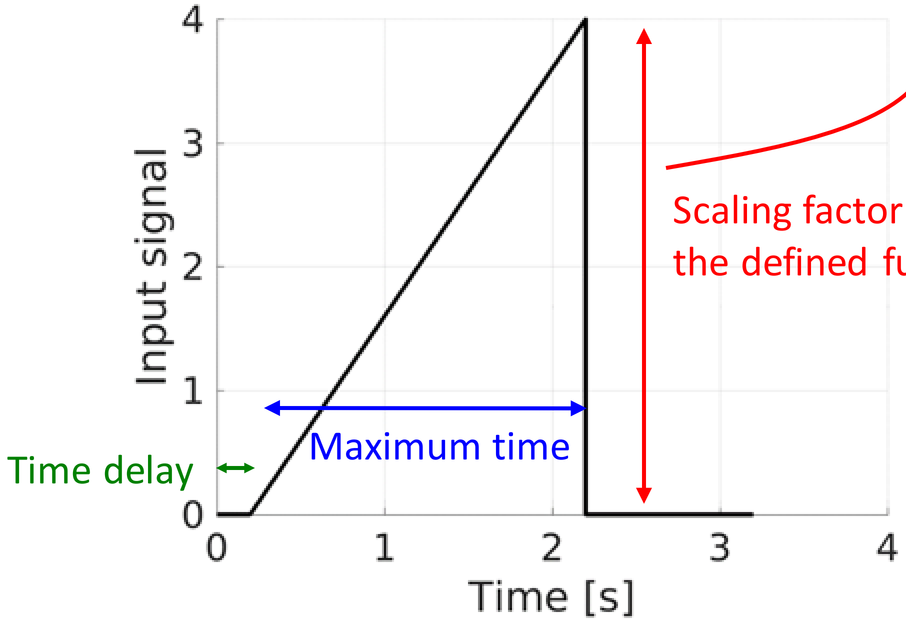
External load:

```
$         ID        NODE       DOF       AMPLIT    TMAX     DELAY
DLOAD     1         2000       3         1.0e+3    0.5      0.0
          sin(2*pi/0.5*t)
```

Simulation Parameters – forcing term

Example of input definition

```
$         ID        NODE      DOF       AMPLIT    TMAX      DELAY
DLOAD     1         2000      3         1.0       2.2       0.2
          2*t
```

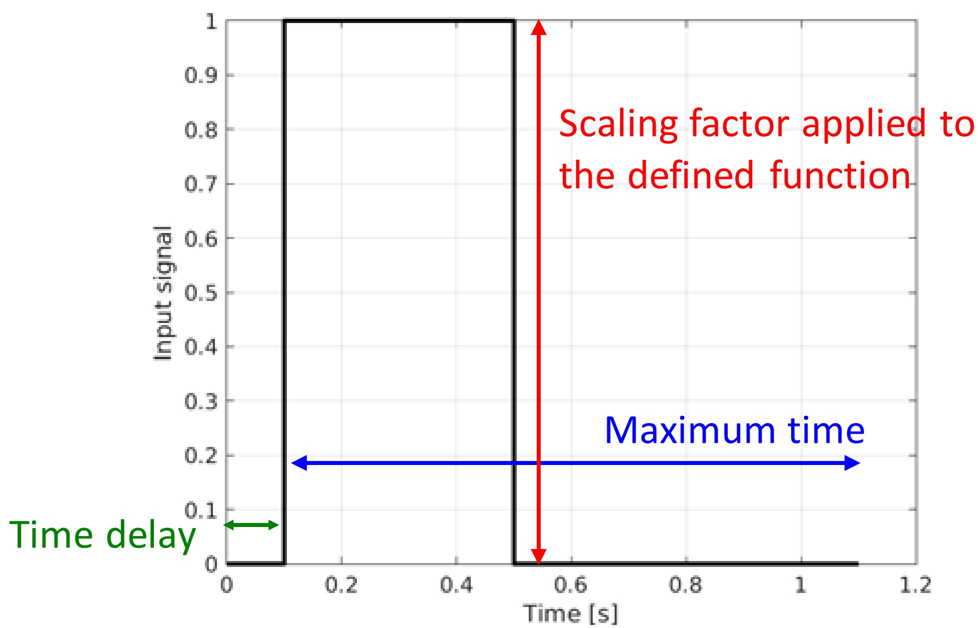


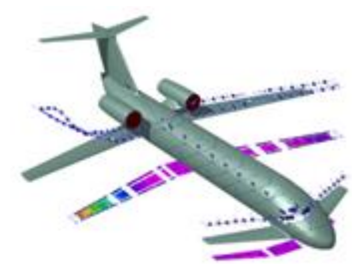
Scaling factor applied to the defined function

Simulation Parameters – forcing term

It is also possible to define the input through a list of points

```
$        ID      NODE    DOF    AMPLIT      TMAX    DELAY     TYPE
DLOAD    2       2020    3         1.0      1.0     0.1       TABLED
         4000
$
TABLED1          4000
                 0.      1.      0.4       1.      0.4       0.        ENDT
```





Simulation Parameters – flight condition and output

Flight condition:

```
PARAM    MODACC   0
PARAM    VREF     238.0
PARAM    RHOREF   1.225
PARAM    MACH     0.7
```

0: Mode acceleration is active

Output selection:

```
SET 1 =     2002    4000
SET 2 =     2000
DISP= ALL
ACCELERATION= 2
IFORCE= 1
```

Node/element sets

Simulation results

The simulation results are stored in the `dyn_model.Res` data structure

An example of how to extract data from this structure is shown in the following examples.

```
dyn_model =
      dlm: [1x1 struct]
     beam: [1x1 struct]
      flu: [1x1 struct]
     gust: [1x1 struct]
      Out: [1x1 struct]
      Res: [1x1 struct]
```

Aerodynamic matrices

Model and results from modal analysis

Flutter results

Aerodynamic matrices associated with gust input

Simulation results

Simulation results

```
>> dyn_model.dlm.data

ans =

          D: [458x458x12 double]
         Cp: [458x25x12 double]
        Qhh: [20x20x12 double]
    c_displ: [458x3x25 double]
    dwnwash: [458x25x12 double]
    n_displ: [1832x3x25 double]
       invD: [458x458x12 double]
        Qnh: [1008x20x12 double]
        Qhd: [20x5x12 double]
        Qdh: [5x20x12 double]
        Qdd: [5x5x12 double]
        Qnd: [1008x5x12 double]
        Hag: [20x1x12 double]
```

```
>> dyn_model.gust

ans =

    dwnwash: [458x1 double]
         Cp: [458x458x12 double]
        Qhg: [20x458x12 double]
        Qng: [1008x458x12 double]
        Qdg: [5x458x12 double]
```

Aerodynamic matrices for mode acceleration

Aerodynamic matrices from control surface deflection

Gust response (1)

This example shows how to simulate a gust response of an aircraft

The following files are used in this portion of the tutorial:

- `inputMain_gust.dat` : Smartcad input file
- `exampleMain_gust.m` : Matlab script

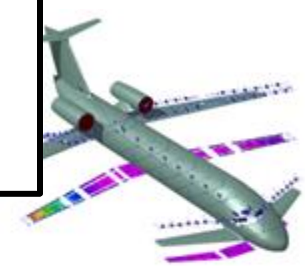
Gust response (2)

Smartcad input file

```
[...]
SOL 146
$
PARAM     MODACC        0
PARAM       VREF 170.000
PARAM     RHOREF 1.22500
PARAM       MACH 0.50000
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
$ Output set (node list)
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
SET 3 =     1004    2018
SET 4 =     1004    2018
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
$ Choose output (displacements, velocities, accelerations and internal forces)
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
IFORCE= ALL
DISP= ALL
VELOCITY= 3
ACCELERATION= 4
HINGEFORCE= ALL
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
GUST= 1
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
$ Gust input
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
GUST    1       17.07   0.40    0.0     3
        1
        (1-cos(2*pi/0.4000*t))*0.5
```

Enable load recovery with mode acceleration

Select the gust ID used in simulation



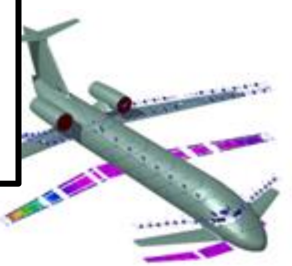
Gust response (3)

Matlab script

```
global beam_model
global dyn_model

filename_sma = 'inputMain_gust.dat';

init_dyn_model(filename_sma)

solve_free_lin_dyn('dT', 5e-3, 'Tmax', 6);

positionWR_global  = find(beam_model.Bar.ID==2002);
positionWR_results = find(beam_model.Param.IFORCE==positionWR_global);

wrbm_dirrec = squeeze(dyn_model.Res.IFORCE(6,1,positionWR_results,:));
wrbm_modac  = squeeze(dyn_model.Res.MODACC.IFORCE(6,1,positionWR_results,:));
wrtm_dirrec = squeeze(dyn_model.Res.IFORCE(4,1,positionWR_results,:));
wrtm_modac  = squeeze(dyn_model.Res.MODACC.IFORCE(4,1,positionWR_results,:));
```

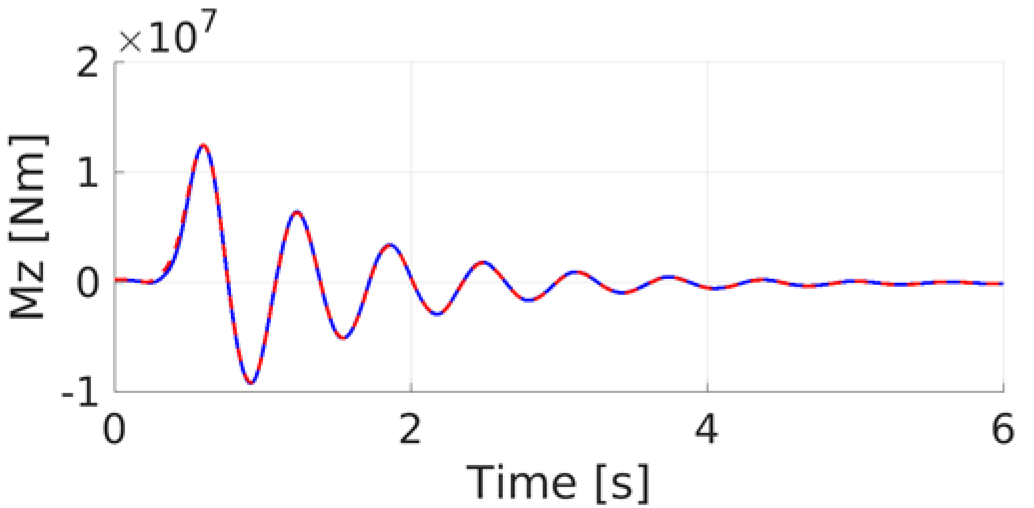
Define time step and duration

Extract loads from solution

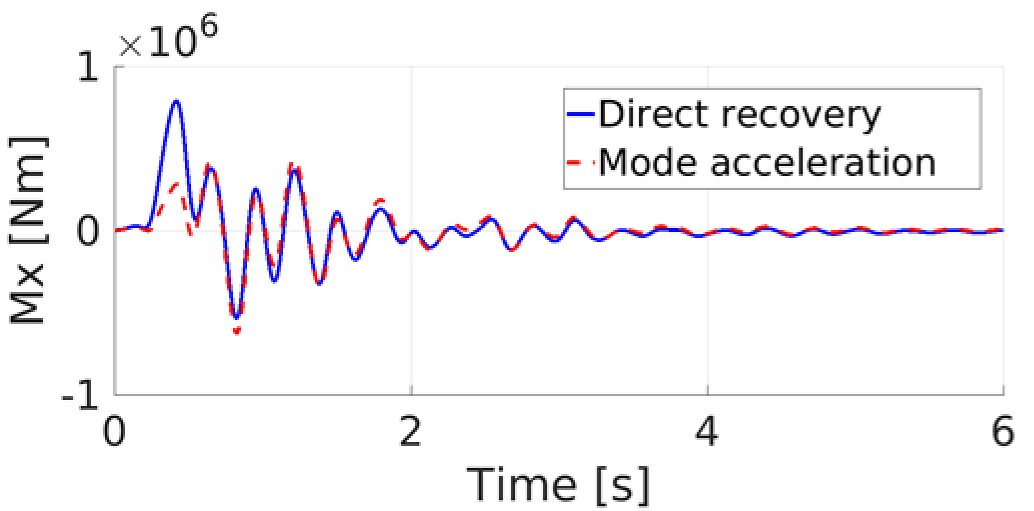


Gust response (4)

Wing root Bending moment



Wing root torsional moment



Gust response (5)

```
>> dyn_model.Res
     Gust_profile: [1x1201 double]
             Time: [1x1201 double]
            Qload: [20x1201 double]
            Qextf: [20x1201 double]
                Q: [20x1201 double]
               Qd: [20x1201 double]
            Qddot: [20x1201 double]
          Cy_mode: [1x1201 double]
          Cz_mode: [1x1201 double]
          Cl_mode: [1x1201 double]
          Cm_mode: [1x1201 double]
          Cn_mode: [1x1201 double]
             DISP: [512x6x1201 double]
         VELOCITY: [2x6x1201 double]
     ACCELERATION: [2x6x1201 double]
           IFORCE: [4-D double]
          HF_gust: [5x1201 double]
          HF_mode: [5x1201 double]
          Cy_gust: [1x1201 double]
          Cz_gust: [1x1201 double]
          Cl_gust: [1x1201 double]
          Cm_gust: [1x1201 double]
          Cn_gust: [1x1201 double]
           MODACC: [1x1 struct]
```

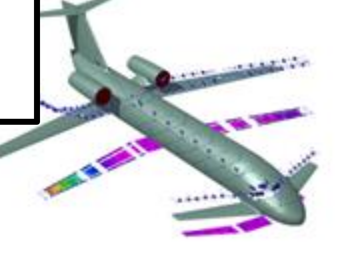
Modal coordinates and time derivatives

Outputs as required by the DISPLACEMENT=, VELOCITY= … cards

Aerodynamic hinge mements divided by components
HF = HF_gust+HF_mode+HF_surf
(HF_surf is not present here since no surface input is used)

Loads obtained with mode acceleration method

```
>> dyn_model.Res.MODACC
    IFORCE: [4-D double]
      DISP: [512x6x1201 double]
```



Gust response (6)

Recover output from a reduced output set.

Example of recovery of accelerations.

```
% Get position of node in dataset starting from node ID
nodeID = 2018;
position = find(beam_model.Node.ID==nodeID);

% Get the position of the node in the results dataset
positionInRespDISP = dyn_model.beam.Param.DISP==position;

% Get the time history (of the 6 components of displacements)
timeHistory = dyn_model.Res.DISP(positionInRespDISP,3,:);
```

Elevator deflection response (1)

This example shows how to simulate the response of an aircraft after the deflection of the elevator

The following files are used in this portion of the tutorial:

- `inputMain_cSurf.dat` : Smartcad input file
- `exampleMain_cSurf.m` : Matlab script

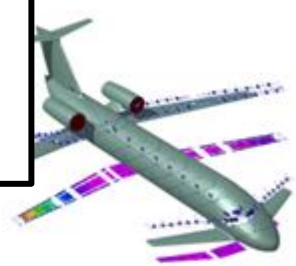
Elevator deflection response (2)

Smartcad input file

```
[...]
SOL 146
$
PARAM     MODACC        0
PARAM       VREF 170.000
PARAM     RHOREF 1.22500
PARAM       MACH 0.50000
[...]
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
SURFDEF= 2
$-------2-------3-------4-------5-------6-------7---
$ Control surface input
$-------2-------3-------4-------5-------6-------7---
SURFDEF 1       aileronr1.0     0.5     0.0
              sin(2*pi/0.5*t)
SURFDEF 2       elev1r  1.0     0.5     0.0
              sin(2*pi/0.5*t)
SURFDEF 3       rudder1 1.0     0.5     0.0
              sin(2*pi/0.5*t)
$
```

Enable load recovery with mode acceleration

Select the ID used in simulation among the Ids of the SURFDEF cards



Elevator deflection response (3)

Matlab script (no variation w.r.t. gust response!)

```
global beam_model
global dyn_model

filename_sma = 'inputMain_cSurf.dat';

init_dyn_model(filename_sma)

solve_free_lin_dyn('dT', 5e-3, 'Tmax', 6);

positionWR_global  = find(beam_model.Bar.ID==2002);
positionWR_results = find(beam_model.Param.IFORCE==positionWR_global);

wrbm_dirrec = squeeze(dyn_model.Res.IFORCE(6,1,positionWR_results,:));
wrbm_modac  = squeeze(dyn_model.Res.MODACC.IFORCE(6,1,positionWR_results,:));
wrtm_dirrec = squeeze(dyn_model.Res.IFORCE(4,1,positionWR_results,:));
wrtm_modac  = squeeze(dyn_model.Res.MODACC.IFORCE(4,1,positionWR_results,:));
```

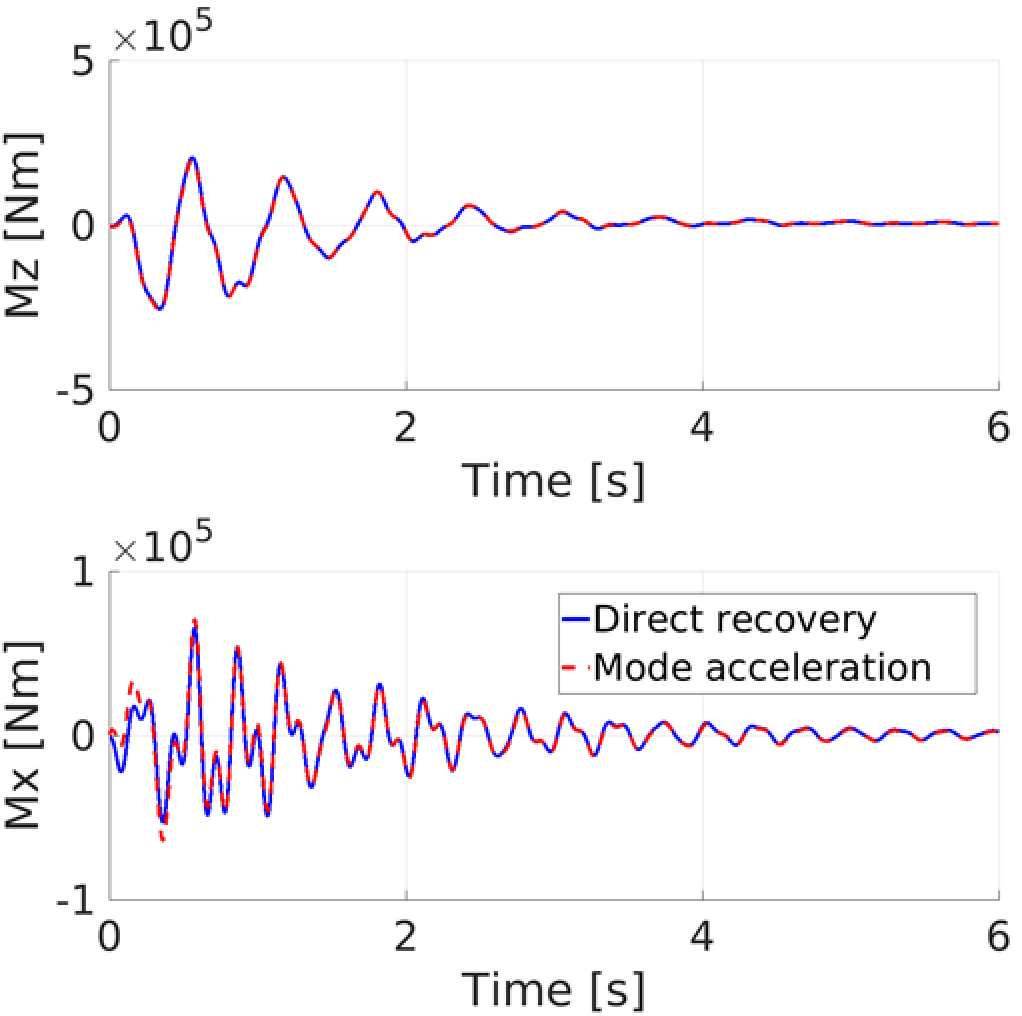
Define time step and duration

Extract loads from solution

Elevator deflection response (4)

Wing root Bending moment

Wing root torsional moment



External load response (1)

This example shows how to simulate the response of an aircraft after an external load applied on two wing nodes in the z-direction

The following files are used in this portion of the tutorial:

- `inputMain_dload.dat` : Smartcad input file
- `exampleMain_dload.m` : Matlab script

External load response (2)

Smartcad input file

```
[...]
SOL 146
$
PARAM     MODACC        0
PARAM       VREF 170.000
PARAM     RHOREF 1.22500
PARAM       MACH 0.50000
[...]
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
LOAD=   1       2
$-------2-------3-------4-------5-------6-------7-------
$ Force input
$-------2-------3-------4-------5-------6-------7-------
$       ID      NODE    DOF     AMPLIT  TMAX    DELAY
DLOAD   1       2002    3       1.0e+3  0.1     0.2
        (1-cos(2*pi/0.1*t))*0.5
$       ID      NODE    DOF     AMPLIT  TMAX    DELAY
DLOAD   2       2020    3       1.0e+3  0.1     0.2
        (1-cos(2*pi/0.1*t))*0.5
```

Enable load recovery with mode acceleration

Select the ID used in simulation among the Ids of the DLOAD cards. The effects of the two loads will be superimposed

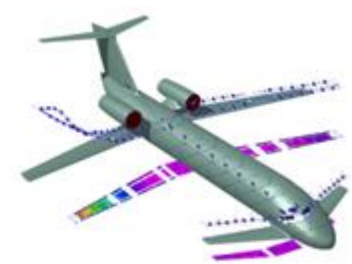
External load response (3)

Matlab script (no variation w.r.t. gust response!)

```
global beam_model
global dyn_model

filename_sma = 'inputMain_dload.dat';

init_dyn_model(filename_sma)

solve_free_lin_dyn('dT', 5e-3, 'Tmax', 6);

positionWR_global  = find(beam_model.Bar.ID==2002);
positionWR_results = find(beam_model.Param.IFORCE==positionWR_global);

wrbm_dirrec = squeeze(dyn_model.Res.IFORCE(6,1,positionWR_results,:));
wrbm_modac  = squeeze(dyn_model.Res.MODACC.IFORCE(6,1,positionWR_results,:));
wrtm_dirrec = squeeze(dyn_model.Res.IFORCE(4,1,positionWR_results,:));
wrtm_modac  = squeeze(dyn_model.Res.MODACC.IFORCE(4,1,positionWR_results,:));
```

Define time step and duration

Extract loads from solution



External load response (4)

```
>> dyn_model.Res

ans =

    Extload_profile: [2x1201 double]
               Time: [1x1201 double]
              Qload: [20x1201 double]
              Qextf: [20x1201 double]
                  Q: [20x1201 double]
                 Qd: [20x1201 double]
              Qddot: [20x1201 double]
            Cy_mode: [1x1201 double]
            Cz_mode: [1x1201 double]
            Cl_mode: [1x1201 double]
            Cm_mode: [1x1201 double]
            Cn_mode: [1x1201 double]
               DISP: [512x6x1201 double]
           VELOCITY: [2x6x1201 double]
       ACCELERATION: [2x6x1201 double]
             IFORCE: [4-D double]
            HF_mode: [5x1201 double]
             MODACC: [1x1 struct]
```

State-space model generation

It is possible to use NeoRESP to generate a linear state-space model of the aeroelastic system at a given flight condition.

The operation requires the definition of a complete solution for

- Modal analysis
- Unsteady aerodynamic

And thus can be performed as a post-processing of a dynamic response procedure

State-Space realization

The DLM provides the aerodynamic forces in the following format

$$f_a = q_\infty H_{am}(j\omega)q + q_\infty H_{ag}(j\omega)v_g + q_\infty H_{a\delta}(j\omega)\delta_c$$

- f_a Aerodynamic forces;
- q_∞ Dynamic pressure;
- q Modal coordinates;
- v_g Gust velocity;
- δ_c Control surface deflection;

State-Space realization

Each subsystem can be expressed in state-space form

$$f_{am} = q_\infty H_{am}(j\omega) q$$

becomes

$$\begin{cases} \dot{x}_{am} = A_{am} x_{am} + B_{am}^0 q + B_{am}^1 \dot{q} + B_{am}^2 \ddot{q} \\ \dfrac{f_{am}}{q_\infty} = C_{am} x_{am} + D_{am}^0 q + D_{am}^1 \dot{q} + D_{am}^2 \ddot{q} \end{cases}$$

A similar formulation can be obtained for f_{ag} and $f_{a\delta}$

A quasi-steady approximation can also be used, where only D_{am}^0, D_{am}^1 and D_{am}^2 are defined

State-Space realization

The transfer function is by first approximated as a right or left Matrix Fraction Description

$$H(s) = D(s)^{-1}N(s)$$ LMFD

or

$$H(s) = N(s)D(s)^{-1}$$ RMFD

$D(s)$ **is a matrix polynomial of order** n

Number of states in the generated model:

- LMFD $\rightarrow n \times n_{forces}$ states;
- RMFD $\rightarrow n \times n_{inputs}$ states;

Reference:

Ripepi, M., and P. Mantegazza. "Improved Matrix Fraction Approximation of Aerodynamic Transfer Matrices." *AIAA journal* 51.5 (2013): 1156-1173.

Generation of the state-space aerodynamics

The aerodynamic system is generated using the function

```
ssAeroModel = getAeroModel(options)
```

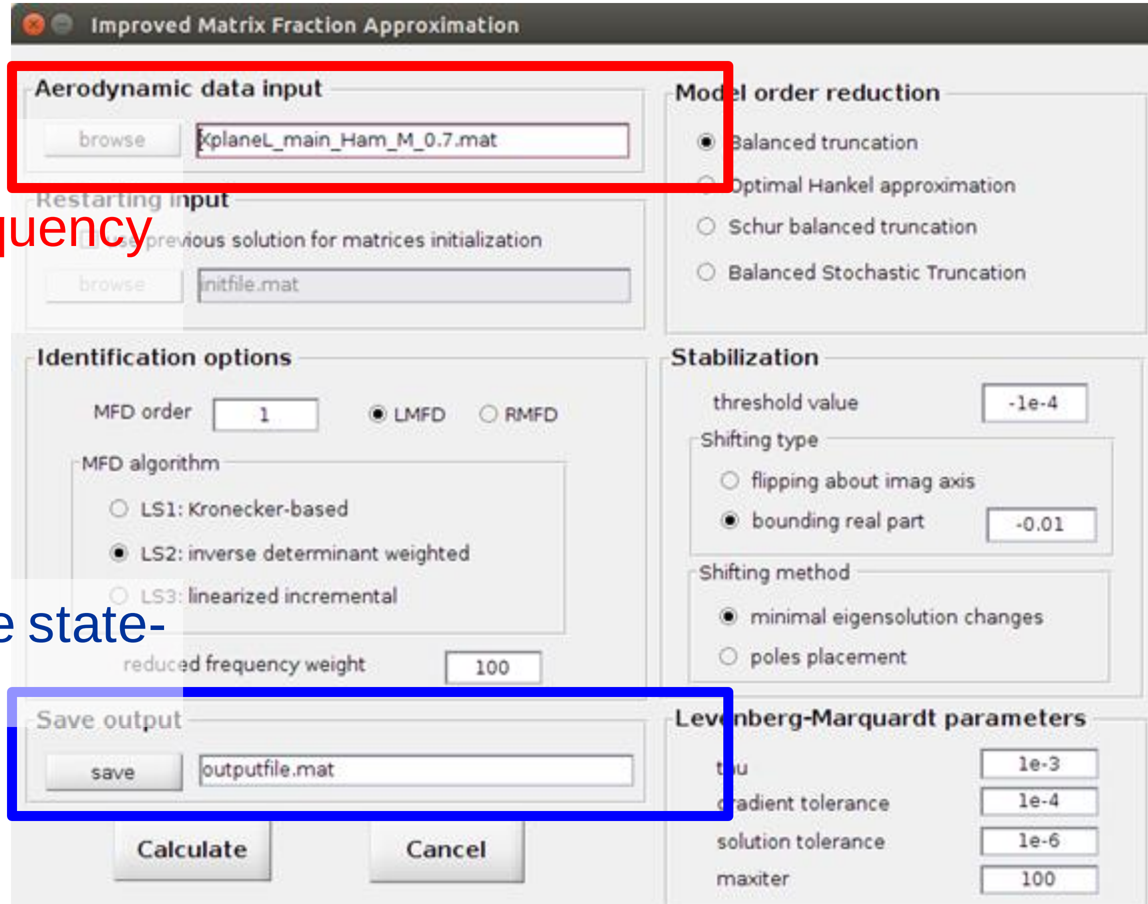
The options can be defined either programmatically (as name/value pair) or through the use of a gui. The programmatic method will be demonstarted in the example matlab script. For allowing the definition of the parameters the function must be called as

```
ssAeroModel = getAeroModel([], 'useGUI', true)
```

Not all the options available can be set using the GUI

GUI – input and output files

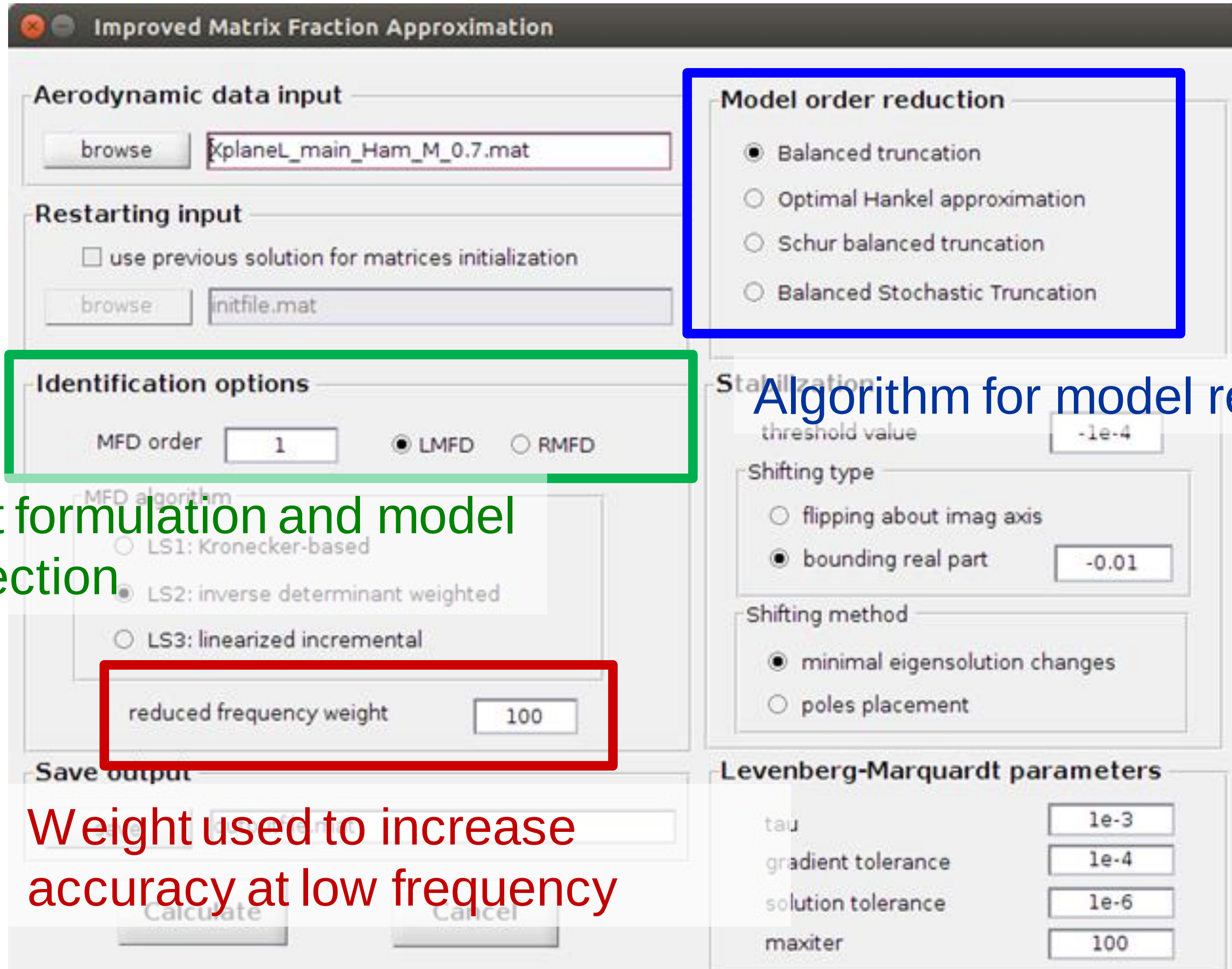
INPUT: .mat file with the aerodynamic matrix in frequency domain



OUTPUT: .mat file with the state-space model

The function `getAeroModel` takes care of defining the correct file names, the user should not select a file different from the default choice

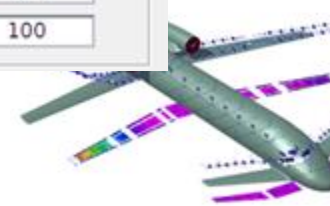
GUI – main parameters



Algorithm for model reduction

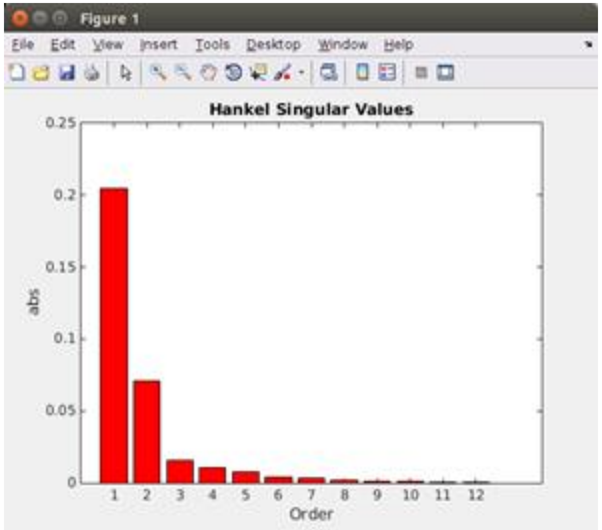
Right/Left formulation and model order selection

Weight used to increase accuracy at low frequency

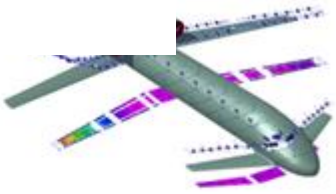


Model Reduction Phase

Increasing the order of the polynomial the accuracy of the state-space model increases, but can lead to a system with many states which have small effect in the input-output relationship. A model reduction procedure is applied to remove these states.




```
Check stability of state-space system: stable

(E,C) refitting before model order reduction:
---------------------------------------------------------------------------
error: 1.452168e-01

Model order reduction:
---------------------------------------------------------------------------
Please enter the desired order: (>=0)
```

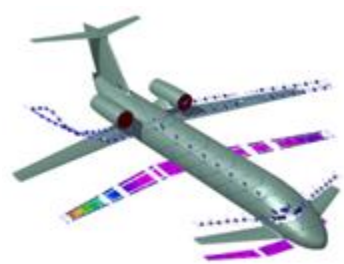


Options for SS model generation

Method: ‘mfd’ is the matrix fraction method, ‘qs’ is the quasi-steady approximation

```
>> setAeroSSoptions();
Available options for the ss model generation:
                    input   :  Input to aerodynamic system ('modal', 'gust', 'control')
                   output   :  Output from aerodynamic system ('modal')
                    selIn   :  Select among input variables
                    selOu   :  Select among output variables
                     mach   :  Selection of Mach number
                    kvect   :  Selection of reduced frequencies (interpolation performed)
                   method   :  Method for generation of the state space model ('mfd', 'qs')
                 mfdOrder   :  MFD order (2)
                   mfdAlg   :  MFD algorithm (2)
                  mfdSide   :  left or right MFD ('rmfd', 'lmfd') Used for reduced model
                mfdWeight   :  weight to enforce low frequency fitting (100) Used for reduced model
              mfdResOrder   :  order of residualization (2) automatically set to 2 with 'lmfd'
               mfdRollOff   :  roll off for discrete gust (2)  (not used)
                    LMtau   :  tau: starting value for LM damping
                LMgradTol   :  tolerance on gradient
                 LMsolTol   :  tolerance on solution variation
                LMmaxIter   :  Max num of iterations
                 eigThres   :  threshold value for the eigenvalues real part
                  eigMeth   :  method for enforce stability ('eigshift', 'polesplace')
                  eigType   :  target poles ('bound', 'flip')
                  igBound   :  bound value
                   algROM   :  Model reduction algorithms: 'balance', 'hankel', 'schur', 'bst'
                  rderROM   :  Order for model reduction, if empty it will be requested during execution
                   useGUI   :  Use the gui for identification
              restartFile   :  Name of file used for restart
                     kmin   :  Minimum reduced frequency in QS approximation
                     kmax   :  Maximum reduced frequency in QS approximation
```

Enable the GUI (and discard all the mfd*, LM* eig* options)



State-space model generation example

Smartcad input file (inputMain_ss.dat)

```
[...]
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
GUST= 1
SURFDEF= 1       2       3
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
[...]
```

All the surface and gusts are selected in order to have them in the SS model

State-space model generation example

Matlab script (inputMain_ss.dat) – aerodynamic system definition

```
global beam_model
global dyn_model

SSopt_j = setAeroSSoptions([], 'method', 'mfd', ...
                               'mfdOrder', 2, ...
                               'selOu', {2:20, []}, ...
                               'selIn', {2:20}, ...
                               'kvect', kvectInterp, ...
                               'mfdweight', 100, ...
                               'mfdside', 'rmfd');
SSopt_g = setAeroSSoptions([], 'method', 'mfd', 'mfdOrder', 8);
SSopt_c = setAeroSSoptions([], 'method', 'mfd');

ssmodel_j = getAeroModel(SSopt_j, 'input', {'j'}, 'output', {'j','h'});
ssmodel_g = getAeroModel(SSopt_g, 'input', {'g'}, 'output', {'j','h'}, 'selOu', {2:20, []});
ssmodel_c = getAeroModel(SSopt_c, 'input', {'c'}, 'output', {'j','h'}, 'selOu', {2:20, []});


ssmodelArray = ssmodel_j;
ssmodelArray(2) = ssmodel_g;
ssmodelArray(3) = ssmodel_c;
% Assembly total aero model
ssmodel = assemblySSmodel(ssmodelArray, {'j', 'g', 'c'}, {'j','h'});
```

Initialize options

Compute the SS aerodynamic model. One for each type of input (modal, gust, control surface)

Merge all the aerodynamic SS models in a syngle system

State-space model generation example

Matlab script (inputMain_ss.dat) – flutter diagrams

```
% Check flutter diagram
[FLeigI, VvectI] = solve_linflutt_cont('Vmin', 20, 'method', 'continuation', 'axesUsed',
'inertial', 'selTracked', [7:20]);
[FLeigB, VvectB] = solve_linflutt_cont('Vmin', 20, 'method', 'pointwise', 'axesUsed', axesUsed);

l_a = dyn_model.dlm.aero.cref;
Vvect = {VvectB, VvectI};
FLeig = {FLeigB, FLeigI};
figHandleList = plotFlutterDiagram(Vvect, FLeig, l_a);
figHandle = figHandleList;
figHandle = flutterDiagramSSCorrected(dyn_model, Vvect{1}, ssmodel, figHandle, 'xg', l_a, ...
                             'axesUsed',    axesUsed, ...
                             'seljUsed',    seljUsed);
```

Computation with frequency domain aerodynamic

Computation with SS aerodynamic model

The comparison between the flutter diagrams obtained using time domain and frequency domain aerodynamic forces can be used to eval the accuracy of the identification.

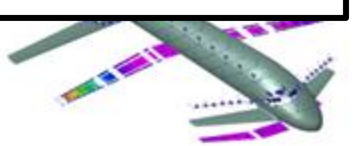
State-space model generation example

Matlab script (inputMain_ss.dat) – generate SS model

```
% Assembly total aeroelastic model

[ssAEmodel_corr, aero, ssmodel, descrForm] = getAEmodelCorrected(dyn_model, ssmodel, ...
                                        'axesUsed', axesUsed, ...
                                        'seljUsed', seljUsed);

[...]

Dt = 1e-3; Tfin = 10;
Agust = 17.07; fgust = 2.5;
% Constant time step
T = 0:Dt:Tfin;
nT = length(T);

[...]


[TsC, YsC] = ssTimeSimulation(ssAEmodel_corr.A, ssAEmodel_corr.B(:,pos_input_corr), ...
                              ssAEmodel_corr.C, ssAEmodel_corr.D(:,pos_input_corr),
                              ag_tot, [], 0, Tfin, Dt, 'CN');
```

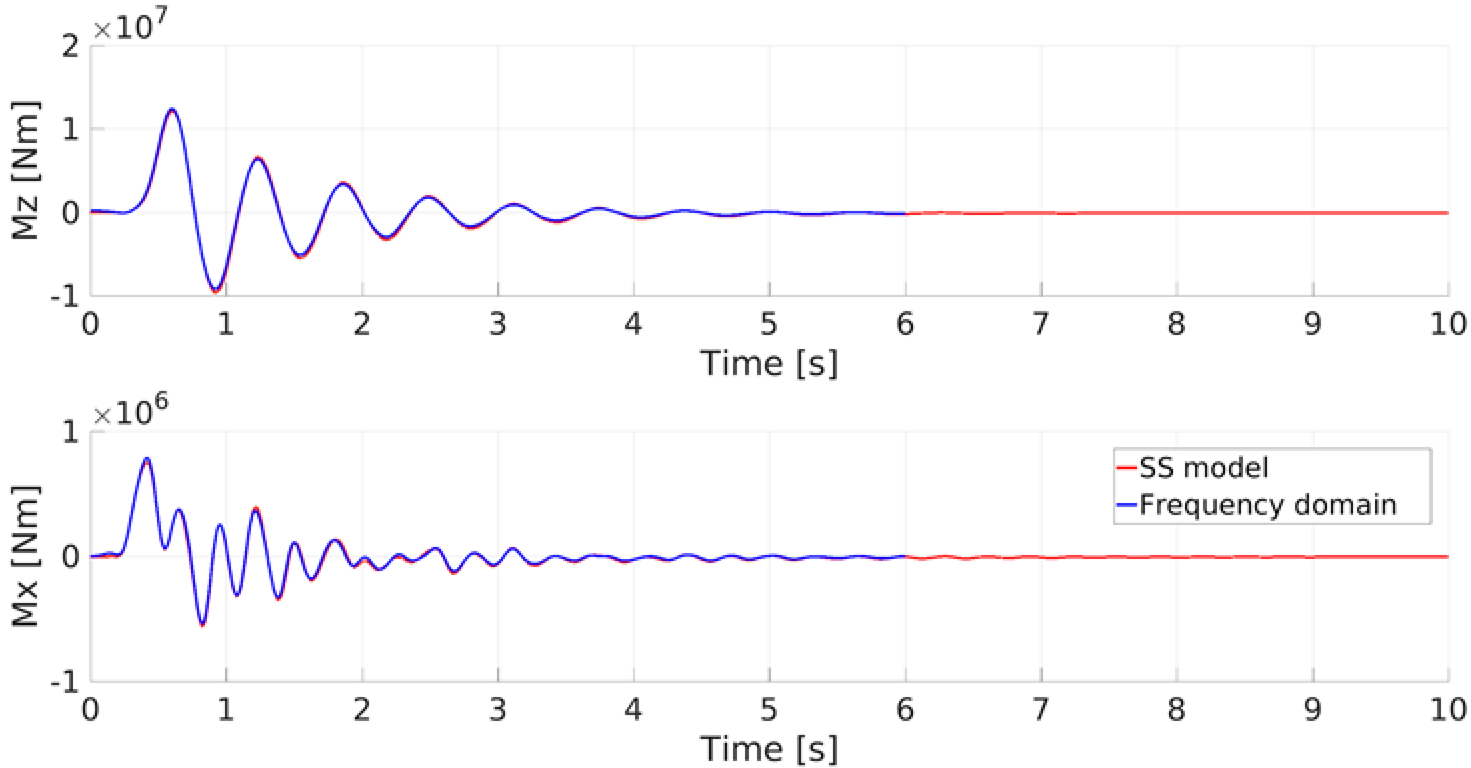
Name/value pair for defining flight condition, modal base, type of output …

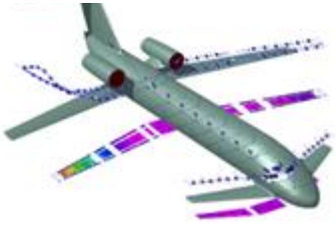
Time simulation



State-space model generation example

Comparison of results obtained from time-domain and frequency domain





Aerodynamic mode correction

Among the options that can be set in the generation of the SS model there is also the possibility to introduce a set of aerodynamic coefficients to correct the steady portion of the aerodynamic forces.

```
[ssAEmodel_corr, aero, ssmodel, descrForm] = getAEmodelCorrected(dyn_model, ssmodel, ...
                                             'aeroCoef', aeroCoef, ...
                                             'axesUsed', axesUsed, ...
                                             'seljUsed', seljUsed);
```

This capability is only described, but it is not included in the tutorial scripts.

Model output

The generated model is in a format compatible with the matlab SS class

```
ssAEmodel_corr =
              A: [40x40 double]
              B: [40x13 double]
              C: [4085x40 double]
              D: [4085x13 double]
     inputGroup: [1x1 struct]
    outputGroup: [1x1 struct]
      inputName: {1x13 cell}
     outputName: {1x4085 cell}
     inputDelay: [13x1 double]
```

The model in the matlab SS class can be obtained using the function convertToSS_neo

```
ssClassModel = convertToSS_neo(ssAEmodel_corr);
```

The system outputs are divided in grups

```
>> ssAEmodel_corr.outputGroup
        accel: [1 2 3 4 5 6 7 8 9 10 11 12]
    modalDisp: []
     modalVel: []
        displ: [1x3072 double]
          vel: [1x12 double]
     barforce: [1x984 double]
    beamforce: []
   aeroangles: []
   modalForce: []
      hingemom: [4081 4082 4083 4084 4085]
```

To each output a name is assigned

```
>> ssAEmodel_corr.outputName{ssAEmodel_corr.outputGroup.barforce(1)}
ans =
FORCE-BAR-1000-1-C1

>> ssAEmodel_corr.outputName{ssAEmodel_corr.outputGroup.displ(1)}
ans =
DISP-1000-C1

>> ssAEmodel_corr.outputName{ssAEmodel_corr.outputGroup.hingemom(1)}
ans =
HMOM-aileronr
```

Definition of aerodynamic coefficients

```
aeroCoef.system = 'stability';
aeroCoef.aeroType = 'aero';
aeroCoef.inputType = 'angles';
aeroCoef.refPoint = [13.463, 0, 1.8275];
aeroCoef.cref = 2.5642;
aeroCoef.bref = 29.654;
aeroCoef.Sref = 73.276;

aeroCoef.Mach = [0.3, 0.52];
```

- Reference system used for the definition of coefficients
- Reference point for the definition of aerodynamic moments
- Reference dimensions for nondimensionalization
- Mach number used for the definition of the coefficients

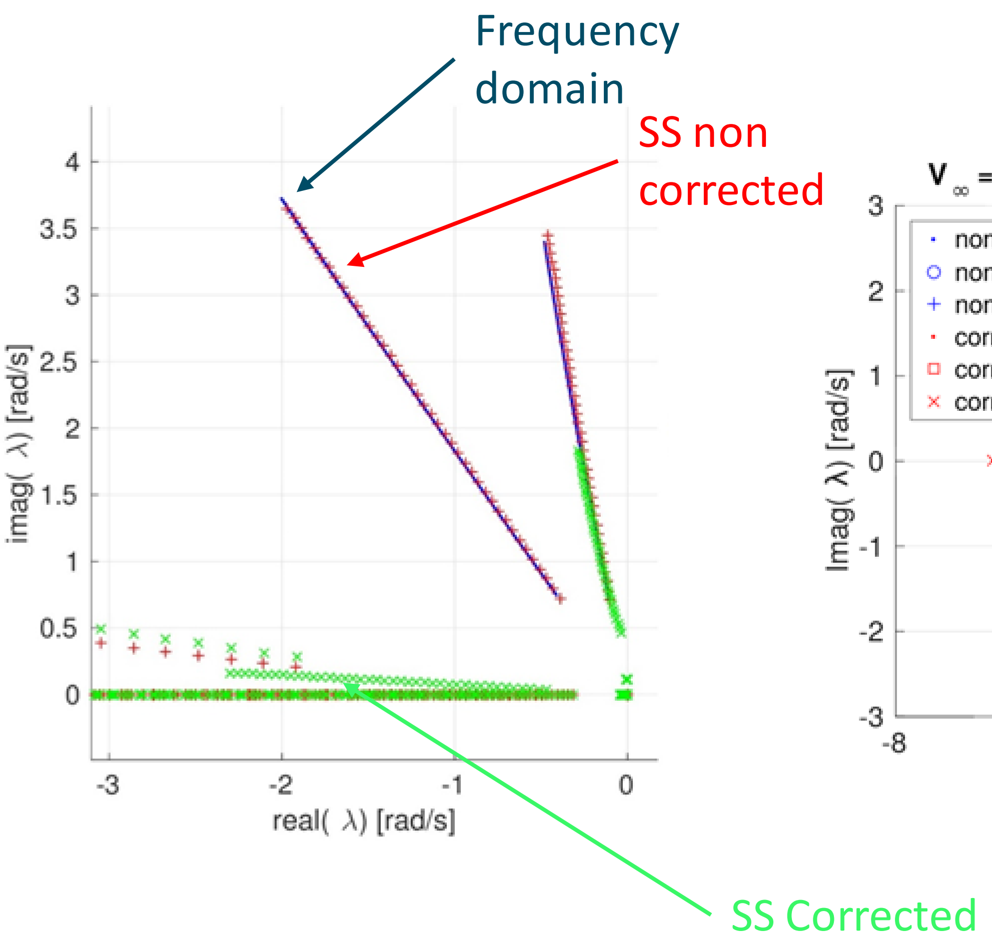
```
aeroCoef.F0(:,:,2) = [
                       0.0379;
                            0;
                        0.337;
                            0;
                         0.06;
                            0];
```

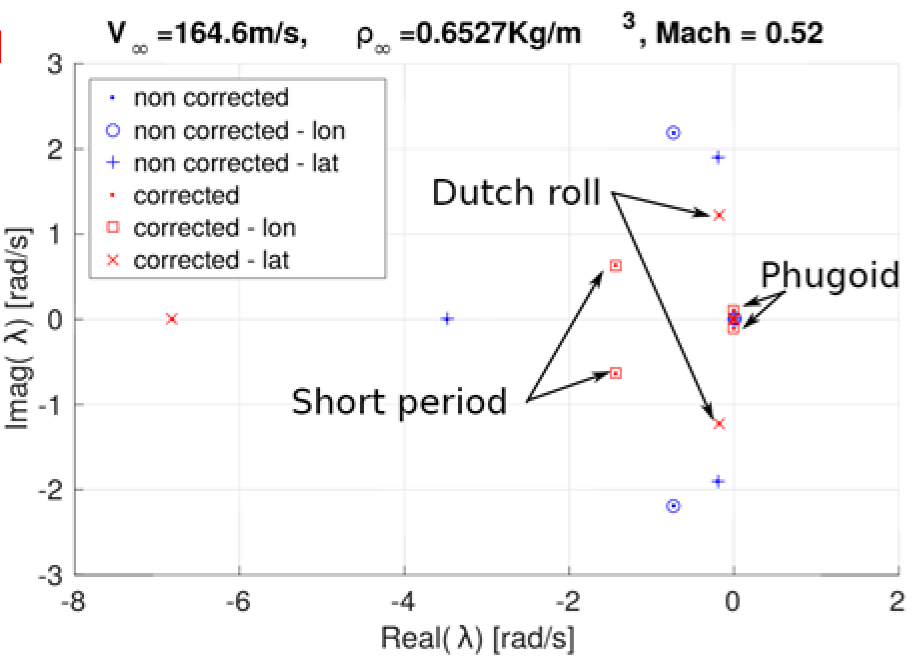
Forces at reference condition

```
aeroCoef.C0 = zeros(6,6,2);
aeroCoef.C0(:,2:6,1) = [
             0,  0.0382,       0,  0.0861,      0;
          1.17,       0,   -0.06,       0,  0.666;
             0,   6.486,       0,    14.6,      0;
        -0.176,       0,  -0.594,       0,  0.123;
             0,  -1.534,       0,  -70.48,      0;
         0.106,       0,  -0.052,       0, -0.231];

aeroCoef.C0(:,2:6,2) = [
             0,  0.0471,       0,  0.0959,      0;
          1.20,       0,   -0.06,       0,  0.666;
             0,   7.173,       0,    14.6,      0;
        -0.180,       0,  -0.625,       0,  0.113;
             0,   -0.15,       0,  -68.53,      0;
         0.115,       0,  -0.042,       0, -0.244];
```

Aero coefficients for each Mach number

Example of results (on a different model)





Thanks for your attention