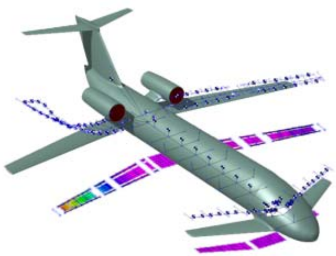
NeoRESP Tutorial

Solving dynamic response for free a/c
“frequency and time domain”

Version 2.2(.790)

August 2017

Outline



Solving **dynamic response** for free aircraft

Input Files:

- *XplaneL_neo.dat*: NEORESP main file
- *XplaneLCONM_CONF3.inc*: aeroelastic model
- *ForceSolverParam.inc*: settings for external force response
- *dyn_model_res.mat*: MATLAB bynary file with results from NEORESP

Steps:

1) Run the preprocessor:

   init_dyn_model('XplaneL_neo.dat')

2) Save the database:

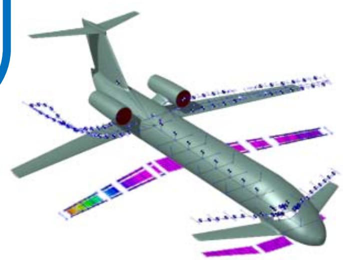
   global *dyn_model;*

   save('XplaneL_neo_neoresp.mat', 'dyn_model');

3) Run the simulation:

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

The response will refer to a time-window of *5 sec (T)*, sampled at *5e-3 secs (dT)*.

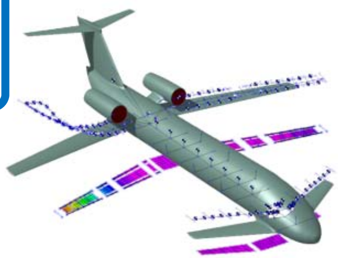


Solving **dynamic response** for free aircraft

In control solver parameter, one have to define also the control surface input function:

```
$-------2-------3-------4-------5-------6-------7-------8---
$ Control input
$-------2-------3-------4-------5-------6-------7-------8---
$       ID      label   Amplit  Period  Delay
$                       degrees sec     sec
SURFDEF 1       elev1r  1.0     0.5     0.0
        sin(2*pi/0.5*t)
```

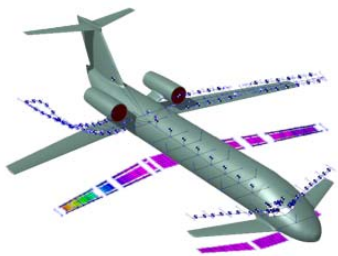
In this case it is sinusoidal, however one could define another shape of the control surface input as long as for gust and nodal forces.



Solving **dynamic response** for free aircraft

In the same file one have to define also other initialization parameters like: frequancy range to be analysed and how many and which frequencies have to be tracked for flutter analysis. Example:

```
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
$ Modes to follow in flutter solution
$-------2-------3-------4-------5-------6-------7-------8-------9-------10
FMODES        7       8       9       10      11      12      13      14
        15      16      17      18      19      20      21      22
        23      24      25      26      27      28
```



Solving **dynamic response** for free aircraft

Steps:

1) run the preprocessor:

__init_dyn_model__('XplaneL_neo.dat')

Created files:

- *XplaneL_neo_M0.700.aer*
- *XplaneL_neo_M0.700.baer*
- *XplaneL_neo.bmod*

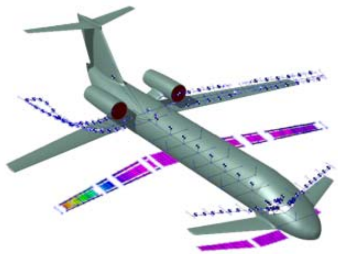
2) save the database:

__global__ dyn_model;

__save(__'XplaneL_neo_neoresp.mat', 'dyn_model');

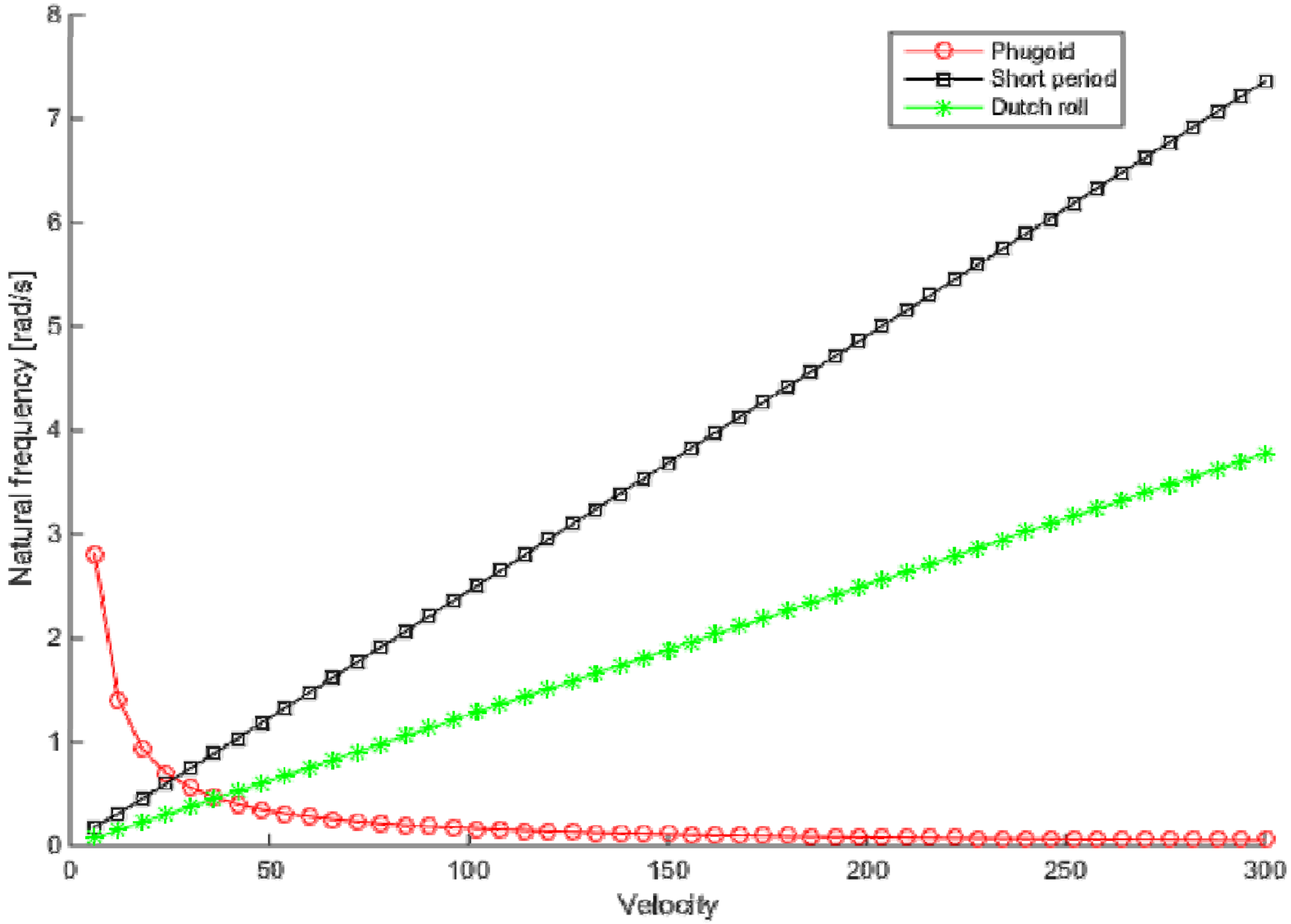
The model contains:

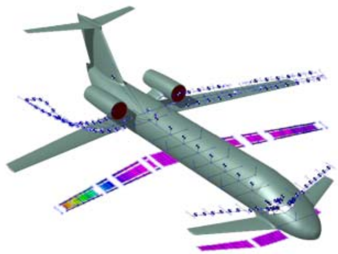
- *Dlm (aerodynamics)*
- *Beam (structures)*
- *Flu (flutter)*
- *Out (principal output)*
- *Res*



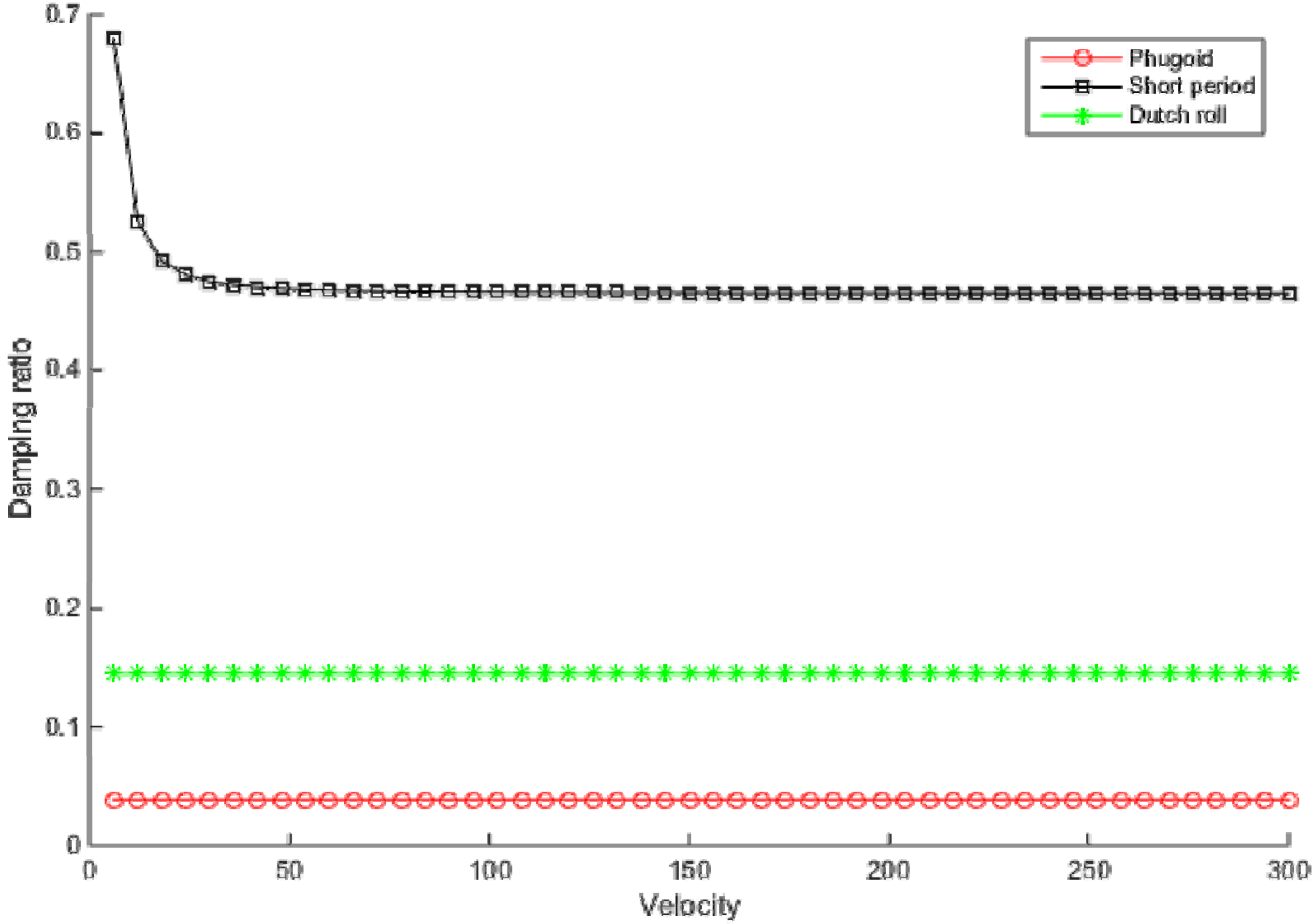
Solving **dynamic response** for free aircraft

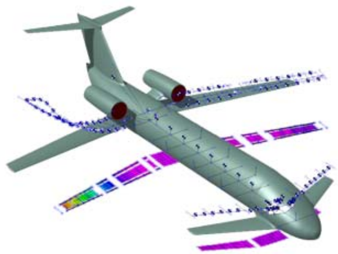
Below some output on init_dyn_model



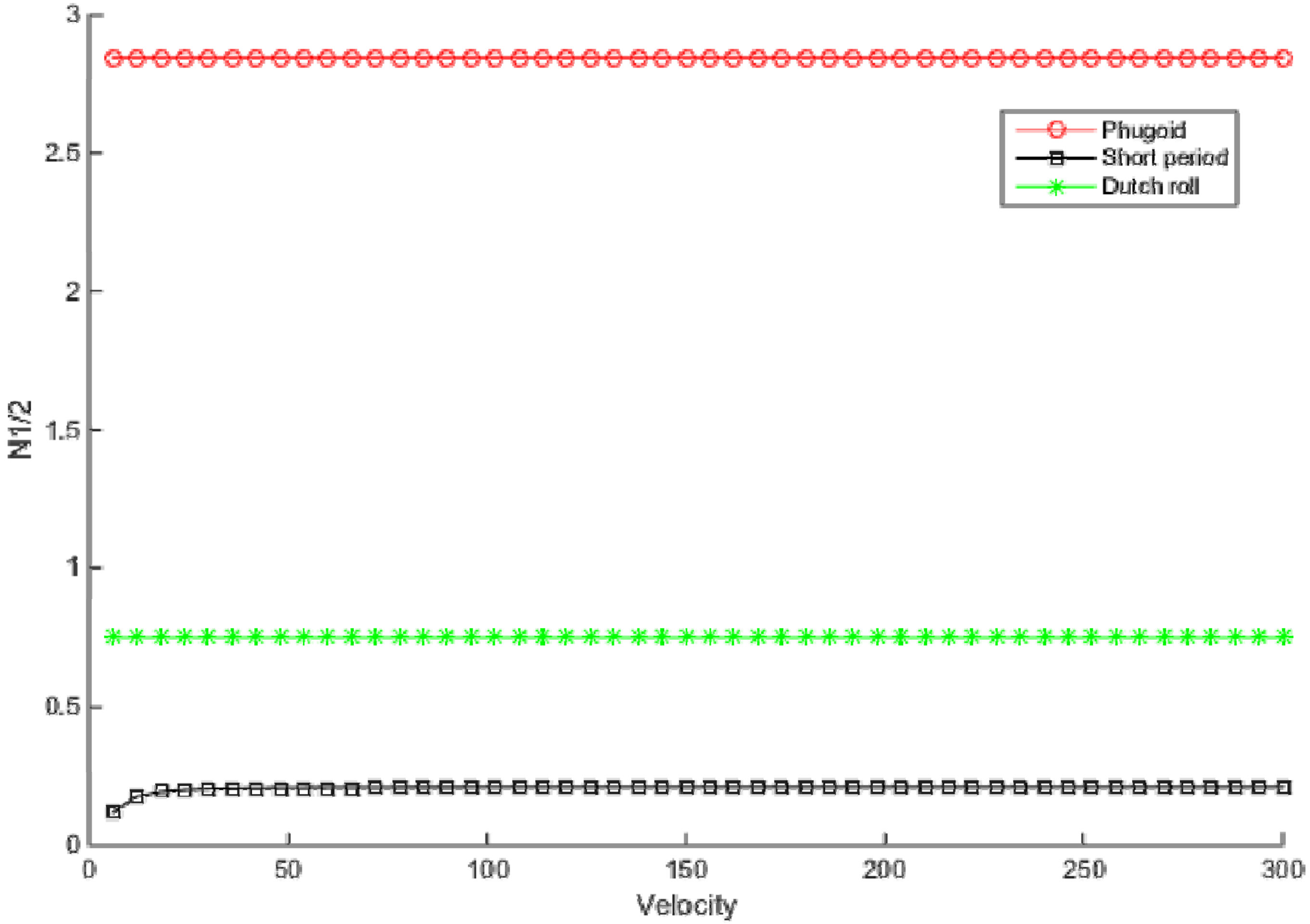


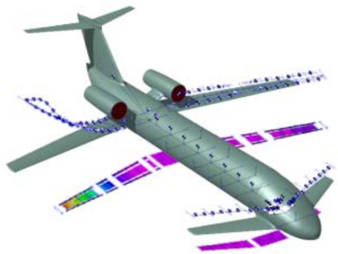
Solving **dynamic response** for free aircraft



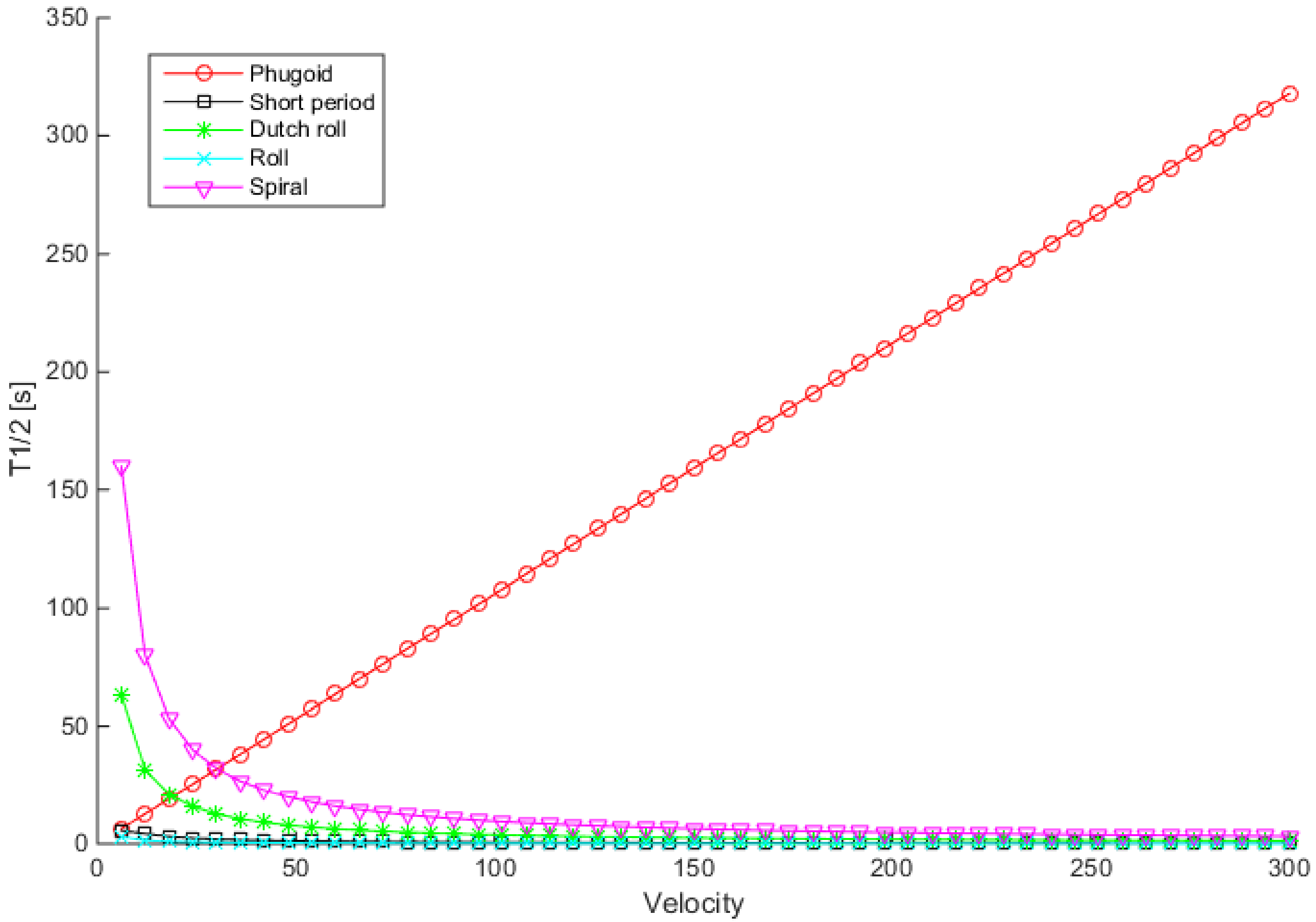


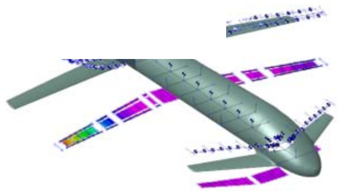
Solving **dynamic response** for free aircraft



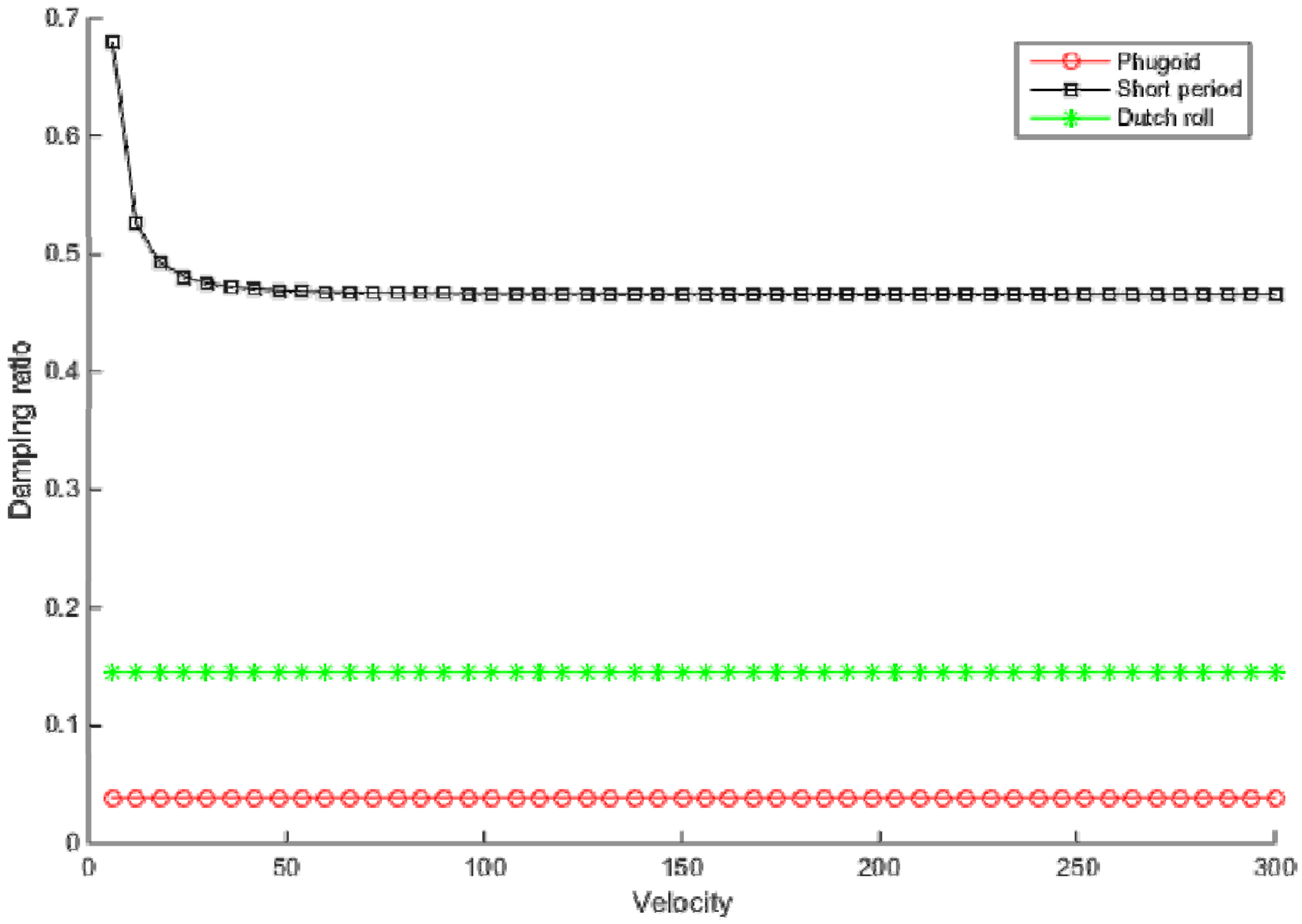


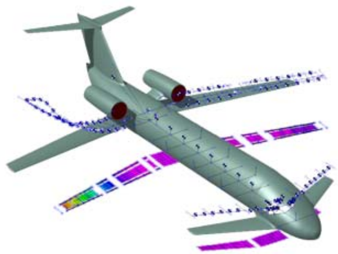
Solving **dynamic response** for free aircraft



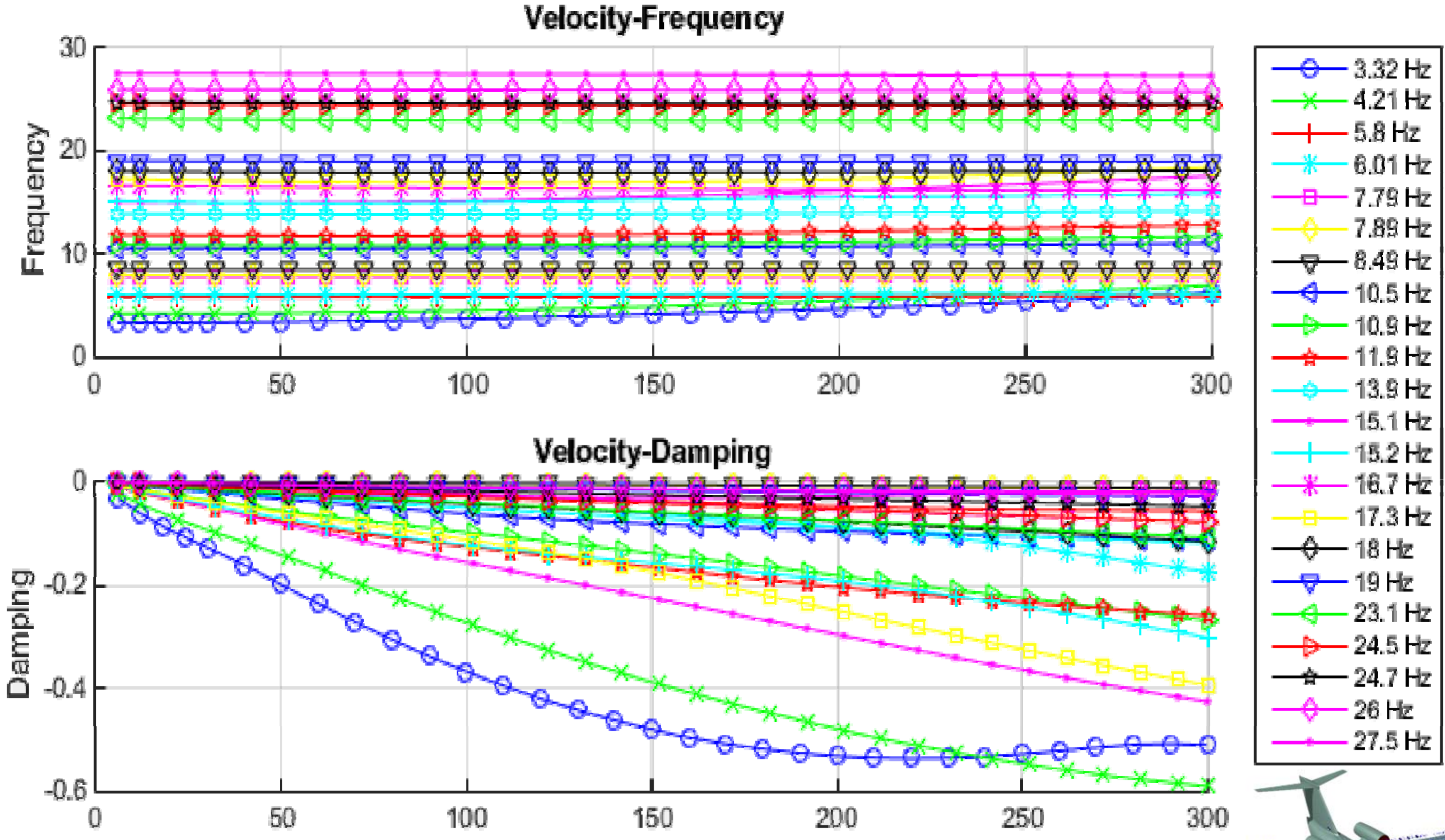


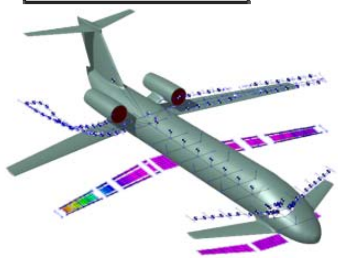
Solving **dynamic response** for free aircraft





Solving **dynamic response** for free aircraft





Solving **dynamic response** for free aircraft

3) After the process of the solver *solve_free_lin_dyn* one could find the results, loading *dyn_model,* into the substructures ***.Out** and **.Res***

.Res. :

Control_profile: [1x1001 double]

Time: [1x1001 double]

Qload: [28x1001 double]

Qextf: [28x1001 double]

Q: [28x1001 double]

Qd: [28x1001 double]

Qddot: [28x1001 double]

Cy_mode: [1x1001 double]

Cz_mode: [1x1001 double]

Cl_mode: [1x1001 double]

Cm_mode: [1x1001 double]

Cn_mode: [1x1001 double]

DISP: [524x6x1001 double]

ACCELERATION:[1x6x1001 double]

IFORCE: [4-D double]

CP_surf: [413x1001 double]

CP_mode: [413x1001 double]

HF_surf: [5x1001 double]

HF_mode: [5x1001 double]

Cy_surf: [1x1001 double]

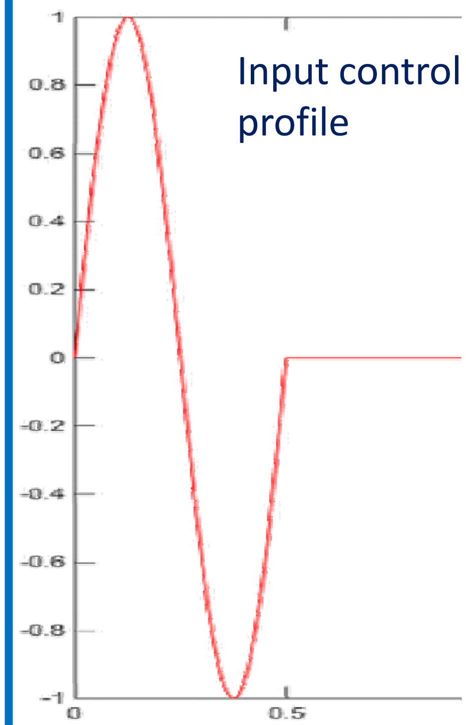
Cz_surf: [1x1001 double]

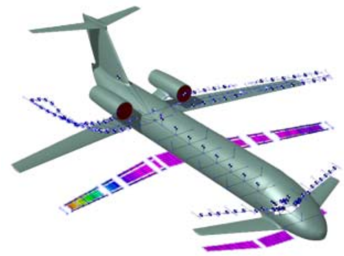
Cl_surf: [1x1001 double]

Cm_surf: [1x1001 double]

Cn_surf: [1x1001 double]

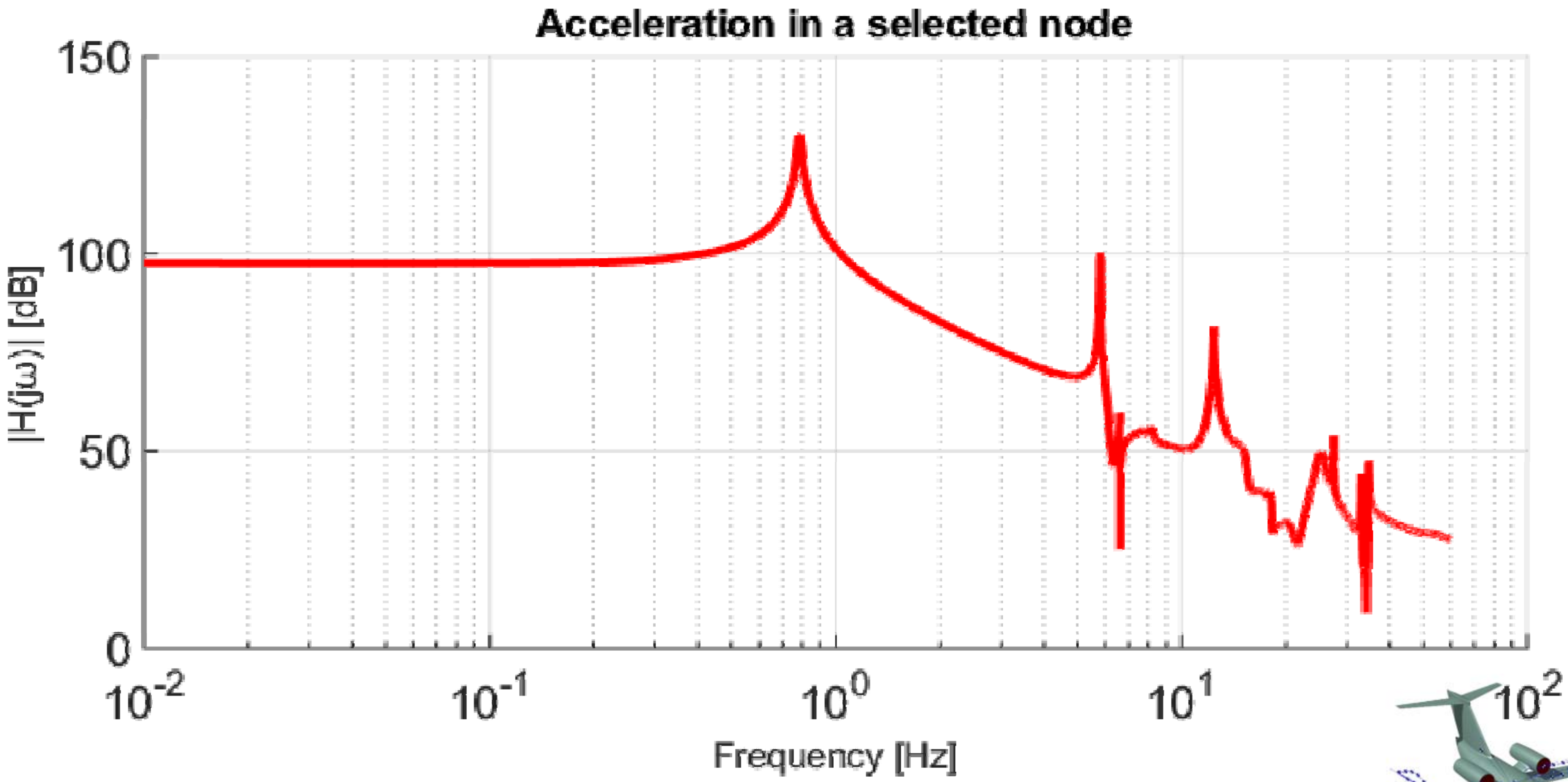
MODACC: [1x1 struct] (IFORCE)

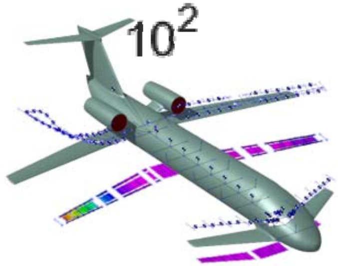




Solving **dynamic response** for free aircraft

Postprocessing values contained in *dyn_model.Res.IFORCE* one could obtain the Bode diagram that highlights the resonance frequencies.





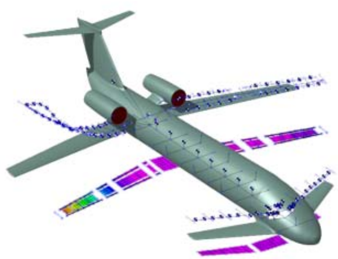
Solving **dynamic response** for free aircraft

The same would be done for gust input and nodal forces input cases. On have only to change the input CARD in the Input Parameters file.

```
$-------2-------3-------4-------5-------6--------
$ Select gust
$-------2-------3-------4-------5-------6--------
GUST= 1
$-------2-------3-------4-------5-------6--------
$ Gust input
$-------2-------3-------4-------5-------6--------
GUST    1      17.07   0.12    0.0     3
        1
        (1-cos(2*pi/0.1200*t))*0.5
```

```
$-------2-------3-------4-------5-------6-------7-------8-------9--------
$ Select control
$-------2-------3-------4-------5-------6-------7-------8-------9--------
LOAD=   1
$-------2-------3-------4-------5-------6-------7-------8-------9--------
$ Force input
$-------2-------3-------4-------5-------6-------7-------8-------9--------
$       ID      NODE    DOF     AMPLIT  TMAX    DELAY
DLOAD   1       2000    3       1.0e+3  0.5     0.0
        sin(2*pi/0.5*t)
```

As one could see, there is the possibility to define a delay of actuation for all the input.



Solving **dynamic response in time domain**

In this case the aerodynamic model is represented as state space system. The first two steps are the same as before. Note: This is applicable to either control or gust or force input.

Steps:

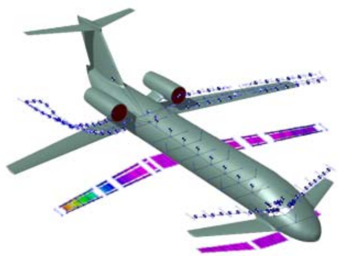
1) Launch NeoRESP **preprocessor**:

   init_dyn_model('XplaneL_neo.dat')

2) **Save the database**:

   global dyn_model;
   save('XplaneL_neo.mat', 'dyn_model');

3) Launch Neoresp for **state space analysis**: *solve_free_lin_dyn_ss*
   Two files with *generalized forces* will be created:
   - one for **Qam** due to motion: *XplaneL_neo_Ham_M_0.7.mat;*
   - one for **Qad** due to elevator: *XplaneL_neo_Had_M_0.7.mat.*



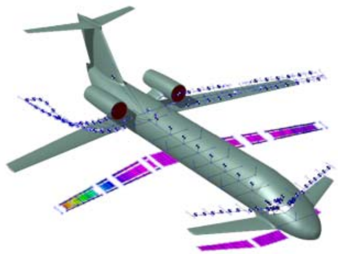
Solving **dynamic response in time domain**

This will be the output in the command window:

- Aerodynamic matrix Ham exported to XplaneL_neo_Ham_M_0.7.mat file for fitting.
  Rows: 12.
  Columns: 12.
  Extra outputs: 10.

- Aerodynamic matrix Had exported to XplaneL_neo_Had_M_0.7.mat file for fitting.
  Rows: 12.
  Columns: 1.
  Extra outputs: 10.

Then, for each H matrix one have to start the fitting process using the function ***aero_ss.***



Solving **dynamic response in time domain**

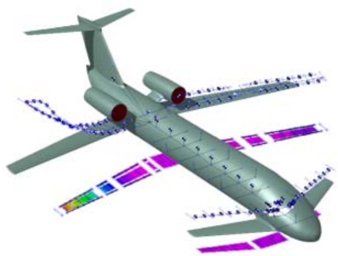
Open the script ***aero_ss.m*** and look at the parameters used for fitting.
Considering Ham we have:

```
opt{1} = 3;          % MFD order
opt{2} = 3;          % MFD algorithm
opt{3} = 'lmfd';     % left or right MFD
opt{4} = 2;          % residualization order
opt{5} = 100;        % weight parameter value W^2
```

This values could be changed as far as the user thinks it is useful.

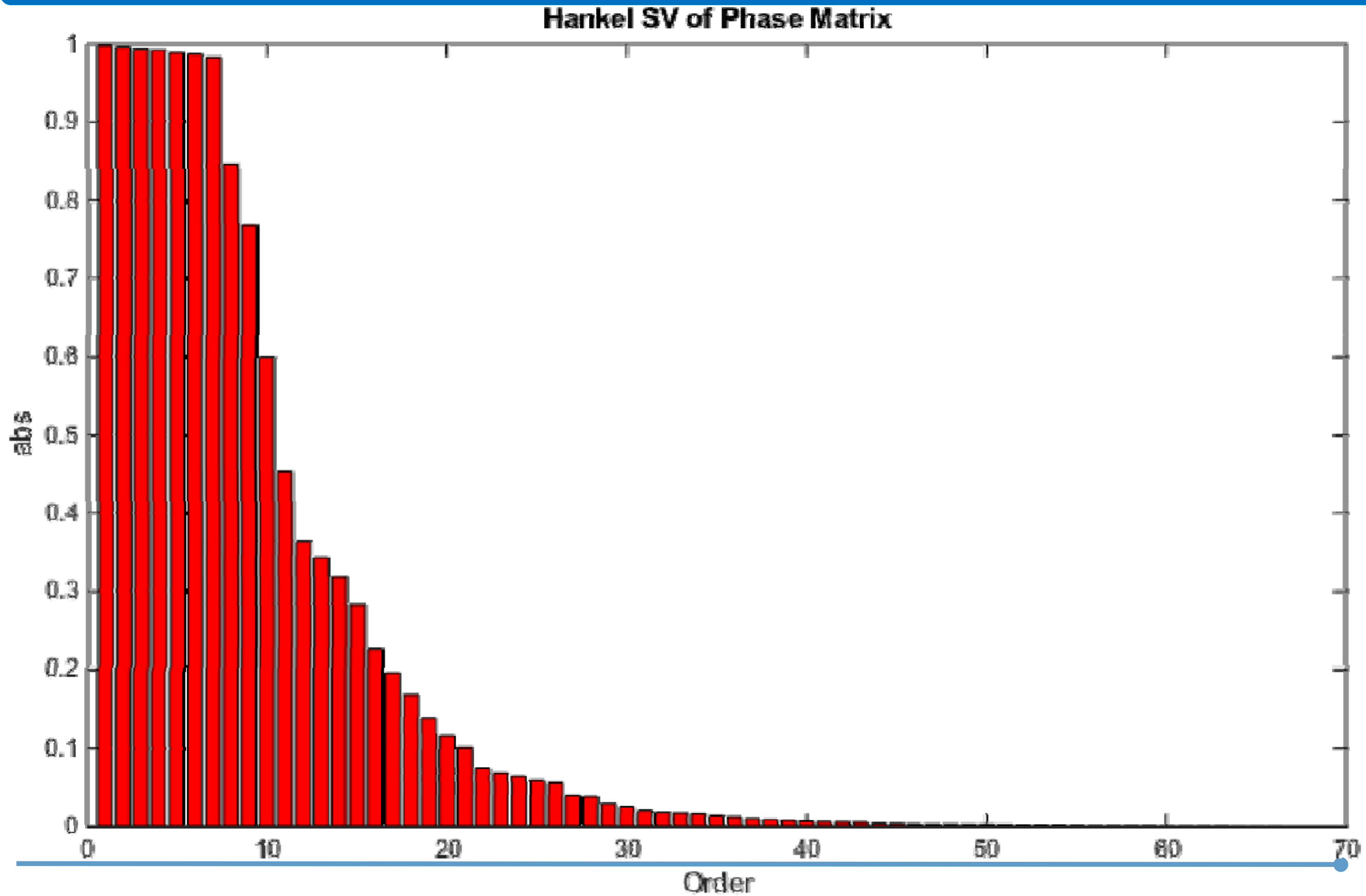
Run the script for Ham:

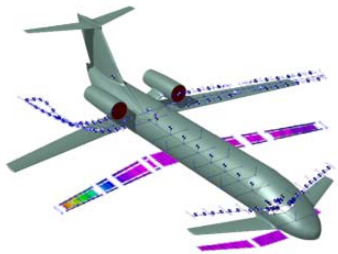
aero_ss('XplaneL_neo_Ham_M_0.7.mat','res1.mat')



Solving **dynamic response in time domain**

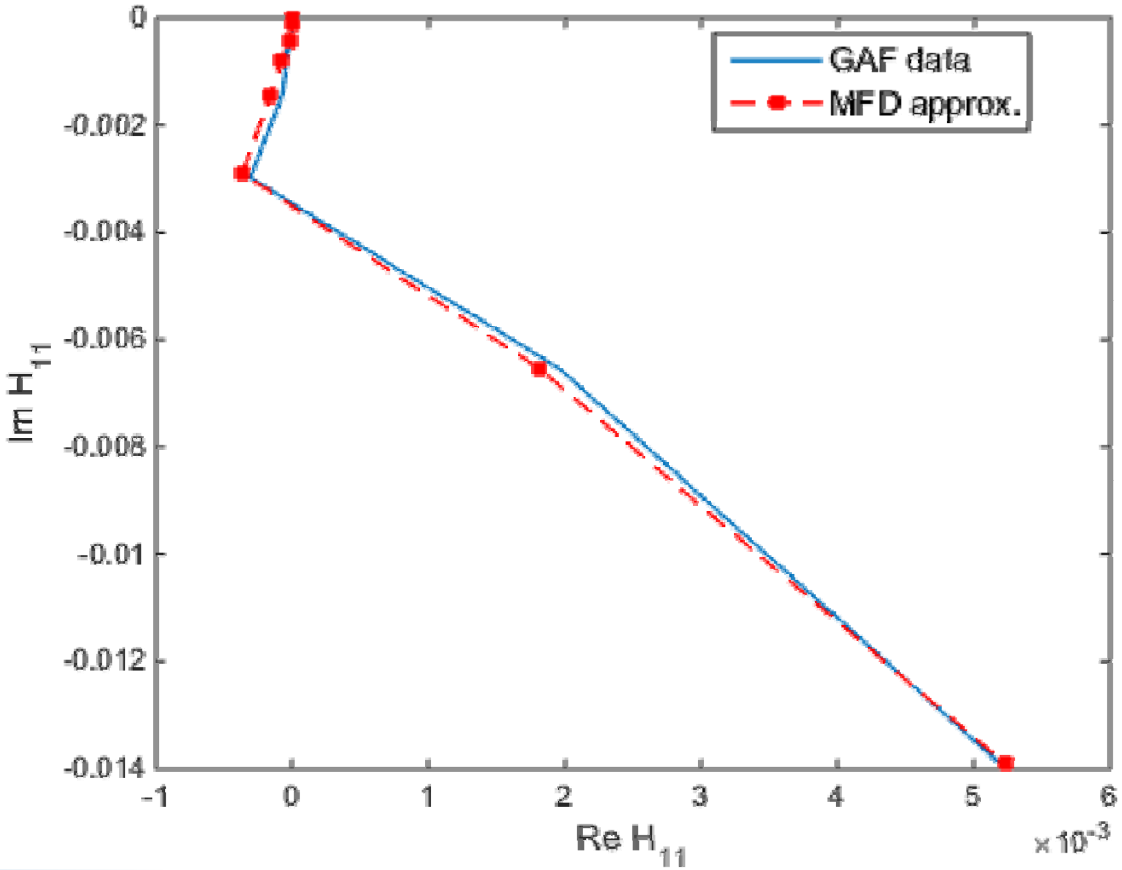
After choosing the fitting order for Ham, check for the quality of the interpolation by looking into the interpolated terms by plotting the fitting results. (Ex. Order = 20)

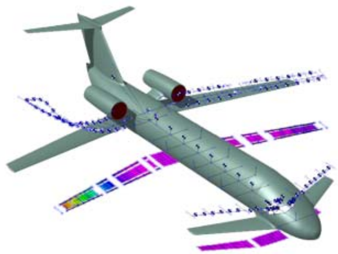




Solving **dynamic response in time domain**

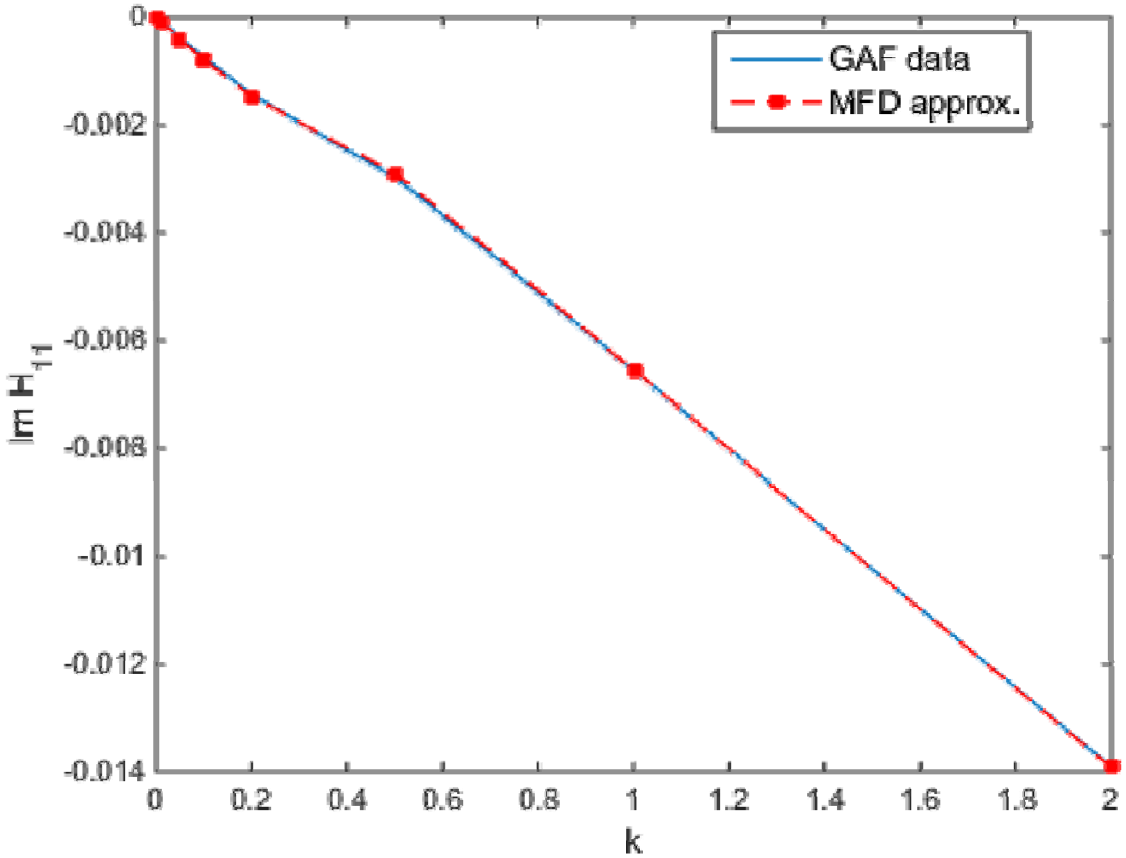
Ham(1,1) in complex domain

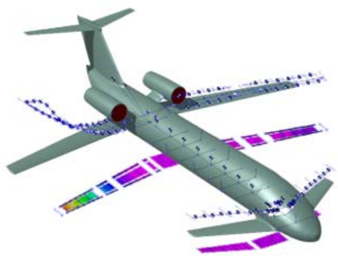




Solving **dynamic response in time domain**

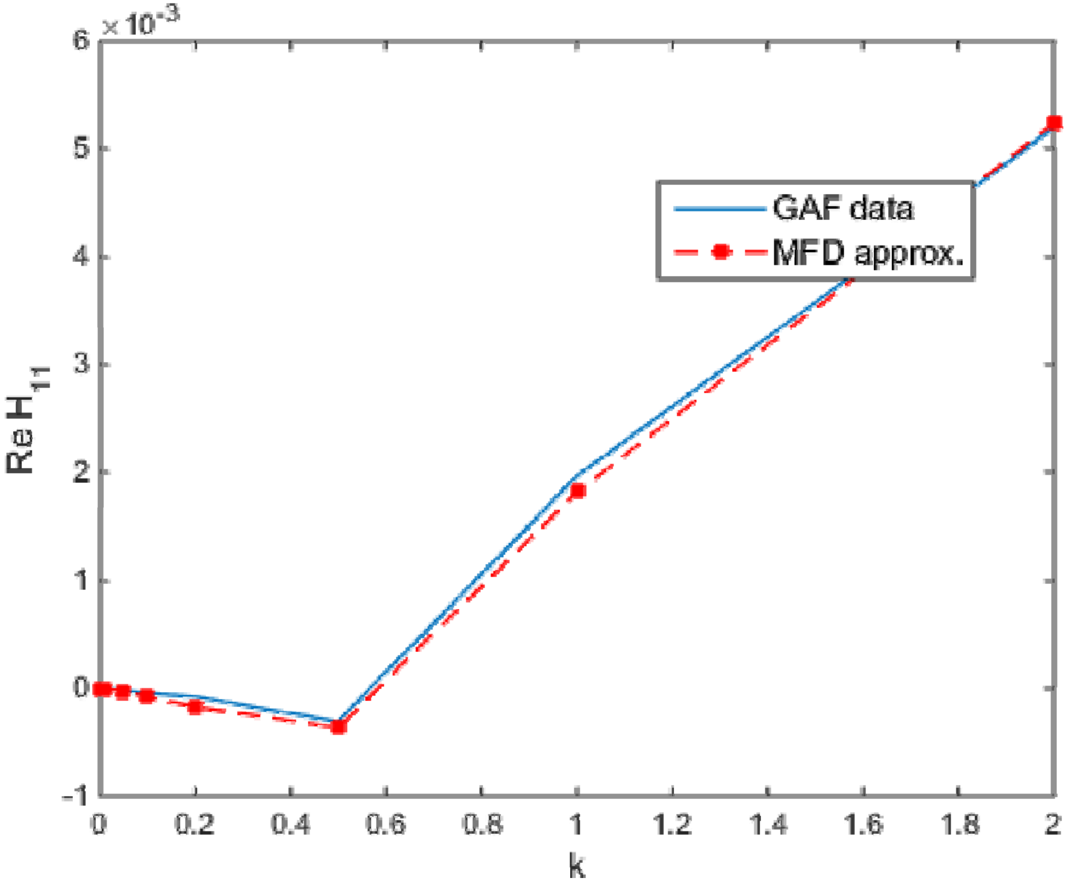
Imaginary part function of k parameter for Ham(1,1)

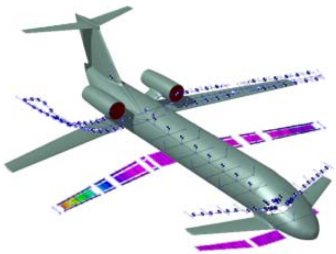




Solving **dynamic response in time domain**

Imaginary part function of k parameter for Had(1,1)





Solving **dynamic response in time domain**

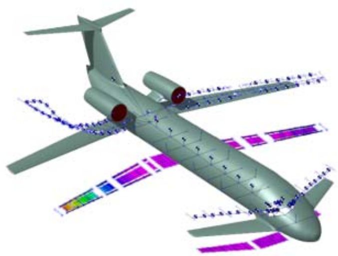
Open again the script ***aero_ss.m*** and look at the parameters used for fitting. Considering Had we have:

```
opt{1} = 4;         % MFD order
opt{2} = 3;         % MFD algorithm
opt{3} = 'rmfd';    % left or right MFD
opt{4} = 2;         % residualization order
opt{5} = 100;       % weight parameter value W^2
```

This values could be changed as far as the user thinks it is useful.

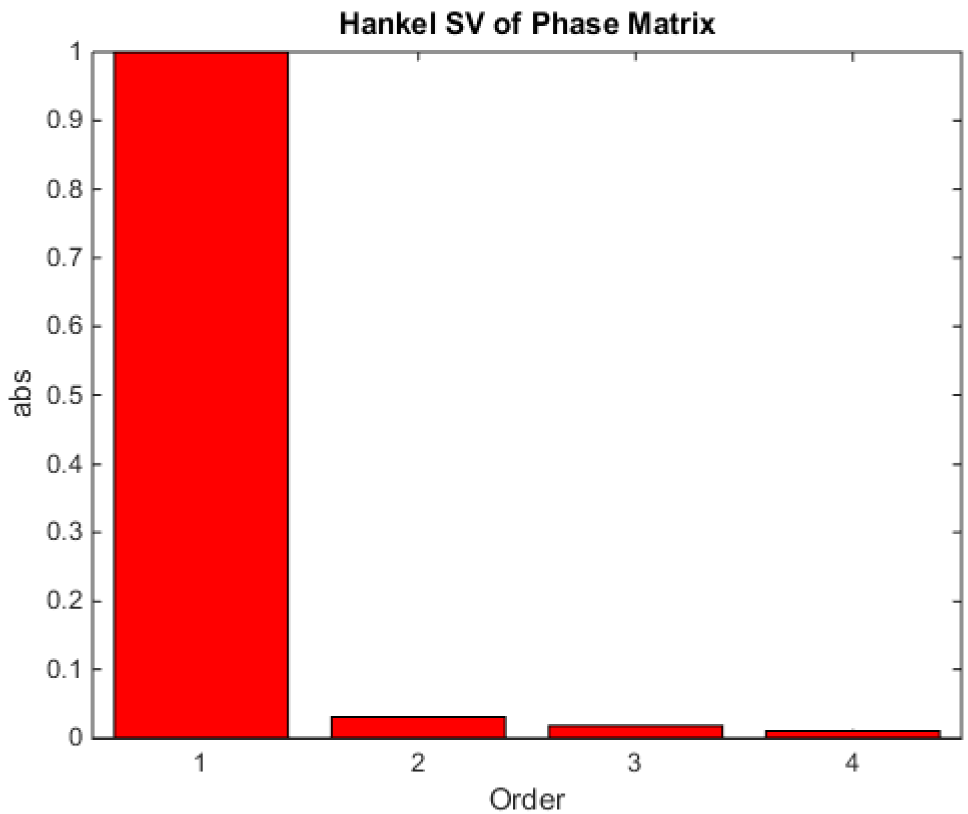
Run the script for Ham:

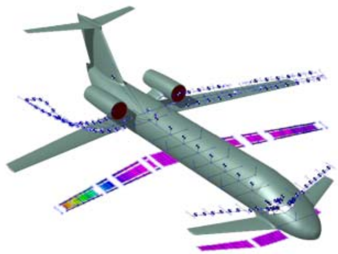
aero_ss('XplaneL_neo_Had_M_0.7.mat','res2.mat')



Solving **dynamic response in time domain**

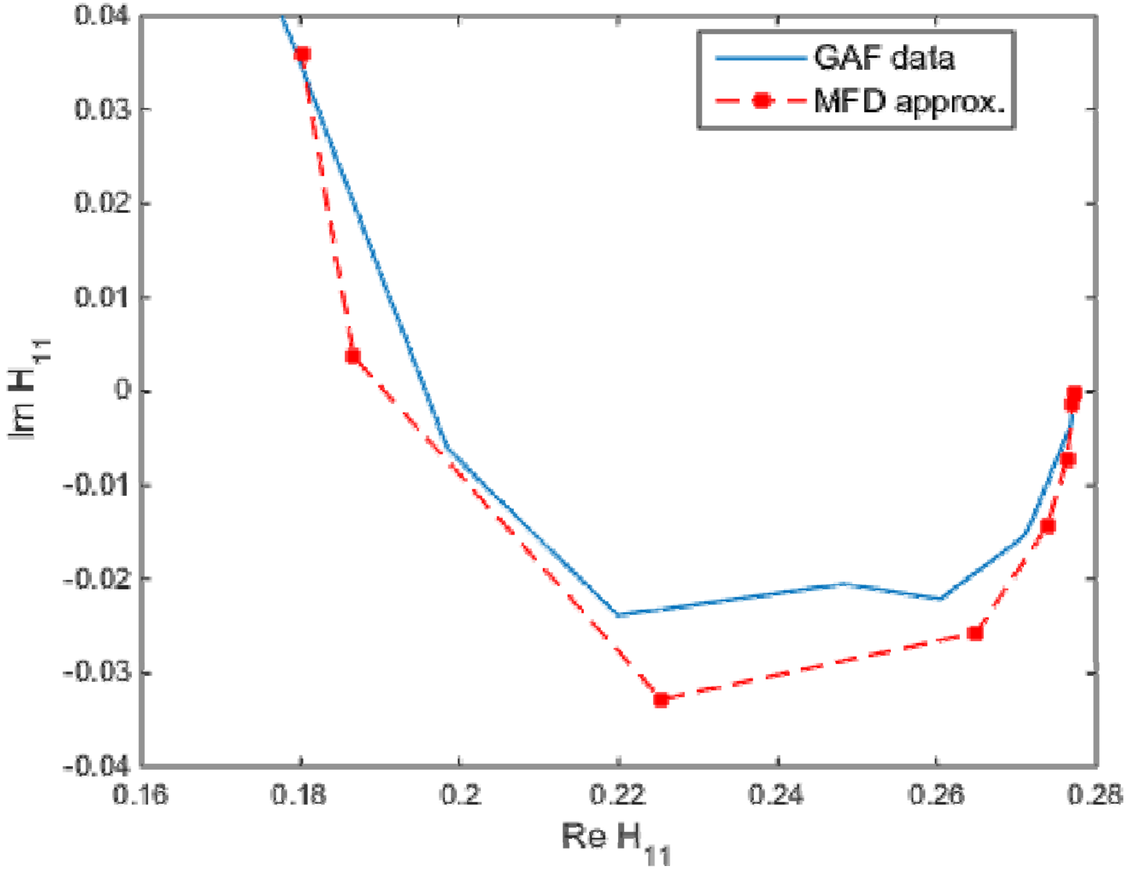
After choosing the fitting order for Had, check for the quality of the interpolation by looking into the interpolated terms by plotting the fitting results. (Ex. Order = 2)

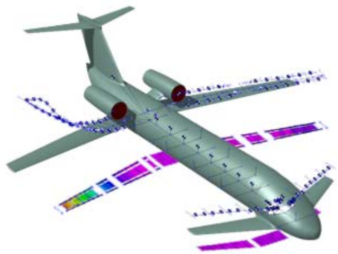




Solving **dynamic response in time domain**

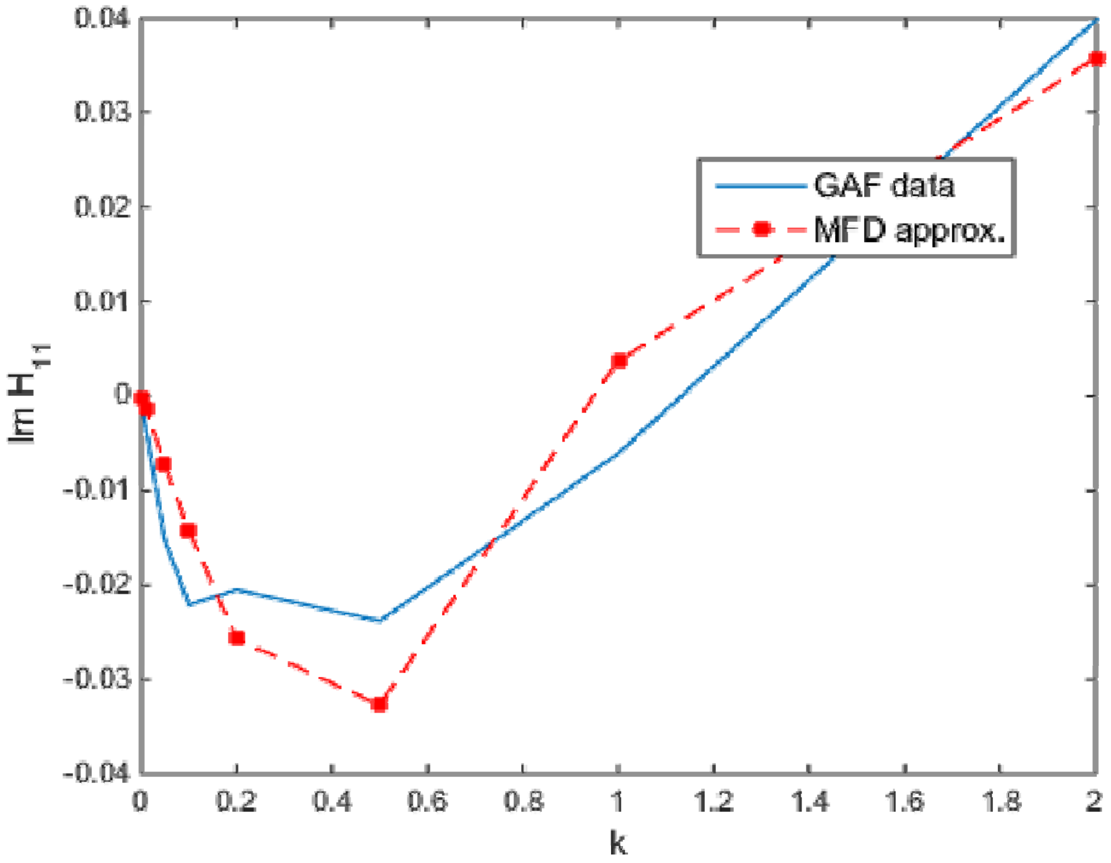
Had(1,1) in complex domain

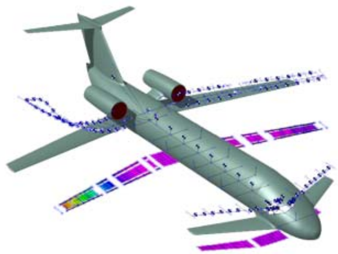




Solving **dynamic response in time domain**

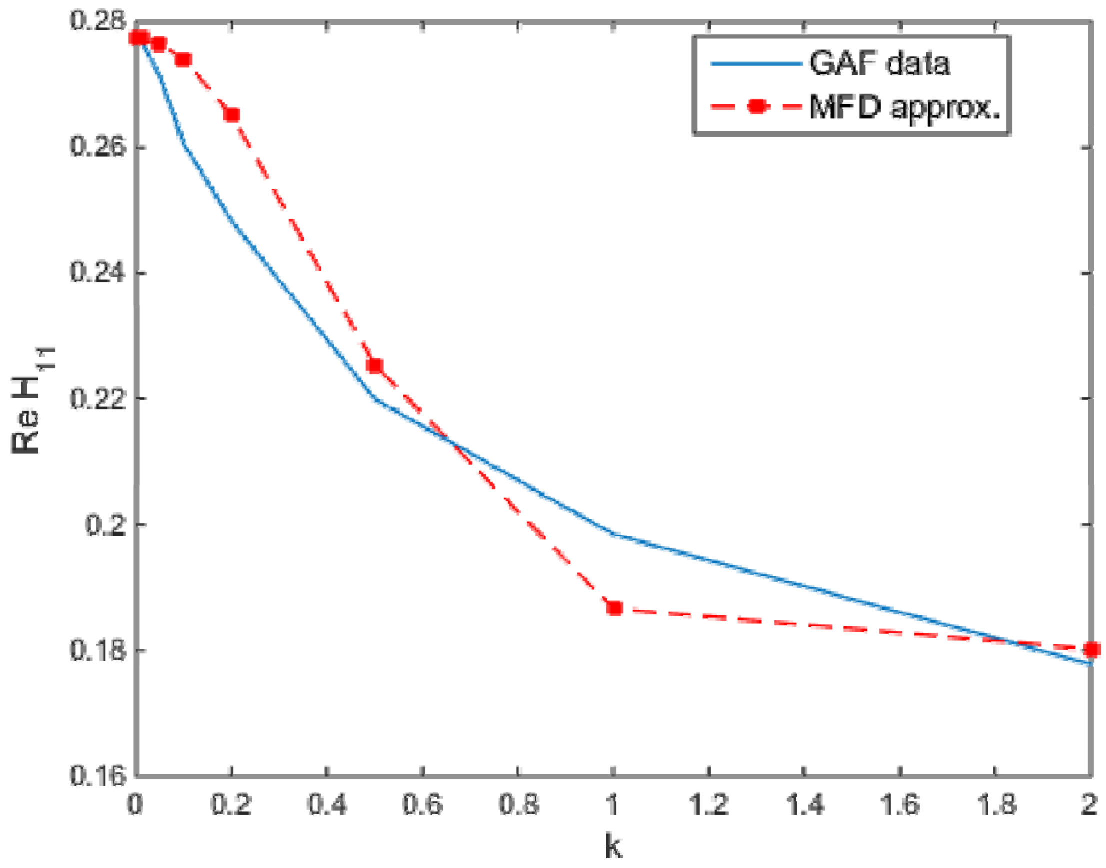
Imaginary part function of k parameter for Had(1,1)

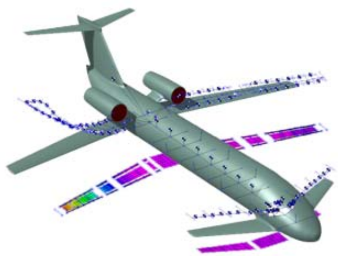




Solving **dynamic response in time domain**

Imaginary part function of k parameter for Had(1,1)





Solving **dynamic response in time domain**

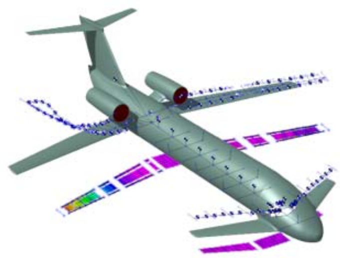
All the preprocesses are completed. Now one could start the actual solver for dynamic response steady state providing input and output:

[**ss_model**,**Y**,**T**,**X**,**U**] = ***solve_free_lin_dyn_ss***('Tmax',**6**,'dT',**0.001**,'Ham','**res1.mat**','Had','**res2.mat**');

The response will refer to a time-window of 6 sec, sampled at 1e-3 secs.

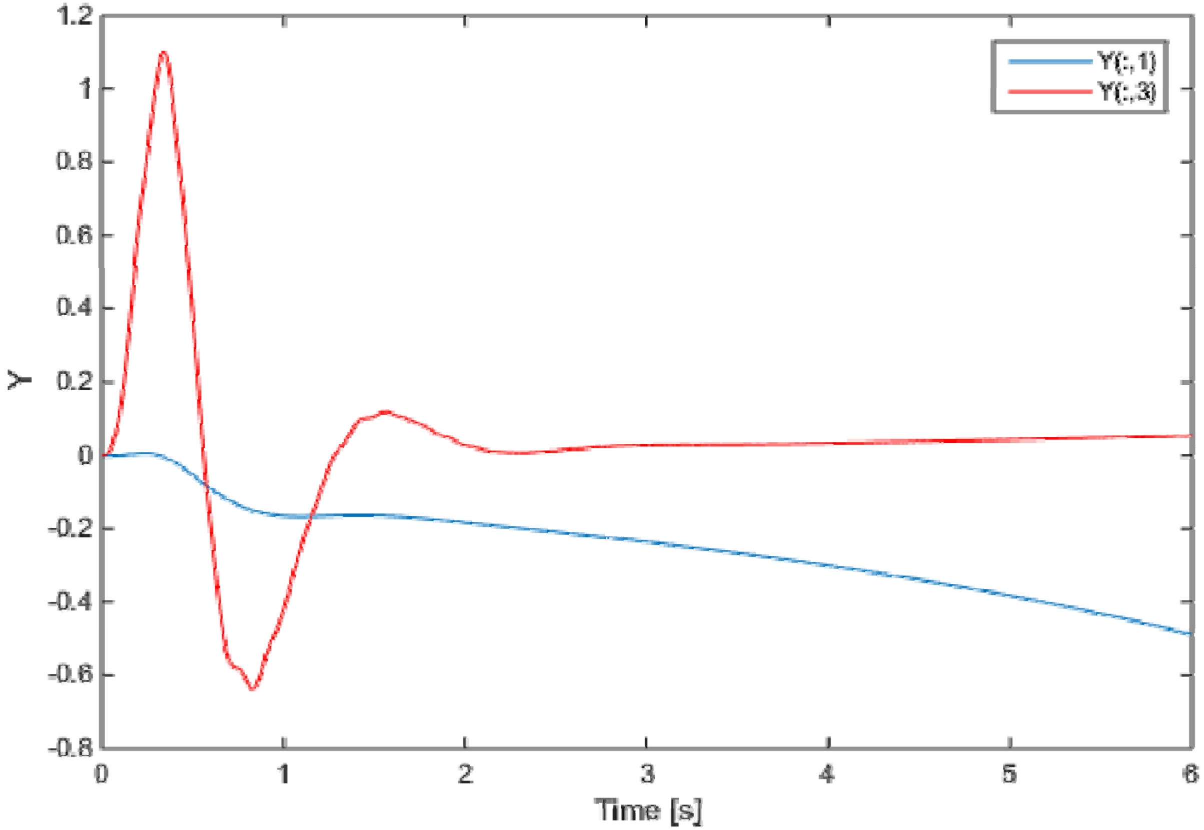
The results are saved in:

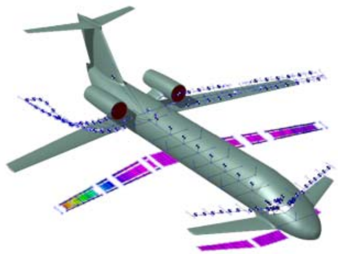
- *ss_model*: steady state model of the aircraft in response to control, gust or nodal force input;
- *Y*: all the modes components for vertical translation;
- *X*: all the modes components for horizontal translation;
- *T*: time steps vector;
- *U*: velocity vector in x,y and z directions.



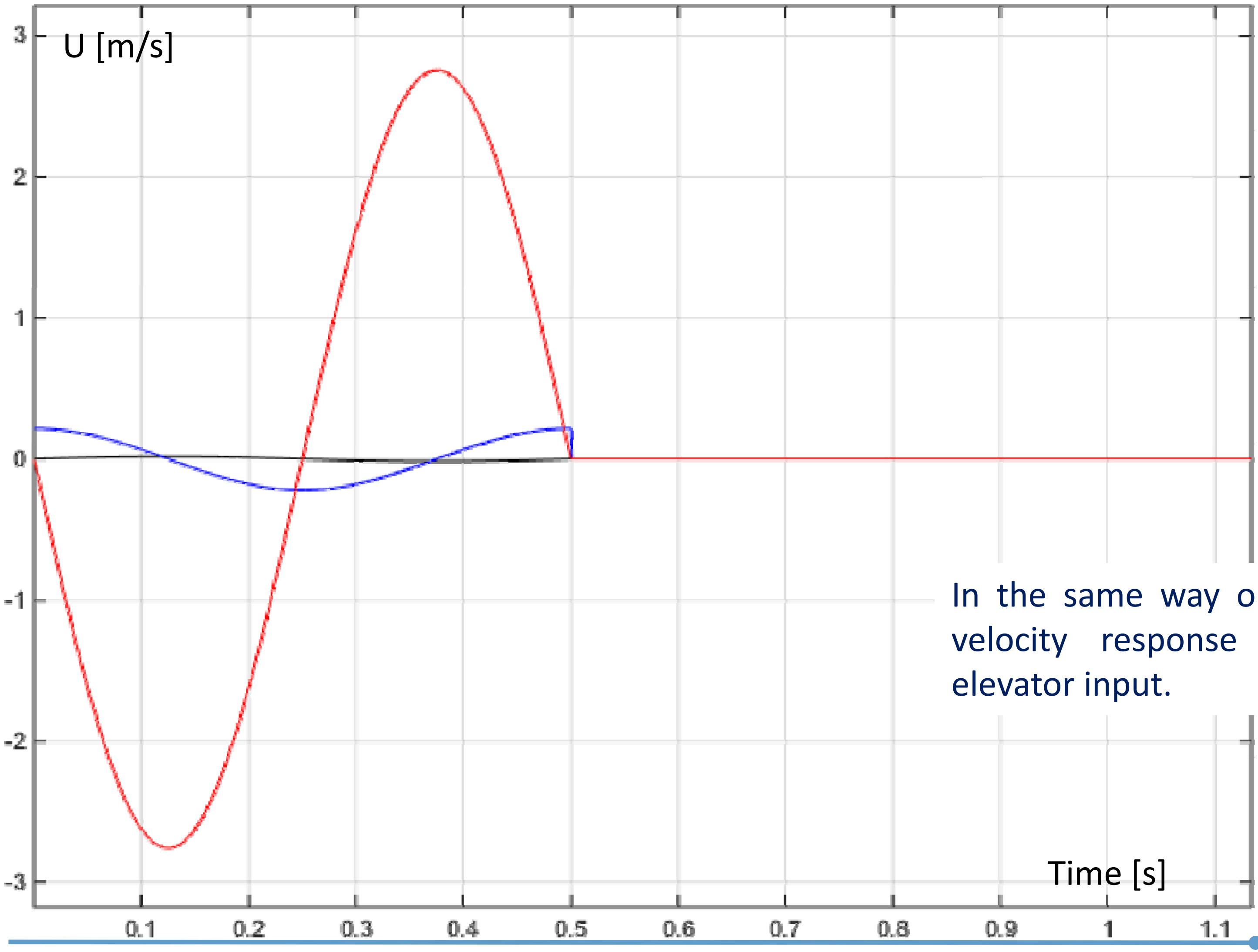
Solving **dynamic response in time domain**

For example the vertical translation can be plotted: *plot(T,Y(:,1));*
and similarly the first elastic mode: *plot(T,Y(:,3));*





Solving **dynamic response in time domain**



In the same way one could plot the velocity response to a sinusoidal elevator input.