GEBTAero

18.09

Generated by Doxygen 1.8.13

Contents

Chapter 1

GEBTAero

GEBTAero is an aeroelasticity simulation toolbox with a computation code coded in Fortran and a pre/postprocessor coded in Python. The computation code is derived from GEBT program developped by Prof. Yu (https://cdmhub.org/resources/gebt). The pre/postprocessor uses several open source programs available in most linux distros repositories:

- calculix : a Finite Element Method solver (http://www.calculix.de/)
- paraview : a data analysis and visualization application (https://www.paraview.org/)
- Mumps : a parallel sparse direct solver (http://mumps.enseeiht.fr/)
- Arpack : a sparse eigenvalue solver (https://www.caam.rice.edu/software/ARPACK/)

Installation

Two options are available:

Debian package

For Ubuntu 18.04 and Debian 10, download the .deb file available in the package folder and launch it. It will automatically install all the dependancies. For other linux distributions, you can ask for a .deb or .rpm package creation.

Compilation

Clone the repository use the MakeFile in the bin folder and adapt it to your system.

Install the dependancies. On Ubuntu:

```
sudo apt install paraview calculix-ccx calculix-cgx libmumps-seq-dev libarpack2-dev python3 python3-numpy
       python3-matplotlib gfortran make
```

Compile gebtaero and unical (mesh format translator from unv to inp)

Testing

The folder cas_test is a set of automated python script designed to test many program functionalities. in a terminal launch:

```
python3 tests.py
```

Usage

Besides cas_test folder, examples folder contains a set of detailled script designed to help you to set your own problems. The pre/postprocessor script must be launch with python3 (not python2).

```
python3 myscript.py
```

You can also directly use the computation code with .dat file (show examples):

```
gebtaero example.dat
```

Acknowledgement

This research work was funded by the French Air Force Academy Research Center in collaboration with ISAE-↩
Supaero

Chapter 2

Modules Index

2.1 Modules List

Here is a list of all modules with brief descriptions:

Chapter 3

Data Type Index

3.1 Data Types List

Here are the data types with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Module Documentation

5.1 cputime Module Reference

A module use to calculate the computation time.

Functions/Subroutines

- subroutine, public tic

  Start the timer.
- real function, public toc ()

  Stop the timer.

Variables

- integer(8) start
- integer(8) rate
- integer(8) finish

5.1.1 Detailed Description

A module use to calculate the computation time.

5.1.2 Function/Subroutine Documentation

5.1.2.1 tic()

```
subroutine, public cputime::tic ( )
```

Start the timer.

Definition at line 44 of file CPUtime.f90.

5.1.2.2 toc()

```
real function, public cputime::toc ( )
```

Stop the timer.

Returns

computation time (seconds)

Definition at line 58 of file CPUtime.f90.

5.1.3 Variable Documentation

5.1.3.1 finish

```
integer(8) cputime::finish  [private]
```

Definition at line 31 of file CPUtime.f90.

5.1.3.2 rate

```
integer(8) cputime::rate  [private]
```

Definition at line 31 of file CPUtime.f90.

5.1.3.3 start

```
integer(8) cputime::start  [private]
```

Definition at line 31 of file CPUtime.f90.

5.2 eigenmumps Module Reference

This module contains the main routines needed for eigen value analysis and allow to call Arpack and MUMPS library.

Functions/Subroutines

- subroutine, public eigensolvemumps (ndof_el, memb_info, v_root_a, omega_a, member, pt_condition, niter, error, ncond_mb, mb_condition, distr_fun, dof_con, x, nev, eigen_val, eigen_vec, aero_flag, grav_flag)

  this routine solve the eigenproblem by linearising about a steady state x
- subroutine arpack (nev, ncv, er, ei, vector, error)

  this subroutine make the interface with the arpack sparse eigensolver library
- subroutine aw (u, nzM, irnM, jcnM, coef_mass, w)

  Solve the linear system $K{*}V = Z$ to avoid matrix inversion with solver MUMPS.

Variables

- type(dmumps_struc) mumps_par

  an array containing the configuration parameters of MUMPS solver
- integer ierr

  error code of the MUMPS solver

5.2.1 Detailed Description

This module contains the main routines needed for eigen value analysis and allow to call Arpack and MUMPS library.

5.2.2 Function/Subroutine Documentation

5.2.2.1 arpack()

```
subroutine eigenmumps::arpack (
            integer, intent(inout) nev,
            integer, intent(in) ncv,
            real(dbl), dimension(:), intent(out) er,
            real(dbl), dimension(:), intent(out) ei,
            real(dbl), dimension(:,:), intent(out) vector,
            character(*), intent(out) error )  [private]
```

this subroutine make the interface with the arpack sparse eigensolver library

Parameters

in,out	*nev*	ioaero::nev
in	*ncv*	size of the subdomain used bye the IRAM algorithm (doc Arpack)
out	*error*	ioaero::error
out	*er*	Real part of the eigenvalue (before problem inversion)
out	*ei*	Imaginary part of the eigenvalue (before problem inversion)
out	*vector*	eigenvector computed by arpack

Definition at line 248 of file EigenSolveMumps.f90.

5.2.2.2 aw()

```
subroutine eigenmumps::aw (
            real(dbl), dimension(nsize), intent(out) u,
            integer, intent(in) nzM,
            integer, dimension(:), intent(in) irnM,
            integer, dimension(:), intent(in) jcnM,
            real(dbl), dimension(:), intent(in) coef_mass,
            real(dbl), dimension(nsize), intent(in) w )  [private]
```

Solve the linear system K∗V = Z to avoid matrix inversion with solver MUMPS.

Parameters

in	*nzm*	number of nonzero coefficient
in	*irnm*	line index of nonzero values
in	*jcnm*	column index of nonzero values
in	*coef_mass*	mass matrix sparse coefficient
in	*w*	RHS vector
out	*u*	solution vector V

Definition at line 357 of file EigenSolveMumps.f90.

5.2.2.3 eigensolvemumps()

```
subroutine, public eigenmumps::eigensolvemumps (
            integer, intent(in) ndof_el,
            type (memberinf), dimension(:), intent(in) memb_info,
            real(dbl), dimension(:), intent(in) v_root_a,
            real(dbl), dimension(:), intent(in) omega_a,
            integer, dimension(:,:), intent(in) member,
            type(prescriinf), dimension(:), intent(in) pt_condition,
            integer, intent(in) niter,
            character(*), intent(out) error,
            integer, intent(in) ncond_mb,
            type(prescriinf), dimension(:), intent(in) mb_condition,
            real(dbl), dimension(:,:), intent(in) distr_fun,
            integer, dimension(:)  dof_con,
            real(dbl), dimension(:), intent(in) x,
            integer, intent(inout) nev,
            real(dbl), dimension(:,:), intent(out) eigen_val,
            real(dbl), dimension(:,:), intent(out) eigen_vec,
            integer, intent(in) aero_flag,
            integer, intent(in) grav_flag )
```

this routine solve the eigenproblem by linearising about a steady state x

Parameters

in	*ndof_el*	ioaero::ndof_el
in	*aero_flag*	ioaero::aero_flag
in	*grav_flag*	ioaero::grav_flag
in	*memb_info*	array containing the characteristics of the beam members
in	*v_root_a*	linear velocity of frame a
in	*omega_a*	angular velocity of frame a
in	*distr_fun*	ioaero::distr_fun
in	*member*	ioaero::member
in	*niter*	ioaero::niter
in	*ncond_mb*	ioaero::ncond_mb
in	*pt_condition*	ioaero::pt_condition
in	*mb_condition*	ioaero::mb_condition
	dof_con	the connecting condition for key point.
out	*error*	ioaero::error
in	*x*	the solution vector of the linear system (steady-state)
in,out	*nev*	ioaero::nev
out	*eigen_val*	ioaero::eigen_val
out	*eigen_vec*	array containing the solution eigenvectors

Definition at line 45 of file EigenSolveMumps.f90.

5.2.3 Variable Documentation

5.2.3.1 ierr

```
integer eigenmumps::ierr
```

error code of the MUMPS solver

Definition at line 27 of file EigenSolveMumps.f90.

5.2.3.2 mumps_par

```
type (dmumps_struc) eigenmumps::mumps_par
```

an array containing the configuration parameters of MUMPS solver

Definition at line 26 of file EigenSolveMumps.f90.

5.3 element Module Reference

This module contains information and calculation for an element within a member.

Functions/Subroutines

- real(dbl) function, dimension(ndof_el+ndof_nd), public elemeqn (ndof_el)

    Compute the value of the Right Hand Side for a finite element.
- subroutine, public elemjacobian (ndof_el, niter, elemJac)

    Caculate the Jacobian matrix for each element.
- subroutine, public elemmass (ndof_el, elemM)

    Caculate the mass matrix for each element.
- subroutine, public extractelementproperties (elem_no, memb_info_i, x_elem, v_root_a, omega_a, ndof_el, init_elem, aero_flag, grav_flag)

    Extract element properties needed for element assembly.

Variables

- real(dbl) dl

    length of the element
- real(dbl) le

    the ending arc length of the current element
- real(dbl), dimension(ndim, ndim) ecab

    direction cosine matrix of the undeformed element
- real(dbl), dimension(nstrn, nstrn) eflex

    flexibility matrix of the elment
- real(dbl), dimension(nstrn, nstrn) emass

    inverse of the mass matrix of the element
- type(distriload), public load

    distributed load
- logical, public exist_load
- logical, public follower_load

    flags to indicate whether distributed load exist and whether they are follower forces
- real(dbl), dimension(ndim) ui
- real(dbl), dimension(ndim) theta
- real(dbl), dimension(ndim) fi
- real(dbl), dimension(ndim) mi
- real(dbl), dimension(ndim) e1gammad
- real(dbl), dimension(ndim) kappa
- real(dbl), dimension(ndim) epi
- real(dbl), dimension(ndim) hi
- real(dbl), dimension(ndim) vi
- real(dbl), dimension(ndim) omegai
- real(dbl), dimension(ndim) ev_i

    initial velocity of the mid point of the element
- real(dbl), dimension(ndim) eomega_a

    initial angular velocity of the element
- real(dbl), dimension(ndim, ndim) ect

    the transpose of the direction cosine matrix corresponding to elastic rotation
- real(dbl), dimension(ndim, ndim) ectcab

    eCT.Cab
- real(dbl), dimension(ndim, ndim) ecabhalfl

    eCab∗dL/2
- real(dbl), dimension(ndim, ndim) ectcabhalfl

    eCTCab∗dL/2

- real(dbl), dimension(ndim) uidot
- real(dbl), dimension(ndim) thetadot
- real(dbl), dimension(ndim) ctcabpdot
- real(dbl), dimension(ndim) ctcabhdot
- real(dbl) rho
- real(dbl) chord
- real(dbl) x_cg
- real(dbl) alpha
- real(dbl) alphadot
- real(dbl) hdot
- real(dbl) aw
- real(dbl) bw
- real(dbl) u
- real(dbl) hdotdot
- real(dbl) alphadotdot
- real(dbl) alpha_ac
- real(dbl) beta_ac
- real(dbl) beta
- integer a_flag
- integer g_flag
- real(dbl), dimension(nstates) lambda
- real(dbl), dimension(nstates) lambdadot
- real(dbl), dimension(nstates, 2 ∗nstates+2) p
- real(dbl), dimension(nstates, nstates) ident
- real(dbl) lambda0
- real(dbl), dimension(nstates, nstates) a
- real(dbl), dimension(nstates) b
- real(dbl), dimension(nstates) c
- real(dbl), dimension(ndim) dir_moment
- real(dbl), dimension(ndim) dir_lift
- real(dbl), dimension(ndim, ndim) jdir_moment
- real(dbl), dimension(ndim, ndim) jdir_lift
- real(dbl), dimension(ndim) jalpha_theta
- real(dbl), dimension(ndim) jalphadot_theta
- real(dbl), dimension(ndim) jhdot_theta
- real(dbl), dimension(ndim+ndim) jalphadot_ph
- real(dbl), dimension(ndim+ndim) jhdot_ph
- real(dbl), dimension(ndim+ndim) jalphadotdot_phdot
- real(dbl), dimension(ndim+ndim) jhdotdot_phdot
- real(dbl), dimension(ndim+ndim) jalphadotdot_ph
- real(dbl), dimension(ndim+ndim) jhdotdot_ph
- real(dbl), dimension(ndim+ndim) jlambda0_phdot
- real(dbl), dimension(nstates) jlambda0_lambda
- real(dbl), dimension(3, 18+nstates) jlift
- real(dbl), dimension(3, 18+nstates) jmoment
- real(dbl), dimension(ndim, ndim) ecaf
- real(dbl), dimension(ndim) wind

5.3.1 Detailed Description

This module contains information and calculation for an element within a member.

5.3.2 Function/Subroutine Documentation

5.3.2.1 elemeqn()

```
real(dbl) function, dimension(ndof_el+ndof_nd), public element::elemeqn (
            integer, intent(in) ndof_el )
```

Compute the value of the Right Hand Side for a finite element.

Parameters

in	*ndof←_el*	ioaero::ndof←_el

Definition at line 79 of file Element.f90.

5.3.2.2 elemjacobian()

```
subroutine, public element::elemjacobian (
            integer, intent(in) ndof_el,
            integer, intent(in) niter,
            real(dbl), dimension(:,:), intent(out) elemJac )
```

Caculate the Jacobian matrix for each element.

Parameters

in	*ndof←_el*	ioaero::ndof_el
in	*niter*	ioaero::niter
out	*elemjac*	the coefficient matrix for each element

Definition at line 209 of file Element.f90.

5.3.2.3 elemmass()

```
subroutine, public element::elemmass (
            integer, intent(in) ndof_el,
            real(dbl), dimension(:,:), intent(out) elemM )
```

Caculate the mass matrix for each element.

Parameters

in	*ndof↩_el*	ioaero::ndof_el
out	*elemm*	the mass matrix for each element

Definition at line 487 of file Element.f90.

5.3.2.4 extractelementproperties()

```
subroutine, public element::extractelementproperties (
            integer, intent(in) elem_no,
            type (memberinf), intent(in) memb_info_i,
            real(dbl), dimension(:), intent(in) x_elem,
            real(dbl), dimension(:), intent(in) v_root_a,
            real(dbl), dimension(:), intent(in) omega_a,
            integer, intent(in) ndof_el,
            real(dbl), dimension(:,:), intent(inout) init_elem,
            integer, intent(in) aero_flag,
            integer, intent(in) grav_flag )
```

Extract element properties needed for element assembly.

Parameters

in	*elem_no*	Element index
in	*memb_↩info_i*	the paramater of the members (see memberinf Type)
in	*x_elem*	the 12 variables of the element u_i, , F_i, M_i, for dynamic analysis, 6 more Pi, Hi. for initial step, x_elem contains, CTCabPdot, CTCabHdot, F_i, M_i, P_i, H_i. Finally for Peters aero the Ns induced flow states
in	*v_root_a*	linear velocity of the root point in inertial frame
in	*omega_a*	angular velocity of the root point in inertial frame
in,out	*init_elem*	initial step: the 12 initial values of the element u_i, $\theta_i, \dot{u}_i, \dot{theta}_i$. Time marching: $2/dtui + \dot{u}_i, 2/dtthetai + \dot{theta}_i, 2/dtCTCabP + \dot{;} 2/dtCTCabP + dot$
in	*ndof_el*	#ioaero::nedof_el
in	*aero_flag*	ioaero::aero_flag
in	*grav_flag*	ioaero::grav_flag

Definition at line 616 of file Element.f90.

5.3.3 Variable Documentation

5.3.3.1 a

```
real(dbl), dimension(nstates,nstates) element::a [private]
```

Definition at line 56 of file Element.f90.

5.3.3.2 a_flag

```
integer element::a_flag [private]
```

Definition at line 52 of file Element.f90.

5.3.3.3 alpha

```
real(dbl) element::alpha [private]
```

Definition at line 51 of file Element.f90.

5.3.3.4 alpha_ac

```
real(dbl) element::alpha_ac [private]
```

Definition at line 51 of file Element.f90.

5.3.3.5 alphadot

```
real(dbl) element::alphadot [private]
```

Definition at line 51 of file Element.f90.

5.3.3.6 alphadotdot

```
real(dbl) element::alphadotdot [private]
```

Definition at line 51 of file Element.f90.

5.3.3.7 aw

`real(dbl) element::aw [private]`

Definition at line 51 of file Element.f90.

5.3.3.8 b

`real(dbl), dimension(nstates) element::b [private]`

Definition at line 56 of file Element.f90.

5.3.3.9 beta

`real(dbl) element::beta [private]`

Definition at line 51 of file Element.f90.

5.3.3.10 beta_ac

`real(dbl) element::beta_ac [private]`

Definition at line 51 of file Element.f90.

5.3.3.11 bw

`real(dbl) element::bw [private]`

Definition at line 51 of file Element.f90.

5.3.3.12 c

`real(dbl), dimension(nstates) element::c [private]`

Definition at line 56 of file Element.f90.

5.3.3.13 chord

```
real(dbl) element::chord [private]
```

Definition at line 51 of file Element.f90.

5.3.3.14 ctcabhdot

```
real(dbl), dimension(ndim) element::ctcabhdot [private]
```

Definition at line 48 of file Element.f90.

5.3.3.15 ctcabpdot

```
real(dbl), dimension(ndim) element::ctcabpdot [private]
```

Definition at line 48 of file Element.f90.

5.3.3.16 dir_lift

```
real(dbl), dimension(ndim) element::dir_lift [private]
```

Definition at line 57 of file Element.f90.

5.3.3.17 dir_moment

```
real(dbl), dimension(ndim) element::dir_moment [private]
```

Definition at line 57 of file Element.f90.

5.3.3.18 dl

```
real(dbl) element::dl [private]
```

length of the element

Definition at line 31 of file Element.f90.

5.3.3.19 e1gammad

`real(dbl), dimension(ndim) element::e1gammad [private]`

Definition at line 39 of file Element.f90.

5.3.3.20 ecab

`real(dbl), dimension(ndim,ndim) element::ecab [private]`

direction cosine matrix of the undeformed element

Definition at line 33 of file Element.f90.

5.3.3.21 ecabhalfl

`real(dbl), dimension(ndim,ndim) element::ecabhalfl [private]`

eCab∗dL/2

Definition at line 45 of file Element.f90.

5.3.3.22 ecaf

`real(dbl), dimension(ndim,ndim) element::ecaf [private]`

Definition at line 63 of file Element.f90.

5.3.3.23 ect

`real(dbl), dimension(ndim,ndim) element::ect [private]`

the transpose of the direction cosine matrix corresponding to elastic rotation

Definition at line 43 of file Element.f90.

5.3.3.24 ectcab

```
real(dbl), dimension(ndim,ndim) element::ectcab [private]
```

eCT.Cab

Definition at line 44 of file Element.f90.

5.3.3.25 ectcabhalfl

```
real(dbl), dimension(ndim,ndim) element::ectcabhalfl [private]
```

eCTCab∗dL/2

Definition at line 46 of file Element.f90.

5.3.3.26 eflex

```
real(dbl), dimension(nstrn,nstrn) element::eflex [private]
```

flexibility matrix of the elment

Definition at line 34 of file Element.f90.

5.3.3.27 emass

```
real(dbl), dimension(nstrn,nstrn) element::emass [private]
```

inverse of the mass matrix of the element

Definition at line 35 of file Element.f90.

5.3.3.28 eomega_a

```
real(dbl), dimension(ndim) element::eomega_a [private]
```

initial angular velocity of the element

Definition at line 41 of file Element.f90.

5.3.3.29 epi

```
real(dbl), dimension(ndim) element::epi  [private]
```

Definition at line 39 of file Element.f90.

5.3.3.30 ev_i

```
real(dbl), dimension(ndim) element::ev_i  [private]
```

initial velocity of the mid point of the element

Definition at line 40 of file Element.f90.

5.3.3.31 exist_load

```
logical, public element::exist_load
```

Definition at line 37 of file Element.f90.

5.3.3.32 fi

```
real(dbl), dimension(ndim) element::fi  [private]
```

Definition at line 39 of file Element.f90.

5.3.3.33 follower_load

```
logical, public element::follower_load
```

flags to indicate whether distributed load exist and whether they are follower forces

Definition at line 37 of file Element.f90.

5.3.3.34 g_flag

```
integer element::g_flag  [private]
```

Definition at line 52 of file Element.f90.

5.3.3.35 hdot

`real(dbl) element::hdot [private]`

Definition at line 51 of file Element.f90.

5.3.3.36 hdotdot

`real(dbl) element::hdotdot [private]`

Definition at line 51 of file Element.f90.

5.3.3.37 hi

`real(dbl), dimension(ndim) element::hi [private]`

Definition at line 39 of file Element.f90.

5.3.3.38 ident

`real(dbl), dimension(nstates,nstates) element::ident [private]`

Definition at line 54 of file Element.f90.

5.3.3.39 jalpha_theta

`real(dbl), dimension(ndim) element::jalpha_theta [private]`

Definition at line 58 of file Element.f90.

5.3.3.40 jalphadot_ph

`real(dbl), dimension(ndim+ndim) element::jalphadot_ph [private]`

Definition at line 58 of file Element.f90.

5.3.3.41 jalphadot_theta

`real(dbl), dimension(ndim) element::jalphadot_theta [private]`

Definition at line 58 of file Element.f90.

5.3.3.42 jalphadotdot_ph

`real(dbl), dimension(ndim+ndim) element::jalphadotdot_ph [private]`

Definition at line 60 of file Element.f90.

5.3.3.43 jalphadotdot_phdot

`real(dbl), dimension(ndim+ndim) element::jalphadotdot_phdot [private]`

Definition at line 59 of file Element.f90.

5.3.3.44 jdir_lift

`real(dbl), dimension(ndim,ndim) element::jdir_lift [private]`

Definition at line 57 of file Element.f90.

5.3.3.45 jdir_moment

`real(dbl), dimension(ndim,ndim) element::jdir_moment [private]`

Definition at line 57 of file Element.f90.

5.3.3.46 jhdot_ph

`real(dbl), dimension(ndim+ndim) element::jhdot_ph [private]`

Definition at line 58 of file Element.f90.

5.3.3.47 jhdot_theta

`real(dbl), dimension(ndim) element::jhdot_theta [private]`

Definition at line 58 of file Element.f90.

5.3.3.48 jhdotdot_ph

`real(dbl), dimension(ndim+ndim) element::jhdotdot_ph [private]`

Definition at line 60 of file Element.f90.

5.3.3.49 jhdotdot_phdot

`real(dbl), dimension(ndim+ndim) element::jhdotdot_phdot [private]`

Definition at line 59 of file Element.f90.

5.3.3.50 jlambda0_lambda

`real(dbl), dimension(nstates) element::jlambda0_lambda [private]`

Definition at line 61 of file Element.f90.

5.3.3.51 jlambda0_phdot

`real(dbl), dimension(ndim+ndim) element::jlambda0_phdot [private]`

Definition at line 61 of file Element.f90.

5.3.3.52 jlift

`real(dbl), dimension(3,18+nstates) element::jlift [private]`

Definition at line 62 of file Element.f90.

5.3.3.53 jmoment

`real(dbl), dimension(3,18+nstates) element::jmoment [private]`

Definition at line 62 of file Element.f90.

5.3.3.54 kappa

`real(dbl), dimension(ndim) element::kappa [private]`

Definition at line 39 of file Element.f90.

5.3.3.55 lambda

`real(dbl), dimension(nstates) element::lambda [private]`

Definition at line 53 of file Element.f90.

5.3.3.56 lambda0

`real(dbl) element::lambda0 [private]`

Definition at line 55 of file Element.f90.

5.3.3.57 lambdadot

`real(dbl), dimension(nstates) element::lambdadot [private]`

Definition at line 53 of file Element.f90.

5.3.3.58 le

`real(dbl) element::le [private]`

the ending arc length of the current element

Definition at line 32 of file Element.f90.

5.3.3.59 load

```
type(distriload), public element::load
```

distributed load

Definition at line 36 of file Element.f90.

5.3.3.60 mi

```
real(dbl), dimension(ndim) element::mi  [private]
```

Definition at line 39 of file Element.f90.

5.3.3.61 omegai

```
real(dbl), dimension(ndim) element::omegai  [private]
```

Definition at line 39 of file Element.f90.

5.3.3.62 p

```
real(dbl), dimension(nstates,2*nstates+2) element::p  [private]
```

Definition at line 53 of file Element.f90.

5.3.3.63 rho

```
real(dbl) element::rho  [private]
```

Definition at line 51 of file Element.f90.

5.3.3.64 theta

```
real(dbl), dimension(ndim) element::theta  [private]
```

Definition at line 39 of file Element.f90.

5.3.3.65 thetadot

```
real(dbl), dimension(ndim) element::thetadot  [private]
```

Definition at line 48 of file Element.f90.

5.3.3.66 u

```
real(dbl) element::u  [private]
```

Definition at line 51 of file Element.f90.

5.3.3.67 ui

```
real(dbl), dimension(ndim) element::ui  [private]
```

Definition at line 39 of file Element.f90.

5.3.3.68 uidot

```
real(dbl), dimension(ndim) element::uidot  [private]
```

Definition at line 48 of file Element.f90.

5.3.3.69 vi

```
real(dbl), dimension(ndim) element::vi  [private]
```

Definition at line 39 of file Element.f90.

5.3.3.70 wind

```
real(dbl), dimension(ndim) element::wind  [private]
```

Definition at line 63 of file Element.f90.

5.3.3.71 x_cg

```
real(dbl) element::x_cg [private]
```

Definition at line 51 of file Element.f90.

5.4 gebtaero Namespace Reference

Namespaces

- CompositeBox
- CompositePlate
- CompositePly
- CrossSection
- ExternalMesh
- Frame
- GebtPlot
- InputFile
- IsoMaterial
- OrthoMaterial
- Simulation
- TimeFunction
- utils
- Wing
- WingSection

5.5 gebtaero.CompositeBox Namespace Reference

Classes

- class CompositeBox

    Class interfacing the solver with 3D FEM calculix computation to obtain the cross section parameter from a composite box.

5.6 gebtaero.CompositePlate Namespace Reference

Classes

- class CompositePlate

5.7 gebtaero.CompositePly Namespace Reference

Classes

- class CompositePly

5.8 gebtaero.CrossSection Namespace Reference

Classes

- class CrossSection

5.9 gebtaero.ExternalMesh Namespace Reference

Classes

- class ExternalMesh

5.10 gebtaero.Frame Namespace Reference

Classes

- class Frame

5.11 gebtaero.GebtPlot Namespace Reference

Classes

- class GebtPlot

5.12 gebtaero.InputFile Namespace Reference

Classes

- class InputFile

5.13 gebtaero.IsoMaterial Namespace Reference

Classes

- class IsoMaterial

5.14 gebtaero.OrthoMaterial Namespace Reference

Classes

- class OrthoMaterial

5.15 gebtaero.Simulation Namespace Reference

Classes

- class Simulation

5.16 gebtaero.TimeFunction Namespace Reference

Classes

- class TimeFunction

5.17 gebtaero.utils Namespace Reference

Functions

- def CreatePeriodicEq (MeshFile, OffsetY=0., OffsetZ=0.)
- def RunFbdFile (FileName)
- def RunInpFile (FileName)
- def RunFrdFile (FileName)
- def RunParaviewScript (FileName)
- def RunUnvConv (FileName, FileOut, Reduced='R')
- def RemoveFiles ()
- def RemoveMeshFiles ()
- def ReadNodesField (self, FileName, FieldName)
- def ReadFlexibilityFromDisp (FileName, nstrain, ncurv, Lx, tol, RigidX=False, RigidZ=False)
- def ReadEigenVec (FileName)
- def CorrelateTab (Tab1, Tab2, Index)
- def ComputeElasticCenterFromFlexMat (FlexMat)
- def ReadFromPipe (Command)
- def ReadModesFromPipe (Command, output="modes")
- def ReadLoadsFromPipe (Command, output="static")

5.17.1 Function Documentation

5.17.1.1 ComputeElasticCenterFromFlexMat()

```
def gebtaero.utils.ComputeElasticCenterFromFlexMat (
              FlexMat )
```

Definition at line 276 of file utils.py.

5.17.1.2 CorrelateTab()

```
def gebtaero.utils.CorrelateTab (
              Tab1,
              Tab2,
              Index )
```

Definition at line 248 of file utils.py.

5.17.1.3 CreatePeriodicEq()

```
def gebtaero.utils.CreatePeriodicEq (
              MeshFile,
              OffsetY = 0.,
              OffsetZ = 0.  )
```

Definition at line 8 of file utils.py.

5.17.1.4 ReadEigenVec()

```
def gebtaero.utils.ReadEigenVec (
              FileName )
```

Definition at line 230 of file utils.py.

5.17.1.5 ReadFlexibilityFromDisp()

```
def gebtaero.utils.ReadFlexibilityFromDisp (
              FileName,
              nstrain,
              ncurv,
              Lx,
              tol,
              RigidX = False,
              RigidZ = False )
```

Definition at line 185 of file utils.py.

5.17.1.6 ReadFromPipe()

```
def gebtaero.utils.ReadFromPipe (
              Command )
```

Definition at line 284 of file utils.py.

5.17.1.7 ReadLoadsFromPipe()

```
def gebtaero.utils.ReadLoadsFromPipe (
              Command,
              output = "static" )
```

Definition at line 400 of file utils.py.

5.17.1.8 ReadModesFromPipe()

```
def gebtaero.utils.ReadModesFromPipe (
              Command,
              output = "modes" )
```

Definition at line 322 of file utils.py.

5.17.1.9 ReadNodesField()

```
def gebtaero.utils.ReadNodesField (
              self,
              FileName,
              FieldName )
```

Definition at line 149 of file utils.py.

5.17.1.10 RemoveFiles()

```
def gebtaero.utils.RemoveFiles ( )
```

Definition at line 141 of file utils.py.

5.17.1.11 RemoveMeshFiles()

```
def gebtaero.utils.RemoveMeshFiles ( )
```

Definition at line 145 of file utils.py.

5.17.1.12 RunFbdFile()

```
def gebtaero.utils.RunFbdFile (
              FileName )
```

Definition at line 121 of file utils.py.

5.17.1.13 RunFrdFile()

```
def gebtaero.utils.RunFrdFile (
              FileName )
```

Definition at line 129 of file utils.py.

5.17.1.14 RunInpFile()

```
def gebtaero.utils.RunInpFile (
              FileName )
```

Definition at line 125 of file utils.py.

5.17.1.15 RunParaviewScript()

```
def gebtaero.utils.RunParaviewScript (
              FileName )
```

Definition at line 133 of file utils.py.

5.17.1.16 RunUnvConv()

```
def gebtaero.utils.RunUnvConv (
              FileName,
              FileOut,
              Reduced = 'R' )
```

Definition at line 137 of file utils.py.

5.18 gebtaero.Wing Namespace Reference

Classes

- class Wing

5.19 gebtaero.WingSection Namespace Reference

Classes

- class WingSection

5.20 globaldatafun Module Reference

This module contains general-purpose global constants,I/O functions/subroutines and math functions/subroutines.

Data Types

- interface writevec

Functions/Subroutines

- logical function, public fileopen (file_unit, file_name, sta_type, rw_type, error)

  To open an old or new file for reading or writing ∗.
- logical function, public ioerror (message, error)

  Check the error of I/O processing ∗.
- character(20) function itochar (n)

  Convert an integer to character ∗.
- logical function, public memoryerror (vari_name, error)

  Check the error of memory allocation ∗.
- subroutine, public titleprint (file_unit, title)

  To print a title for a block of data ∗.
- subroutine, public writeerror (EIN, error)

  Write error to the echo file ∗.
- subroutine writeintvector (file_unit, vec)

  Write an integer vector to the file_unit ∗.
- subroutine writerealvector (file_unit, vec)

  Write a real vector to the file_unit ∗.
- subroutine, public ct_theta (theta, eCT, ekttek, eCTtheta)

  Calculate eC^T derivative w.r.t theta ∗ return derivative and ekttek ∗.
- real(dbl) function, dimension(3, 3), public ct_theta_t (theta, eCT, x)

  Calculate $e\dot{C}^T.x$ *derivative w.r.t* $th\dot{e}ta$ *∗.*
- real(dbl) function, dimension(3, 3), public dircosinetrodrigues (theta)

  Calculate the transpose of the direction cosine in ∗ terms of rodrigues parameters ∗.
- subroutine, public invert (matrix_in, matrix, vari_name, error)

  Invert a small square matrix ∗.
- real(dbl) function, dimension(size(mat, 1), size(mat, 2)), public matmul3 (mat, vec)

  Multiply a rank 3 matrix with a vector with every colum ∗ of the resulting matrix is equal to the multiplication ∗.
- real(dbl) function, public norm (vector)

  Calculate the L2 norm of a real vector ∗.
- real(dbl) function, dimension(size(vec1), size(vec2)), public outerproduct (vec1, vec2)

  Calculate the outer product of two vectors ∗.
- real(dbl) function, dimension(3, 3), public tilde (vect)

Carry out the tilde operation for a real vector ∗.
- real(dbl) function, dimension(3), public crossproduct (a, b)

  Carry out cross product of two real vectors ∗.
- subroutine, public insert1delement (nz, tmpR, irn, jcn, elemCoef1D, str_r1, str_c1, str_r2, str_c2, str_r3, str↩_c3, str_r4, str_c4)

  Insert a real matrix into the 1D coefficient matrix.
- subroutine, public extract2delement (nz, irn, jcn, elemCoef1D, tmpR, str_r1, str_c1)

  Back a 2D array from the 1D coefficient matrix.
- real(dbl) function, dimension(nsize), public matmul_sparse (vector, nsize, ne, irn, jcn, matrix1D)

  Matmul(vector, matrix) with matrix stored in a spare format.
- real(dbl) function, dimension(n, 2 ∗n+2), public peters (n)

  Functions used for the Finite State Unsteady Thin Airfoil Theory of Peters see Introduction to Structural Dynamics and Aeroelasticity by Hodges p 139.
- real(dbl) function, dimension(n, n), public mata (n)

  Compute the Peters A matrix.
- real(dbl) function, dimension(n, n) matd (n)

  Compute the Peters D matrix.
- real(dbl) function, dimension(n) vecb (n)

  Compute the Peters B vector.
- real(dbl) function, dimension(n) vecc (n)

  Compute Peters C vector.
- real(dbl) function, dimension(n) vecd (n)

  Compute Peters D vector.
- real(dbl) function, dimension(n, n) prod (Vec1, Vec2, n)

  Compute the product of two square matrix.
- real(dbl) function factoriel (n)

  Compute the value of factoriel(n)

Variables

- integer, parameter, public ndim =3

  All the beams could behavior in the 3D space.
- integer, parameter, public ndof_nd =12

  degrees of freedom per node/element is 12.
- integer, parameter, public nstrn =6

  Number of strain measures/dofs in Timoshenko model.
- integer, parameter, public memb_const =7

  Number of labels needed for member properties.
- integer, parameter, public nstates = 6

  Number of induces-flow states in Peters theory.
- integer, parameter, public dbl =SELECTED_REAL_KIND(15, 307)
- real(dbl), parameter, public pi = 3.1415926535897932D0
- real(dbl), parameter, public deg_2_rad = 1.7453292519943296D-2

  the ratio between radians and degrees
- real(dbl), parameter, public rad_2_deg = 5.7295779513082321D1

  convert radian to degree
- real(dbl), parameter, public tolerance = EPSILON(1.0_DBL)

  a smart number of the double precision real number
- real(dbl), parameter, public grav = 9.81

  gravity acceleration

- real(dbl), dimension(3, 3), parameter, public i3 = RESHAPE((/1.D0, 0.D0, 0.D0, 0.D0, 1.D0, 0.D0, 0.D0, 0.D0, 1.D0/), (/3,3/))

  The 3x3 identity matrix.
- real(dbl), dimension(3), parameter, public e1 =(/1._DBL,0._DBL,0._DBL/)

  The e1 unit vector.
- integer, public in_stat

  flag to indicate if the I/O process is successful: if positive, an error occured; if negative, an end-of-file or end-of-record condition occurred; zero, no error, end-of-file, or end-of-record condition occurred.
- integer, public allo_stat

  flag to indicate status of allocating memory
- character(∗), parameter, public fmt_real ='ES15.7'

  format for output real numbers
- character(∗), parameter, public fmt_int ='I8'

  format for output integer numbers
- integer, public runmod =0

  Define the output behavior of the program; 0: legacy mode of the computation code with output of a .out text file; 1: mode compatible with the python pre/postrpocessor (argument -p in the terminal), 2: silent mode (argument -s in the terminal)
- integer, public arpack_mod =0

  parameter WHICH of arpack solver (1:LI, 2:LM, 3:LR, 4:SR, default : LM in dnaupd and LI in dneupd) =>cf Arpack doc
- integer, public eigen_output =0

  define wich eigenvalue data to output (0: eigenvalues and eigenvectors, 1: eigenvalues only)
- character(10), public solver ='MUMPS'

  linear solver used (HSL : ddep.f, mc19.f + ma28 or MUMPS : linux library)
- integer, public flutter_flag =0

  used in temporal simulation : 0= deformation are under a "flutter" state; 1= deformation are over a "flutter" state
- real(dbl), public flutter_limit

  the value of maximale angular deformaton use to trigger the flutter flag

5.20.1 Detailed Description

This module contains general-purpose global constants,I/O functions/subroutines and math functions/subroutines.

5.20.2 Function/Subroutine Documentation

5.20.2.1 crossproduct()

```
real(dbl) function, dimension(3), public globaldatafun::crossproduct (
            real(dbl), dimension(3), intent(in) a,
            real(dbl), dimension(3), intent(in) b )
```

Carry out cross product of two real vectors ∗.

Definition at line 653 of file GlobalDataFun.f90.

5.20.2.2 ct_theta()

```
subroutine, public globaldatafun::ct_theta (
            real(dbl), dimension(:), intent(in) theta,
            real(dbl), dimension(:,:), intent(in) eCT,
            real(dbl), dimension(:,:,:), intent(out) ekttek,
            real(dbl), dimension(:,:,:), intent(out) eCTtheta )
```

Calculate eC^T derivative w.r.t theta $*$ return derivative and ekttek $*$.

Parameters

in	*theta*	Rodrigues rotation parameters
in	*ect*	Direction Cosine matrix between frame b and B
out	*ekttek*	=OuterProduct(ek,theta)+OuterProduct(theta,ek)
out	*ecttheta*	Derivatives of eCT relative to theta

Definition at line 426 of file GlobalDataFun.f90.

5.20.2.3 ct_theta_t()

```
real(dbl) function, dimension(3,3), public globaldatafun::ct_theta_t (
            real(dbl), dimension(:), intent(in) theta,
            real(dbl), dimension(:,:), intent(in) eCT,
            real(dbl), dimension(:), intent(in) x )
```

Calculate $e\dot{C}^T.x$ derivative w.r.t $th\dot{e}ta$ $*$.

Parameters

in	*theta*	Rodrigues rotation parameters
in	*ect*	Direction Cosine matrix between frame b and B
in	*x*	vector

Definition at line 453 of file GlobalDataFun.f90.

5.20.2.4 dircosinetrodrigues()

```
real(dbl) function, dimension(3,3), public globaldatafun::dircosinetrodrigues (
            real(dbl), dimension(:), intent(in) theta )
```

Calculate the transpose of the direction cosine in $*$ terms of rodrigues parameters $*$.

Definition at line 479 of file GlobalDataFun.f90.

5.20.2.5 extract2delement()

```
subroutine, public globaldatafun::extract2delement (
            integer, intent(in) nz,
            integer, dimension(:), intent(in) irn,
            integer, dimension(:), intent(in) jcn,
            real(dbl), dimension(:), intent(in) elemCoef1D,
            real(dbl), dimension(:,:), intent(out) tmpR,
            integer, intent(in) str_r1,
            integer, intent(in) str_c1 )
```

Back a 2D array from the 1D coefficient matrix.

Definition at line 725 of file GlobalDataFun.f90.

5.20.2.6 factoriel()

```
real(dbl) function globaldatafun::factoriel (
            integer, intent(in) n )  [private]
```

Compute the value of factoriel(n)

Definition at line 891 of file GlobalDataFun.f90.

5.20.2.7 fileopen()

```
logical function, public globaldatafun::fileopen (
            integer, intent(in) file_unit,
            character(*), intent(in) file_name,
            character(*), intent(in) sta_type,
            character(*), intent(in) rw_type,
            character(*), intent(out) error )
```

To open an old or new file for reading or writing ∗.

Parameters

in	*file_unit*	File Unit (see fortran IO doc)
in	*sta_type*	status type
in	*rw_type*	rewrite configuration
out	*error*	ioaero::error

Definition at line 167 of file GlobalDataFun.f90.

5.20.2.8 insert1delement()

```
subroutine, public globaldatafun::insert1delement (
            integer, intent(inout) nz,
            real(dbl), dimension(:,:), intent(in) tmpR,
            integer, dimension(:), intent(inout) irn,
            integer, dimension(:), intent(inout) jcn,
            real(dbl), dimension(:), intent(inout) elemCoef1D,
            integer, intent(in) str_r1,
            integer, intent(in) str_c1,
            integer, intent(in), optional str_r2,
            integer, intent(in), optional str_c2,
            integer, intent(in), optional str_r3,
            integer, intent(in), optional str_c3,
            integer, intent(in), optional str_r4,
            integer, intent(in), optional str_c4 )
```

Insert a real matrix into the 1D coefficient matrix.

Definition at line 671 of file GlobalDataFun.f90.

5.20.2.9 invert()

```
subroutine, public globaldatafun::invert (
            real(dbl), dimension(:,:), intent(in) matrix_in,
            real(dbl), dimension(:,:), intent(out) matrix,
            character(*), intent(in) vari_name,
            character(*), intent(out) error )
```

Invert a small square matrix $*$.

Parameters

in	*matrix_in*	the matrix to be inverted
out	*matrix*	the inverse of the matrix

Definition at line 498 of file GlobalDataFun.f90.

5.20.2.10 ioerror()

```
logical function, public globaldatafun::ioerror (
            character(*), intent(in) message,
            character(*), intent(out) error )
```

Check the error of I/O processing $*$.

Parameters

in	*message*	a character variable to hold error message
out	*error*	ioaero::error

Definition at line 198 of file GlobalDataFun.f90.

5.20.2.11 itochar()

```
character(20) function globaldatafun::itochar (
            integer, intent(in) n )  [private]
```

Convert an integer to character ∗.

Definition at line 222 of file GlobalDataFun.f90.

5.20.2.12 mata()

```
real(dbl) function, dimension(n,n), public globaldatafun::mata (
            integer, intent(in) n )
```

Compute the Peters A matrix.

Definition at line 796 of file GlobalDataFun.f90.

5.20.2.13 matd()

```
real(dbl) function, dimension(n,n) globaldatafun::matd (
            integer, intent(in) n )  [private]
```

Compute the Peters D matrix.

Definition at line 807 of file GlobalDataFun.f90.

5.20.2.14 matmul3()

```
real(dbl) function, dimension(size(mat,1),size(mat,2)), public globaldatafun::matmul3 (
            real(dbl), dimension(:,:,:), intent(in) mat,
            real(dbl), dimension(:), intent(in) vec )
```

Multiply a rank 3 matrix with a vector with every colum ∗ of the resulting matrix is equal to the multiplication ∗.

Definition at line 554 of file GlobalDataFun.f90.

5.20.2.15 matmul_sparse()

```
real(dbl) function, dimension(nsize), public globaldatafun::matmul_sparse (
            real(dbl), dimension(:), intent(in) vector,
            integer, intent(in) nsize,
            integer, intent(in) ne,
            integer, dimension(:), intent(in) irn,
            integer, dimension(:), intent(in) jcn,
            real(dbl), dimension(:), intent(in) matrix1D )
```

Matmul(vector, matrix) with matrix stored in a spare format.

Definition at line 758 of file GlobalDataFun.f90.

5.20.2.16 memoryerror()

```
logical function, public globaldatafun::memoryerror (
            character(*), intent(in) vari_name,
            character(*), intent(out) error )
```

Check the error of memory allocation *.

Parameters

in	*vari_name*	a character variable to hold variable name
out	*error*	ioaero::error

Definition at line 267 of file GlobalDataFun.f90.

5.20.2.17 norm()

```
real(dbl) function, public globaldatafun::norm (
            real(dbl), dimension(:), intent(in) vector )
```

Calculate the L2 norm of a real vector *.

Definition at line 578 of file GlobalDataFun.f90.

5.20.2.18 outerproduct()

```
real(dbl) function, dimension(size(vec1),size(vec2)), public globaldatafun::outerproduct (
            real(dbl), dimension(:), intent(in) vec1,
            real(dbl), dimension(:), intent(in) vec2 )
```

Calculate the outer product of two vectors *.

Definition at line 595 of file GlobalDataFun.f90.

5.20.2.19 peters()

```
real(dbl) function, dimension(n,2*n+2), public globaldatafun::peters (
            integer, intent(in) n )
```

Functions used for the Finite State Unsteady Thin Airfoil Theory of Peters see Introduction to Structural Dynamics and Aeroelasticity by Hodges p 139.

Parameters

in	*n*	Number of induced flow states (Ns)

Returns

Matrix containing ordered in line : the A matrix, the B vector the C vector, the Identity matrix

Definition at line 780 of file GlobalDataFun.f90.

5.20.2.20 prod()

```
real(dbl) function, dimension(n,n) globaldatafun::prod (
            real(dbl), dimension(n), intent(in) Vec1,
            real(dbl), dimension(n), intent(in) Vec2,
            integer, intent(in) n )  [private]
```

Compute the product of two square matrix.

Definition at line 875 of file GlobalDataFun.f90.

5.20.2.21 tilde()

```
real(dbl) function, dimension(3,3), public globaldatafun::tilde (
            real(dbl), dimension(3), intent(in) vect )
```

Carry out the tilde operation for a real vector $*$.

Definition at line 625 of file GlobalDataFun.f90.

5.20.2.22 titleprint()

```
subroutine, public globaldatafun::titleprint (
            integer, intent(in) file_unit,
            character(*), intent(in) title )
```

To print a title for a block of data $*$.

Definition at line 292 of file GlobalDataFun.f90.

5.20.2.23 vecb()

```
real(dbl) function, dimension(n) globaldatafun::vecb (
            integer, intent(in) n )  [private]
```

Compute the Peters B vector.

Definition at line 825 of file GlobalDataFun.f90.

5.20.2.24 vecc()

```
real(dbl) function, dimension(n) globaldatafun::vecc (
            integer, intent(in) n )  [private]
```

Compute Peters C vector.

Definition at line 848 of file GlobalDataFun.f90.

5.20.2.25 vecd()

```
real(dbl) function, dimension(n) globaldatafun::vecd (
            integer, intent(in) n )  [private]
```

Compute Peters D vector.

Definition at line 863 of file GlobalDataFun.f90.

5.20.2.26 writeerror()

```
subroutine, public globaldatafun::writeerror (
            integer, intent(in) EIN,
            character(*), intent(in) error )
```

Write error to the echo file *.

Parameters

in	*ein*	file unit to write the error message

Definition at line 340 of file GlobalDataFun.f90.

5.20.2.27 writeintvector()

```
subroutine globaldatafun::writeintvector (
            integer, intent(in) file_unit,
            integer, dimension(:), intent(in) vec )  [private]
```

Write an integer vector to the file_unit *.

Parameters

in	*file_unit*	File unit to write the vector

Definition at line 375 of file GlobalDataFun.f90.

5.20.2.28 writerealvector()

```
subroutine globaldatafun::writerealvector (
            integer, intent(in) file_unit,
            real(dbl), dimension(:), intent(in) vec )  [private]
```

Write a real vector to the file_unit *.

Parameters

in	*file_unit*	File unit to write the vector

Definition at line 392 of file GlobalDataFun.f90.

5.20.3 Variable Documentation

5.20.3.1 allo_stat

```
integer, public globaldatafun::allo_stat
```

flag to indicate status of allocating memory

Definition at line 137 of file GlobalDataFun.f90.

5.20.3.2 arpack_mod

```
integer, public globaldatafun::arpack_mod =0
```

parameter WHICH of arpack solver (1:LI, 2:LM, 3:LR, 4:SR, default : LM in dnaupd and LI in dneupd) =>cf Arpack doc

Definition at line 144 of file GlobalDataFun.f90.

5.20.3.3 dbl

```
integer, parameter, public globaldatafun::dbl =SELECTED_REAL_KIND(15, 307)
```

Definition at line 121 of file GlobalDataFun.f90.

5.20.3.4 deg_2_rad

```
real(dbl), parameter, public globaldatafun::deg_2_rad = 1.7453292519943296D-2
```

the ratio between radians and degrees

Definition at line 124 of file GlobalDataFun.f90.

5.20.3.5 e1

```
real(dbl), dimension(3), parameter, public globaldatafun::e1 =(/1._DBL,0._DBL,0._DBL/)
```

The e1 unit vector.

Definition at line 132 of file GlobalDataFun.f90.

5.20.3.6 eigen_output

```
integer, public globaldatafun::eigen_output =0
```

define wich eigenvalue data to output (0: eigenvalues and eigenvectors, 1: eigenvalues only)

Definition at line 145 of file GlobalDataFun.f90.

5.20.3.7 flutter_flag

```
integer, public globaldatafun::flutter_flag =0
```

used in temporal simulation : 0= deformation are under a "flutter" state; 1= deformation are over a "flutter" state

Definition at line 147 of file GlobalDataFun.f90.

5.20.3.8 flutter_limit

`real(dbl), public globaldatafun::flutter_limit`

the value of maximale angular deformaton use to trigger the flutter flag

Definition at line 148 of file GlobalDataFun.f90.

5.20.3.9 fmt_int

`character(*), parameter, public globaldatafun::fmt_int ='I8'`

format for output integer numbers

Definition at line 141 of file GlobalDataFun.f90.

5.20.3.10 fmt_real

`character(*), parameter, public globaldatafun::fmt_real ='ES15.7'`

format for output real numbers

Definition at line 140 of file GlobalDataFun.f90.

5.20.3.11 grav

`real(dbl), parameter, public globaldatafun::grav = 9.81`

gravity acceleration

Definition at line 127 of file GlobalDataFun.f90.

5.20.3.12 i3

`real(dbl), dimension(3,3), parameter, public globaldatafun::i3 = RESHAPE((/1.D0, 0.D0, 0.D0, 0.D0, 1.D0, 0.D0, 0.D0, 0.D0, 1.D0/), (/3,3/))`

The 3x3 identity matrix.

Definition at line 129 of file GlobalDataFun.f90.

5.20.3.13 in_stat

`integer, public globaldatafun::in_stat`

flag to indicate if the I/O process is successful: if positive, an error occured; if negative, an end-of-file or end-of-record condition occurred; zero, no error, end-of-file, or end-of-record condition occurred.

Definition at line 134 of file GlobalDataFun.f90.

5.20.3.14 memb_const

`integer, parameter, public globaldatafun::memb_const =7`

Number of labels needed for member properties.

Definition at line 118 of file GlobalDataFun.f90.

5.20.3.15 ndim

`integer, parameter, public globaldatafun::ndim =3`

All the beams could behavior in the 3D space.

Definition at line 115 of file GlobalDataFun.f90.

5.20.3.16 ndof_nd

`integer, parameter, public globaldatafun::ndof_nd =12`

degrees of freedom per node/element is 12.

Definition at line 116 of file GlobalDataFun.f90.

5.20.3.17 nstates

`integer, parameter, public globaldatafun::nstates = 6`

Number of induces-flow states in Peters theory.

Definition at line 119 of file GlobalDataFun.f90.

5.20.3.18 nstrn

```
integer, parameter, public globaldatafun::nstrn =6
```

Number of strain measures/dofs in Timoshenko model.

Definition at line 117 of file GlobalDataFun.f90.

5.20.3.19 pi

```
real(dbl), parameter, public globaldatafun::pi = 3.1415926535897932D0
```

Definition at line 123 of file GlobalDataFun.f90.

5.20.3.20 rad_2_deg

```
real(dbl), parameter, public globaldatafun::rad_2_deg = 5.7295779513082321D1
```

convert radian to degree

Definition at line 125 of file GlobalDataFun.f90.

5.20.3.21 runmod

```
integer, public globaldatafun::runmod =0
```

Define the output behavior of the program; 0: legacy mode of the computation code with output of a .out text file; 1: mode compatible with the python pre/postrpocessor (argument -p in the terminal), 2: silent mode (argument -s in the terminal)

Definition at line 143 of file GlobalDataFun.f90.

5.20.3.22 solver

```
character(10), public globaldatafun::solver ='MUMPS'
```

linear solver used (HSL : ddep.f, mc19.f + ma28 or MUMPS : linux library)

Definition at line 146 of file GlobalDataFun.f90.

5.20.3.23 tolerance

```
real(dbl), parameter, public globaldatafun::tolerance = EPSILON(1.0_DBL)
```

a smart number of the double precision real number

Definition at line 126 of file GlobalDataFun.f90.

5.21 internaldata Module Reference

This module contains the variables needed internally in the program. Not necessary to be defined in the outside environment.

Data Types

- type memberinf

  structure containing the caracteritics of a finite element.

Variables

- logical, parameter debug =.FALSE.
- integer, parameter iout =30
- character(64) deb_name
- integer nsize
- integer, dimension(:,:), allocatable dof_all
- integer, dimension(:,:), allocatable follower_all
- real(dbl), dimension(:,:), allocatable cond_all
- real(dbl), dimension(:,:), allocatable init_memb
- integer init_flag
- real(dbl) two_divide_dt

  2/dt
- integer assemble_flag =0

  for the purpose to share the routines between assembly of stiffness matrix and mass matrix
- integer, dimension(:,:), allocatable index_kp

  the starting row and column for each kp
- integer, dimension(:,:), allocatable index_mb

  the starting row and column for each member
- real(dbl), dimension(ndim) xyz_pt1

  the coordinate of the starting point of the first member
- integer, parameter nzelemmax =500
- integer nemax

5.21.1 Detailed Description

This module contains the variables needed internally in the program. Not necessary to be defined in the outside environment.

5.21.2 Variable Documentation

5.21.2.1 assemble_flag

```
integer internaldata::assemble_flag =0
```

for the purpose to share the routines between assembly of stiffness matrix and mass matrix

Definition at line 47 of file InternalData.f90.

5.21.2.2 cond_all

```
real(dbl), dimension(:,:), allocatable internaldata::cond_all
```

Definition at line 41 of file InternalData.f90.

5.21.2.3 deb_name

```
character(64) internaldata::deb_name
```

Definition at line 30 of file InternalData.f90.

5.21.2.4 debug

```
logical, parameter internaldata::debug =.FALSE.
```

Definition at line 25 of file InternalData.f90.

5.21.2.5 dof_all

```
integer, dimension(:,:), allocatable internaldata::dof_all
```

Definition at line 39 of file InternalData.f90.

5.21.2.6 follower_all

```
integer, dimension(:,:), allocatable internaldata::follower_all
```

Definition at line 40 of file InternalData.f90.

5.21.2.7 index_kp

```
integer, dimension(:,:), allocatable internaldata::index_kp
```

the starting row and column for each kp

Definition at line 48 of file InternalData.f90.

5.21.2.8 index_mb

```
integer, dimension(:,:), allocatable internaldata::index_mb
```

the starting row and column for each member

Definition at line 49 of file InternalData.f90.

5.21.2.9 init_flag

```
integer internaldata::init_flag
```

Definition at line 45 of file InternalData.f90.

5.21.2.10 init_memb

```
real(dbl), dimension(:,:), allocatable internaldata::init_memb
```

Definition at line 43 of file InternalData.f90.

5.21.2.11 iout

```
integer, parameter internaldata::iout =30
```

Definition at line 29 of file InternalData.f90.

5.21.2.12 nemax

`integer internaldata::nemax`

Definition at line 65 of file InternalData.f90.

5.21.2.13 nsize

`integer internaldata::nsize`

Definition at line 34 of file InternalData.f90.

5.21.2.14 nzelemmax

`integer, parameter internaldata::nzelemmax =500`

Definition at line 63 of file InternalData.f90.

5.21.2.15 two_divide_dt

`real(dbl) internaldata::two_divide_dt`

2/dt

Definition at line 46 of file InternalData.f90.

5.21.2.16 xyz_pt1

`real(dbl), dimension(ndim) internaldata::xyz_pt1`

the coordinate of the starting point of the first member

Definition at line 50 of file InternalData.f90.

5.22 ioaero Module Reference

This module handle I/O of the computation code. Allow to read a .dat command file possibly with a .ini file and output a .out text output file or/and a folder with .vtk file intended to be used with paraview.

Functions/Subroutines

- subroutine, public input
- subroutine, public output
- subroutine, public outputvtk

Variables

- integer, parameter, private char_len =256
- integer, parameter in =10
- character(char_len) inp_name
- integer, parameter, public ein =20

  file for echoing the inputs: inp_name.ech
- character(char_len+3) ech_name
- integer, parameter out =40

  file for output: inp_name.out
- character(char_len+3) out_name
- integer, parameter init =50

  file for initial conditions: inp_name.ini
- character(char_len+3) init_name
- integer, public nkp

  number of key points
- integer, public nelem

  total number of elements
- integer, public nmemb

  number of members
- integer, public nmate

  number of cross-sectional properties sets
- integer, public nframe

  number of frames
- integer, public ncond_pt

  number of point conditions for concentrated loads and boundary conditions
- integer, public ndistrfun

  number of distributed functions
- integer, public ncurv

  number of initial curvatures/twists
- integer, public analysis_flag

  0: static analysis; 1: steady state response; 2: transient analysis; 3: eigenvalue analysis
- integer, public nev

  number of frequencies and modeshapes.
- integer, public aero_flag

  0: no aero analasys; 1: stationary aerodynamic, 2: unsteady aerodynamic
- integer, public grav_flag
- integer, public ncond_mb

  number of member conditions for distributed loads
- integer, public ntimefun

  number of time functions
- integer, public niter

  number of maximum iterations
- integer, public nstep

  number of time steps/load steps

- integer, public nvtk

  number of the aerodynamic cycle
- integer, dimension(:,:), allocatable, public member

  member property array: member(nmemb,MEMB_CONST)
- integer, public ndof_el

  dofs per element: 12 for static analysis, 18 for dynamic analysis, +Ns if aero_flag = 3
- integer, dimension(ndim), public omega_a_tf

  time function numbers for the angular velocity of frame a
- integer, dimension(ndim), public v_root_a_tf

  time function numbers for the velocity of the starting point of the first member
- real(dbl), dimension(:,:), allocatable, public coord

  nodal coordinates: coord(nkp,NDIM)
- real(dbl), dimension(:,:,:), allocatable, public material

  flexibility matrix: (nmate,12,6)
- real(dbl), dimension(:,:), allocatable, public aerodyn_coef

  2D aerodynamic coefficient : (nmate,:)
- real(dbl), dimension(:,:,:), allocatable, public frame

  member frames: (nframe,3,3)
- real(dbl), dimension(:,:), allocatable, public distr_fun

  prescribed functions: (ndistrfun,6)
- real(dbl), dimension(:,:), allocatable, public curvature

  curvatures: (ncurv,NDIM)
- real(dbl), dimension(:,:,:), allocatable, public sol_pt

  solutions for points sol_pt(nstep,nkp,NDIM+NDOF_ND)
- real(dbl), dimension(:,:,:), allocatable, public sol_mb

  solutions for member sol_mb(nstep,nelem,NDIM+ndof_el): nelem: total number of elements
- real(dbl), dimension(2), public simu_time

  start and end time of the simulation.
- real(dbl), dimension(ndim), public omega_a0

  the magnitude of angular velocity of frame a
- real(dbl), dimension(ndim), public v_root_a0

  the magnitude of linear velocity of the starting point of the first member
- real(dbl), dimension(:,:), allocatable, public init_cond

  initial conditions: init_cond(nelem,12); init_cond(nelem,1:6) for initial displacements/rotations init_cond(nelem,7:12) for initial velocities init_cond(nelem,13:12+NSTATES) Peters finite state parameter at time t+dt
- real(dbl), dimension(:,:), allocatable, public eigen_val

  arrays for holding eigenvalues and eigenvectors
- real(dbl), dimension(:,:,:), allocatable, public eigen_vec_pt

  arrays for holding eigenvalues and eigenvectors
- real(dbl), dimension(:,:,:), allocatable, public eigen_vec_mb

  arrays for holding eigenvalues and eigenvectors
- type(prescriinf), dimension(:), allocatable, public pt_condition

  prescribed information concentrated at nodes
- type(prescriinf), dimension(:), allocatable, public mb_condition

  prescribed information distributed along beam members
- type(timefunction), dimension(:), allocatable, public time_function

  time functions
- character(char_len), public velocity_str = ''
- integer arpack

  Dummy input variable for ARPACK_MOD.
- integer eigenoutput

  Dummy input variable for EIGEN_OUTPUT.
- character(300), public error

5.22.1 Detailed Description

This module handle I/O of the computation code. Allow to read a .dat command file possibly with a .ini file and output a .out text output file or/and a folder with .vtk file intended to be used with paraview.

5.22.2 Function/Subroutine Documentation

5.22.2.1 input()

```
subroutine, public ioaero::input ( )
```

Definition at line 140 of file IOaero.f90.

5.22.2.2 output()

```
subroutine, public ioaero::output ( )
```

Definition at line 651 of file IOaero.f90.

5.22.2.3 outputvtk()

```
subroutine, public ioaero::outputvtk ( )
```

Definition at line 854 of file IOaero.f90.

5.22.3 Variable Documentation

5.22.3.1 aero_flag

```
integer, public ioaero::aero_flag
```

0: no aero analasys; 1: stationary aerodynamic, 2: unsteady aerodynamic

Definition at line 76 of file IOaero.f90.

5.22.3.2 aerodyn_coef

```
real(dbl), dimension(:,:), allocatable, public ioaero::aerodyn_coef
```

2D aerodynamic coefficient : (nmate,:)

Definition at line 97 of file IOaero.f90.

5.22.3.3 analysis_flag

```
integer, public ioaero::analysis_flag
```

0: static analysis; 1: steady state response; 2: transient analysis; 3: eigenvalue analysis

Definition at line 74 of file IOaero.f90.

5.22.3.4 arpack

```
integer ioaero::arpack  [private]
```

Dummy input variable for ARPACK_MOD.

Definition at line 121 of file IOaero.f90.

5.22.3.5 char_len

```
integer, parameter, private ioaero::char_len =256  [private]
```

Definition at line 43 of file IOaero.f90.

5.22.3.6 coord

```
real(dbl), dimension(:,:), allocatable, public ioaero::coord
```

nodal coordinates: coord(nkp,NDIM)

Definition at line 95 of file IOaero.f90.

5.22.3.7 curvature

`real(dbl), dimension(:,:), allocatable, public ioaero::curvature`

curvatures: (ncurv,NDIM)

Definition at line 100 of file IOaero.f90.

5.22.3.8 distr_fun

`real(dbl), dimension(:,:), allocatable, public ioaero::distr_fun`

prescribed functions: (ndistrfun,6)

Definition at line 99 of file IOaero.f90.

5.22.3.9 ech_name

`character(char_len+3) ioaero::ech_name [private]`

Definition at line 48 of file IOaero.f90.

5.22.3.10 eigen_val

`real(dbl), dimension(:,:), allocatable, public ioaero::eigen_val`

arrays for holding eigenvalues and eigenvectors

Definition at line 110 of file IOaero.f90.

5.22.3.11 eigen_vec_mb

`real(dbl), dimension(:,:,:), allocatable, public ioaero::eigen_vec_mb`

arrays for holding eigenvalues and eigenvectors

Definition at line 112 of file IOaero.f90.

5.22.3.12 eigen_vec_pt

```
real(dbl), dimension(:,:,:), allocatable, public ioaero::eigen_vec_pt
```

arrays for holding eigenvalues and eigenvectors

Definition at line 111 of file IOaero.f90.

5.22.3.13 eigenoutput

```
integer ioaero::eigenoutput [private]
```

Dummy input variable for EIGEN_OUTPUT.

Definition at line 122 of file IOaero.f90.

5.22.3.14 ein

```
integer, parameter, public ioaero::ein =20
```

file for echoing the inputs: inp_name.ech

Definition at line 47 of file IOaero.f90.

5.22.3.15 error

```
character(300), public ioaero::error
```

Definition at line 127 of file IOaero.f90.

5.22.3.16 frame

```
real(dbl), dimension(:,:,:), allocatable, public ioaero::frame
```

member frames: (nframe,3,3)

Definition at line 98 of file IOaero.f90.

5.22.3.17 grav_flag

```
integer, public ioaero::grav_flag
```

Parameters

grav_flag	0: without gravity; 1: with gravity

Definition at line 77 of file IOaero.f90.

5.22.3.18 in

```
integer, parameter ioaero::in =10  [private]
```

Definition at line 44 of file IOaero.f90.

5.22.3.19 init

```
integer, parameter ioaero::init =50  [private]
```

file for initial conditions: inp_name.ini

Definition at line 53 of file IOaero.f90.

5.22.3.20 init_cond

```
real(dbl), dimension(:,:), allocatable, public ioaero::init_cond
```

initial conditions: init_cond(nelem,12); init_cond(nelem,1:6) for initial displacements/rotations init_cond(nelem,7:12) for initial velocities init_cond(nelem,13:12+NSTATES) Peters finite state parameter at time t+dt

Definition at line 106 of file IOaero.f90.

5.22.3.21 init_name

```
character(char_len+3) ioaero::init_name  [private]
```

Definition at line 54 of file IOaero.f90.

5.22.3.22 inp_name

```
character(char_len) ioaero::inp_name [private]
```

Definition at line 45 of file IOaero.f90.

5.22.3.23 material

```
real(dbl), dimension(:,:,:), allocatable, public ioaero::material
```

flexibility matrix: (nmate,12,6)

Definition at line 96 of file IOaero.f90.

5.22.3.24 mb_condition

```
type(prescriinf), dimension(:), allocatable, public ioaero::mb_condition
```

prescribed information distributed along beam members

Definition at line 117 of file IOaero.f90.

5.22.3.25 member

```
integer, dimension(:,:), allocatable, public ioaero::member
```

member property array: member(nmemb,MEMB_CONST)

Definition at line 87 of file IOaero.f90.

5.22.3.26 ncond_mb

```
integer, public ioaero::ncond_mb
```

number of member conditions for distributed loads

Definition at line 82 of file IOaero.f90.

5.22.3.27 ncond_pt

`integer, public ioaero::ncond_pt`

number of point conditions for concentrated loads and boundary conditions

Definition at line 71 of file IOaero.f90.

5.22.3.28 ncurv

`integer, public ioaero::ncurv`

number of initial curvatures/twists

Definition at line 73 of file IOaero.f90.

5.22.3.29 ndistrfun

`integer, public ioaero::ndistrfun`

number of distributed functions

Definition at line 72 of file IOaero.f90.

5.22.3.30 ndof_el

`integer, public ioaero::ndof_el`

dofs per element: 12 for static analysis, 18 for dynamic analysis, +Ns if aero_flag = 3

Definition at line 88 of file IOaero.f90.

5.22.3.31 nelem

`integer, public ioaero::nelem`

total number of elements

Definition at line 67 of file IOaero.f90.

5.22.3.32 nev

```
integer, public ioaero::nev
```

number of frequencies and modeshapes.

Definition at line 75 of file IOaero.f90.

5.22.3.33 nframe

```
integer, public ioaero::nframe
```

number of frames

Definition at line 70 of file IOaero.f90.

5.22.3.34 niter

```
integer, public ioaero::niter
```

number of maximum iterations

Definition at line 84 of file IOaero.f90.

5.22.3.35 nkp

```
integer, public ioaero::nkp
```

number of key points

Parameters

nkp	number of key points

Definition at line 66 of file IOaero.f90.

5.22.3.36 nmate

```
integer, public ioaero::nmate
```

number of cross-sectional properties sets

Definition at line 69 of file IOaero.f90.

5.22.3.37 nmemb

```
integer, public ioaero::nmemb
```

number of members

Definition at line 68 of file IOaero.f90.

5.22.3.38 nstep

```
integer, public ioaero::nstep
```

number of time steps/load steps

Definition at line 85 of file IOaero.f90.

5.22.3.39 ntimefun

```
integer, public ioaero::ntimefun
```

number of time functions

Definition at line 83 of file IOaero.f90.

5.22.3.40 nvtk

```
integer, public ioaero::nvtk
```

number of the aerodynamic cycle

Definition at line 86 of file IOaero.f90.

5.22.3.41 omega_a0

`real(dbl), dimension(ndim), public ioaero::omega_a0`

the magnitude of angular velocity of frame a

Definition at line 104 of file IOaero.f90.

5.22.3.42 omega_a_tf

`integer, dimension(ndim), public ioaero::omega_a_tf`

time function numbers for the angular velocity of frame a

Definition at line 89 of file IOaero.f90.

5.22.3.43 out

`integer, parameter ioaero::out =40 [private]`

file for output: inp_name.out

Definition at line 50 of file IOaero.f90.

5.22.3.44 out_name

`character(char_len+3) ioaero::out_name [private]`

Definition at line 51 of file IOaero.f90.

5.22.3.45 pt_condition

`type(prescriinf), dimension(:), allocatable, public ioaero::pt_condition`

prescribed information concentrated at nodes

Definition at line 116 of file IOaero.f90.

5.22.3.46 simu_time

`real(dbl), dimension(2), public ioaero::simu_time`

start and end time of the simulation.

Definition at line 103 of file IOaero.f90.

5.22.3.47 sol_mb

`real(dbl), dimension(:,:,:), allocatable, public ioaero::sol_mb`

solutions for member sol_mb(nstep,nelem,NDIM+ndof_el): nelem: total number of elements

Definition at line 102 of file IOaero.f90.

5.22.3.48 sol_pt

`real(dbl), dimension(:,:,:), allocatable, public ioaero::sol_pt`

solutions for points sol_pt(nstep,nkp,NDIM+NDOF_ND)

Definition at line 101 of file IOaero.f90.

5.22.3.49 time_function

`type(timefunction), dimension(:), allocatable, public ioaero::time_function`

time functions

Definition at line 118 of file IOaero.f90.

5.22.3.50 v_root_a0

`real(dbl), dimension(ndim), public ioaero::v_root_a0`

the magnitude of linear velocity of the starting point of the first member

Definition at line 105 of file IOaero.f90.

5.22.3.51 v_root_a_tf

```
integer, dimension(ndim), public ioaero::v_root_a_tf
```

time function numbers for the velocity of the starting point of the first member

Definition at line 90 of file IOaero.f90.

5.22.3.52 velocity_str

```
character(char_len), public ioaero::velocity_str = ''
```

Definition at line 120 of file IOaero.f90.

5.23 member Module Reference

This module assemles within a member without considering the particular conditions of the end points.

Functions/Subroutines

- subroutine, public assemblememberrhs (ndof_el, memb_info_i, v_root_a, omega_a, x_memb, rhs_memb, aero_flag, grav_flag, init_cond)

  Assemble the equations for each member ∗.
- subroutine, public assemblememberjacobian (ndof_el, niter, memb_info_i, v_root_a, omega_a, x_memb, nz_memb, irn_memb, jcn_memb, coef_memb, aero_flag, grav_flag)

  Assemble the Jacobian for each member ∗.
- subroutine, public extractmemberproperties (memb_no, memb_info_i, member, ncond_mb, mb_condition, distr_fun, error, init_cond)

  Extract member properties ∗.

Variables

- integer, public ndiv

  number of divisions
- integer, public ncol_memb

  the total number of columns of the member

5.23.1 Detailed Description

This module assemles within a member without considering the particular conditions of the end points.

5.23.2 Function/Subroutine Documentation

5.23.2.1 assemblememberjacobian()

```
subroutine, public member::assemblememberjacobian (
            integer, intent(in) ndof_el,
            integer, intent(in) niter,
            type (memberinf), intent(in) memb_info_i,
            real(dbl), dimension(:), intent(in) v_root_a,
            real(dbl), dimension(:), intent(in) omega_a,
            real(dbl), dimension(:), intent(in) x_memb,
            integer, intent(out) nz_memb,
            integer, dimension(:), intent(out) irn_memb,
            integer, dimension(:), intent(out) jcn_memb,
            real(dbl), dimension(:), intent(out) coef_memb,
            integer, intent(in) aero_flag,
            integer, intent(in) grav_flag )
```

Assemble the Jacobian for each member ∗.

Parameters

in	*ndof_el*	ioaero::ndof_el
in	*niter*	ioaero::niter
in	*aero_flag*	ioaero::aero_flag
in	*grav_flag*	ioaero::grav_flag
in	*memb_↩ info_i*	array containing the characteristics of a beam member
in	*v_root_a*	linear velocity of frame a
in	*omega_a*	angular velocity of frame a
in	*x_memb*	solution vector of a beam member
out	*nz_memb*	Number of nonzero value of the member Jacobian
out	*irn_memb*	line index of the nonzero coefficient
out	*jcn_memb*	column index of the nonzero coefficient
out	*coef_memb*	value of the nonzero coefficient

Definition at line 82 of file Member.f90.

5.23.2.2 assemblememberrhs()

```
subroutine, public member::assemblememberrhs (
            integer, intent(in) ndof_el,
            type (memberinf), intent(in) memb_info_i,
            real(dbl), dimension(:), intent(in) v_root_a,
            real(dbl), dimension(:), intent(in) omega_a,
            real(dbl), dimension(:), intent(in) x_memb,
            real(dbl), dimension(:), intent(out) rhs_memb,
            integer, intent(in) aero_flag,
            integer, intent(in) grav_flag,
            real(dbl), dimension(:,:), intent(in), optional init_cond )
```

Assemble the equations for each member ∗.

Parameters

in	*ndof_el*	ioaero::ndof_el
in	*aero_flag*	ioaero::aero_flag
in	*grav_flag*	ioaero::grav_flag
in	*memb_info_i*	array containing the characteristics of a beam member
in	*v_root_a*	linear velocity of frame a
in	*omega_a*	angular velocity of frame a
in	*x_memb*	solution vector of a beam member
out	*rhs_memb*	RHS vector of a beam member
in	*init_cond*	ioaero::init_cond

Definition at line 38 of file Member.f90.

5.23.2.3 extractmemberproperties()

```
subroutine, public member::extractmemberproperties (
            integer, intent(in) memb_no,
            type (memberinf), intent(in) memb_info_i,
            integer, dimension(:,:), intent(in) member,
            integer, intent(in) ncond_mb,
            type(prescriinf), dimension(:), intent(in) mb_condition,
            real(dbl), dimension(:,:), intent(in) distr_fun,
            character(*), intent(out) error,
            real(dbl), dimension(:,:), intent(in), optional init_cond )
```

Extract member properties ∗.

Parameters

in	*memb_no*	Number of the current member
in	*ncond_mb*	ioaero::ncond_mb
in	*member*	ioaero::member
in	*memb_info_i*	array containing the characteristics of a beam member
in	*distr_fun*	ioaero::distr_fun
in	*mb_condition*	ioaero::mb_condition
out	*error*	ioaero::error
in	*init_cond*	ioaero::init_cond

Definition at line 149 of file Member.f90.

5.23.3 Variable Documentation

5.23.3.1 ncol_memb

```
integer, public member::ncol_memb
```

the total number of columns of the member

Definition at line 27 of file Member.f90.

5.23.3.2 ndiv

```
integer, public member::ndiv
```

number of divisions

Definition at line 26 of file Member.f90.

5.24 prepromodule Module Reference

This module preprocess the finite element model including connectivity and member information. This information are time step indepedent.

Functions/Subroutines

- subroutine, public preprocess (nkp, nelem, ndof_el, member, material, frame, coord, curvature, dof_con, memb_info, error, aero_flag, grav_flag, aerodyn_coef)

  Obtaining the connection condition for each key point ∗ if a point is connected to more than one member, it is ∗ a connection point, otherwise it is a boundary point. ∗ It also calculates the size of the problem ∗.
- subroutine memberproperties (memb_no, ndof_el, member, material, frame, coord, curvature, memb_info_i, error, aero_flag, grav_flag, aerodyn_coef)

  Extract member properties for each division ∗.
- real(dbl) function curvebeamfun (kn, mL, kn2, k12, kn4, xvar)

  Function for evaluating arc length of initially curved and twisted beams.
- real(dbl) function rtbis (func, kn, mL, kn2, k12, kn4, x1, x2, xacc, maxit, error)

  Use biosection to find root of a function ∗ from the book of Numerical Recipes ∗.

Variables

- integer ndiv

  member::ndiv
- integer ncol_memb

  member::ncol_memb

5.24.1 Detailed Description

This module preprocess the finite element model including connectivity and member information. This information are time step indepedent.

5.24.2 Function/Subroutine Documentation

5.24.2.1 curvebeamfun()

```
real(dbl) function prepromodule::curvebeamfun (
            real(dbl), intent(in) kn,
            real(dbl), intent(in) mL,
            real(dbl), intent(in) kn2,
            real(dbl), intent(in) k12,
            real(dbl), intent(in) kn4,
            real(dbl), intent(in) xvar )  [private]
```

Function for evaluating arc length of initially curved and twisted beams.

Definition at line 337 of file Preprocess.f90.

5.24.2.2 memberproperties()

```
subroutine prepromodule::memberproperties (
            integer, intent(in) memb_no,
            integer, intent(in) ndof_el,
            integer, dimension(:,:), intent(in) member,
            real(dbl), dimension(:,:,:), intent(in) material,
            real(dbl), dimension(:,:,:), intent(in) frame,
            real(dbl), dimension(:,:), intent(in) coord,
            real(dbl), dimension(:,:), intent(in) curvature,
            type (memberinf), intent(out) memb_info_i,
            character(*), intent(out) error,
            integer, intent(in) aero_flag,
            integer, intent(in) grav_flag,
            real(dbl), dimension(:,:), intent(in), optional aerodyn_coef )  [private]
```

Extract member properties for each division $*$.

Parameters

in	*memb_no*	member index
in	*ndof_el*	ioaero::ndof_el
in	*member*	ioaero::member
in	*aero_flag*	ioaero::aero_flag
in	*grav_flag*	ioaero::grav_flag
in	*material*	ioaero::material
in	*frame*	ioaero::frame
in	*coord*	ioaero::coord
in	*curvature*	ioaero::curvature
in	*aerodyn_coef*	ioaero::aerodyn_coef
out	*memb_info_i*	the characteristics of the ith beam member
out	*error*	ioaero::error

Definition at line 188 of file Preprocess.f90.

5.24.2.3 preprocess()

```
subroutine, public prepromodule::preprocess (
            integer, intent(in) nkp,
            integer, intent(in) nelem,
            integer, intent(in) ndof_el,
            integer, dimension(:,:), intent(in) member,
            real(dbl), dimension(:,:,:), intent(inout) material,
            real(dbl), dimension(:,:,:), intent(in) frame,
            real(dbl), dimension(:,:), intent(in) coord,
            real(dbl), dimension(:,:), intent(in) curvature,
            integer, dimension(:), intent(out) dof_con,
            type (memberinf), dimension(:), intent(out) memb_info,
            character(*), intent(out) error,
            integer, intent(in) aero_flag,
            integer, intent(in) grav_flag,
            real(dbl), dimension(:,:), intent(in), optional aerodyn_coef )
```

Obtaining the connection condition for each key point ∗ if a point is connected to more than one member, it is ∗ a connection point, otherwise it is a boundary point. ∗ It also calculates the size of the problem ∗.

Parameters

in	*nkp*	ioaero::nkp
in	*nelem*	ioaero::nelem
in	*aero_flag*	ioaero::aero_flag
in	*grav_flag*	ioaero::grav_flag
in	*ndof_el*	ioaero::ndof_el
in	*member*	ioaero::member
in,out	*material*	ioaero::material
in	*frame*	ioaero::frame
in	*coord*	ioaero::coord
in	*curvature*	ioaero::curvature
in	*aerodyn_coef*	ioaero::aerodyn_coef
out	*dof_con*	the connecting condition for key point.
out	*memb_info*	the member parameters of the whole structure

Definition at line 43 of file Preprocess.f90.

5.24.2.4 rtbis()

```
real(dbl) function prepromodule::rtbis (
             func,
            real(dbl), intent(in) kn,
            real(dbl), intent(in) mL,
```

```
real(dbl), intent(in) kn2,
real(dbl), intent(in) k12,
real(dbl), intent(in) kn4,
real(dbl), intent(in) x1,
real(dbl), intent(in) x2,
real(dbl), intent(in) xacc,
integer, intent(in) maxit,
character(*), intent(out) error ) [private]
```

Use biosection to find root of a function ∗ from the book of Numerical Recipes ∗.

Definition at line 368 of file Preprocess.f90.

5.24.3 Variable Documentation

5.24.3.1 ncol_memb

```
integer prepromodule::ncol_memb [private]
```

member::ncol_memb

Definition at line 28 of file Preprocess.f90.

5.24.3.2 ndiv

```
integer prepromodule::ndiv [private]
```

member::ndiv

Definition at line 27 of file Preprocess.f90.

5.25 prescribedcondition Module Reference

A module for defining prescribed conditions including both concentrated information and distributed information.

Data Types

- type distriload

  Define the distributed load condition.
- type prescriinf

  Define the prescribed condition.

Functions/Subroutines

- subroutine, public existpi (location, prescri_inf, exist_pi, follower_pi)

  Determine whether prescribed condition exist, and whether any of such conditions is a follower condition.
- type(distriload) function, public getdistributedload (memb_no, mb_condition, distr_fun)

  Obtain the distributed load condition.
- real(dbl) function, dimension(nstrn), public getload (flag, dL, Le, eCT, load, follower_load)

  Obtain the distributed load, transform if follower.
- real(dbl) function, dimension(nstrn, 3), public getloadj (flag, dL, Le, eCTtheta, load)

  Obtain the jacobian due to follower distributed load $*$.
- subroutine, public getprescribeddof (nkp, pt_condition, kp_dof, kp_follower)

  Obtain Prescribed dof and follower condition $*$.
- subroutine, public getprescribedval (nkp, pt_condition, kp_cond)

  Obtain Prescribed value $*$.
- elemental type(prescriinf) function, public initpi ()

  Initialize Prescribed Conditions $*$.
- type(prescriinf) function, public inputechoprescribedconditions (IN, EIN, error)

  Input and echo Prescribed Conditions $*$.
- real(dbl) function, dimension(nstrn), public updatefollower (kp_dof, kp_follower, kp_cond, x_pt)

  Obtain Prescribed DOF and value needed for rhs assume only follower force/moments, and no displacements or rotations can be prescribed for follower quantities. And the first three prescribed dofs for the point with a follower component should be either 7 8 9 or 10 11 12. This assumption is made for the easiness to locate the rotation parameters.
- subroutine, public updatepi (prescri_inf, time_fun, t)

  Update the prescribed information based on the current time the value is stored in: value_current.
- real(dbl) function, dimension(size(vec), size(vec)), public followerj (follower, vec, eCTtheta)

  Calculating the Jacobian due to follower conditions $J = diffC^T.vec/diff\theta$*, return a 3x3 matrix with ith column corresponding to the derivative withe respect to* θ_i.
- real(dbl) function loadintegration (flag, dL, Le, func)

  Caculate the load using Chebychev polynomials $*$.
- real(dbl) function, dimension(size(vec)) transferfollower (follower, vec, CT)

  Transfer follower according to C^T.vec note vec is a 3x1 vector, and whether a component is a follower or not is determined by follower.
- elemental type(prescriinf) function, public initpiaero (i)

5.25.1 Detailed Description

A module for defining prescribed conditions including both concentrated information and distributed information.

5.25.2 Function/Subroutine Documentation

5.25.2.1 existpi()

```
subroutine, public prescribedcondition::existpi (
            integer, intent(in) location,
            type(prescriinf), dimension(:), intent(in) prescri_inf,
            logical, intent(out) exist_pi,
            logical, intent(out) follower_pi )
```

Determine whether prescribed condition exist, and whether any of such conditions is a follower condition.

Definition at line 59 of file PrescribedCondition.f90.

5.25.2.2 followerj()

```
real(dbl) function, dimension(size(vec),size(vec)), public prescribedcondition::followerj (
            integer, dimension(:), intent(in) follower,
            real(dbl), dimension(:), intent(in) vec,
            real(dbl), dimension(:,:,:), intent(in) eCTtheta )
```

Calculating the Jacobian due to follower conditions $J = diffC^T.vec/diff\theta$, return a 3x3 matrix with ith column corresponding to the derivative withe respect to θ_i.

Definition at line 385 of file PrescribedCondition.f90.

5.25.2.3 getdistributedload()

```
type(distriload) function, public prescribedcondition::getdistributedload (
            integer, intent(in) memb_no,
            type(prescriinf), dimension(:), intent(in) mb_condition,
            real(dbl), dimension(:,:), intent(in) distr_fun )
```

Obtain the distributed load condition.

Parameters

in	*memb_no*	index of the beam member
in	*distr_fun*	ioaero::distr_fun
in	*mb_condition*	distributed load information

Definition at line 83 of file PrescribedCondition.f90.

5.25.2.4 getload()

```
real(dbl) function, dimension(nstrn), public prescribedcondition::getload (
            integer, intent(in) flag,
            real(dbl), intent(in) dL,
            real(dbl), intent(in) Le,
            real(dbl), dimension(:,:), intent(in) eCT,
            type (distriload), intent(in) load,
            logical, intent(in) follower_load )
```

Obtain the distributed load, transform if follower.

Parameters

in	*flag*	if flag=-1, the starting portion, if flag=1, the ending portion

Definition at line 116 of file PrescribedCondition.f90.

5.25.2.5 getloadj()

```
real(dbl) function, dimension(nstrn,3), public prescribedcondition::getloadj (
            integer, intent(in) flag,
            real(dbl), intent(in) dL,
            real(dbl), intent(in) Le,
            real(dbl), dimension(:,:,:), intent(in) eCTtheta,
            type (distriload), intent(in) load )
```

Obtain the jacobian due to follower distributed load ∗.

Definition at line 149 of file PrescribedCondition.f90.

5.25.2.6 getprescribeddof()

```
subroutine, public prescribedcondition::getprescribeddof (
            integer, intent(in) nkp,
            type(prescriinf), dimension(:), intent(in) pt_condition,
            integer, dimension(:,:), intent(out) kp_dof,
            integer, dimension(:,:), intent(out) kp_follower )
```

Obtain Prescribed dof and follower condition ∗.

Parameters

in	*nkp*	ioaero::nkp

Definition at line 183 of file PrescribedCondition.f90.

5.25.2.7 getprescribedval()

```
subroutine, public prescribedcondition::getprescribedval (
            integer, intent(in) nkp,
            type(prescriinf), dimension(:), intent(in) pt_condition,
            real(dbl), dimension(:,:), intent(out) kp_cond )
```

Obtain Prescribed value ∗.

Parameters

in	*nkp*	ioaero::nkp
in	*pt_condition*	ioaero::pt_condition

Definition at line 212 of file PrescribedCondition.f90.

5.25.2.8 initpi()

```
elemental type (prescriinf) function, public prescribedcondition::initpi ( )
```

Initialize Prescribed Conditions ∗.

Definition at line 237 of file PrescribedCondition.f90.

5.25.2.9 initpiaero()

```
elemental type (prescriinf) function, public prescribedcondition::initpiaero (
            integer, intent(in) i )
```

Definition at line 494 of file PrescribedCondition.f90.

5.25.2.10 inputechoprescribedconditions()

```
type (prescriinf) function, public prescribedcondition::inputechoprescribedconditions (
            integer, intent(in) IN,
            integer, intent(in) EIN,
            character(*), intent(out) error )
```

Input and echo Prescribed Conditions ∗.

Definition at line 258 of file PrescribedCondition.f90.

5.25.2.11 loadintegration()

```
real(dbl) function prescribedcondition::loadintegration (
            integer, intent(in) flag,
            real(dbl), intent(in) dL,
            real(dbl), intent(in) Le,
            real(dbl), dimension(nstrn), intent(in) func )  [private]
```

Caculate the load using Chebychev polynomials ∗.

Parameters

in	*flag*	if flag=-1, it is for the starting point, if flag=1, it is for the ending point
in	*dl*	length of the element
in	*le*	the ending point of the element at the arc length of the member
in	*func*	distributed load function for this element

Returns

the load for each element

Definition at line 413 of file PrescribedCondition.f90.

5.25.2.12 transferfollower()

```
real(dbl) function, dimension(size(vec)) prescribedcondition::transferfollower (
            integer, dimension(:), intent(in) follower,
            real(dbl), dimension(:), intent(in) vec,
            real(dbl), dimension(:,:), intent(in) CT )  [private]
```

Transfer follower according to C^T.vec note vec is a 3x1 vector, and whether a component is a follower or not is determined by follower.

Definition at line 472 of file PrescribedCondition.f90.

5.25.2.13 updatefollower()

```
real(dbl) function, dimension(nstrn), public prescribedcondition::updatefollower (
            integer, dimension(:), intent(in) kp_dof,
            integer, dimension(:), intent(in) kp_follower,
            real(dbl), dimension(:), intent(in) kp_cond,
            real(dbl), dimension(:), intent(in) x_pt )
```

Obtain Prescribed DOF and value needed for rhs assume only follower force/moments, and no displacements or rotations can be prescribed for follower quantities. And the first three prescribed dofs for the point with a follower component should be either 7 8 9 or 10 11 12. This assumption is made for the easiness to locate the rotation parameters.

Definition at line 309 of file PrescribedCondition.f90.

5.25.2.14 updatepi()

```
subroutine, public prescribedcondition::updatepi (
            type (prescriinf), dimension(:), intent(inout) prescri_inf,
            type (timefunction), dimension(:), intent(in) time_fun,
            real(dbl), intent(in) t )
```

Update the prescribed information based on the current time the value is stored in: value_current.

Definition at line 341 of file PrescribedCondition.f90.

5.26 solvemumps Module Reference

This module contains the linear & nonlinear solver interfaced with MUMPS direct solver library.

Functions/Subroutines

- subroutine, public linearsolutionmumps (ndof_el, memb_info, v_root_a, omega_a, member, error, ncond_mb, mb_condition, distr_fun, dof_con, x, aero_flag, grav_flag, init_cond)

  The linear solver is basically the Newton-Raphson with ∗ initial guess equal to zero and only uses one iterations ∗.
- subroutine, public newtonraphsonmumps (ndof_el, memb_info, v_root_a, omega_a, member, niter, error, ncond_mb, mb_condition, distr_fun, dof_con, x, aero_flag, grav_flag, init_cond)

  Use Newton-Raphson method to solve the nonlinear system ∗.
- subroutine, public extractsolution (ndof_el, member, coord, memb_info, x, dof_con, sol_pt_i, sol_mb_i)

  The subroutine extracts the solution for each key point and each member from the solution vector ∗.
- subroutine, public extractelementvalues (ndof_el, member, x, sol_mb_i)

  The subroutine extracts elemental values from ∗ the solution vector ∗.
- subroutine, public insertelementvalues (ndof_el, member, x, init_cond)

  The subroutine insert the elemental values into the solution vector: needed for initial guess for starting time marching: replace the first six valumes for each element with given initial conditions ∗.
- real(dbl) function, dimension(size(sol_mb_i, 1), 6), public ctcabph (niter, member, memb_info, sol_mb_i)

  Transfer the vector PH (linear and angular momenta) form frame B to frame a.

Variables

- type(dmumps_struc) mumps_par

  an array containing the configuration parameters of MUMPS solver
- integer ierr

  error code of the MUMPS solver

5.26.1 Detailed Description

This module contains the linear & nonlinear solver interfaced with MUMPS direct solver library.

5.26.2 Function/Subroutine Documentation

5.26.2.1 ctcabph()

```
real(dbl) function, dimension(size(sol_mb_i,1),6), public solvemumps::ctcabph (
            integer, intent(in) niter,
            integer, dimension(:,:), intent(in) member,
            type (memberinf), dimension(:), intent(in) memb_info,
            real(dbl), dimension(:,:), intent(in) sol_mb_i )
```

Transfer the vector PH (linear and angular momenta) form frame B to frame a.

Parameters

in	*niter*	ioaero::niter
in	*member*	ioaero::member
in	*sol_↩ mb_i*	solutions for members for ith step

Definition at line 534 of file SolveMumps.f90.

5.26.2.2 extractelementvalues()

```
subroutine, public solvemumps::extractelementvalues (
            integer, intent(in) ndof_el,
            integer, dimension(:,:), intent(in) member,
            real(dbl), dimension(:), intent(in) x,
            real(dbl), dimension(:,:), intent(out) sol_mb_i )
```

The subroutine extracts elemental values from $*$ the solution vector $*$.

Parameters

in	*ndof_el*	ioaero::ndof_el
in	*member*	ioaero::member
in	*x*	the solution vector
out	*sol_↩ mb_i*	solutions for all the elements for ith step

Definition at line 463 of file SolveMumps.f90.

5.26.2.3 extractsolution()

```
subroutine, public solvemumps::extractsolution (
            integer, intent(in) ndof_el,
            integer, dimension(:,:), intent(in) member,
            real(dbl), dimension(:,:), intent(in) coord,
            type (memberinf), dimension(:), intent(in) memb_info,
            real(dbl), dimension(:), intent(in) x,
            integer, dimension(:), intent(in) dof_con,
            real(dbl), dimension(:,:), intent(out) sol_pt_i,
            real(dbl), dimension(:,:), intent(out) sol_mb_i )
```

The subroutine extracts the solution for each key point and each member from the solution vector $*$.

Parameters

in	*ndof_el*	ioaero::ndof_el
in	*member*	ioaero::member

Parameters

in	*coord*	ioaero::coord
in	*memb_info*	contains the member parameters of the whole structure
in	*x*	the solution vector
in	*dof_con*	the connecting condition for key point.
out	*sol_pt_i*	solutions for points for ith step
out	*sol_mb_i*	solutions for members for ith step

Definition at line 383 of file SolveMumps.f90.

5.26.2.4 insertelementvalues()

```
subroutine, public solvemumps::insertelementvalues (
            integer, intent(in) ndof_el,
            integer, dimension(:,:), intent(in) member,
            real(dbl), dimension(:), intent(inout) x,
            real(dbl), dimension(:,:), intent(in) init_cond )
```

The subroutine insert the elemental values into the solution vector: needed for initial guess for starting time marching: replace the first six valumes for each element with given initial conditions ∗.

Parameters

in	*ndof_el*	ioaero::ndof_el
in	*member*	#ioaero::memeber
in,out	*x*	the solution vector
in	*init_cond*	ioaero::init_cond

Definition at line 500 of file SolveMumps.f90.

5.26.2.5 linearsolutionmumps()

```
subroutine, public solvemumps::linearsolutionmumps (
            integer, intent(in) ndof_el,
            type (memberinf), dimension(:), intent(in) memb_info,
            real(dbl), dimension(:), intent(in) v_root_a,
            real(dbl), dimension(:), intent(in) omega_a,
            integer, dimension(:,:), intent(in) member,
            character(*), intent(out) error,
            integer, intent(in) ncond_mb,
            type(prescriinf), dimension(:), intent(in) mb_condition,
            real(dbl), dimension(:,:), intent(in) distr_fun,
            integer, dimension(:)  dof_con,
            real(dbl), dimension(:), intent(out) x,
            integer, intent(in) aero_flag,
```

```
integer, intent(in) grav_flag,
real(dbl), dimension(:,:), intent(in), optional init_cond )
```

The linear solver is basically the Newton-Raphson with $*$ initial guess equal to zero and only uses one iterations $*$.

Parameters

in	*ndof_el*	ioaero::ndof_el
in	*aero_flag*	ioaero::aero_flag
in	*grav_flag*	ioaero::grav_flag
in	*v_root_a*	linear velocity of frame a
in	*omega_a*	angular velocity of frame a
in	*distr_fun*	ioaero::distr_fun
in	*member*	ioaero::member
in	*ncond_mb*	ioaero::ncond_mb
in	*mb_condition*	ioaero::mb_condition
	dof_con	this array is passed by value
out	*error*	ioaero::error
out	*x*	The solution vector
in	*init_cond*	ioaero::init_cond

Definition at line 42 of file SolveMumps.f90.

5.26.2.6 newtonraphsonmumps()

```
subroutine, public solvemumps::newtonraphsonmumps (
              integer, intent(in) ndof_el,
              type (memberinf), dimension(:), intent(in) memb_info,
              real(dbl), dimension(:), intent(in) v_root_a,
              real(dbl), dimension(:), intent(in) omega_a,
              integer, dimension(:,:), intent(in) member,
              integer, intent(in) niter,
              character(*), intent(out) error,
              integer, intent(in) ncond_mb,
              type(prescriinf), dimension(:), intent(in) mb_condition,
              real(dbl), dimension(:,:), intent(in) distr_fun,
              integer, dimension(:)  dof_con,
              real(dbl), dimension(:), intent(inout) x,
              integer, intent(in) aero_flag,
              integer, intent(in) grav_flag,
              real(dbl), dimension(:,:), intent(in), optional init_cond )
```

Use Newton-Raphson method to solve the nonlinear system $*$.

Parameters

in	*ndof_el*	ioaero::ndof_el
in	*aero_flag*	ioaero::aero_flag
in	*grav_flag*	ioaero::grav_flag
in	*memb_info*	contains the member parameters of the whole structure
in	*v_root_a*	linear velocity of frame a
in	*omega_a*	angular velocity of frame a
in	*distr_fun*	ioaero::distr_fun
in	*member*	ioaero::member
in	*niter*	ioaero::niter

Parameters

in	*ncond_mb*	ioaero::ncond_mb
in	*mb_condition*	ioaero::mb_condition
in,out	*x*	The solution vector
out	*error*	ioaero::error
in	*init_cond*	ioaero::init_cond

Definition at line 141 of file SolveMumps.f90.

5.26.3 Variable Documentation

5.26.3.1 ierr

```
integer solvemumps::ierr
```

error code of the MUMPS solver

Definition at line 25 of file SolveMumps.f90.

5.26.3.2 mumps_par

```
type (dmumps_struc) solvemumps::mumps_par
```

an array containing the configuration parameters of MUMPS solver

Definition at line 24 of file SolveMumps.f90.

5.27 system Module Reference

This module assemles the system including the coefficient matrix (jacobian matrix) and the right hand side (negative of the equation values) ∗.

Functions/Subroutines

- subroutine, public assemblejacobian (ndof_el, niter, memb_info, v_root_a, omega_a, member, error, ncond↩_mb, mb_condition, distr_fun, dof_con, x, aero_flag, grav_flag, init_cond)

  Assemble the coefficient matrix of the beam system ∗.
- subroutine, public assemblerhs (ndof_el, memb_info, v_root_a, omega_a, member, error, ncond_mb, mb_↩condition, distr_fun, dof_con, x, rhs, aero_flag, grav_flag, init_cond)

  Assemble the right hand side.
- subroutine pointfollowerj (flag, nrow, ncol, eCTtheta)

  Add the contribution to Jacobian matrix due to follower point force or moments.

Variables

- real(dbl), dimension(nstrn) x_pt
    the solution from the previous step for a key point.
- integer, dimension(nstrn) kp_dof
    prescribed dof
- real(dbl), dimension(nstrn) kp_cond
    prescribed value
- integer, dimension(nstrn) kp_follower
    follower condition
- integer, public ne
    Number of nonzero coefficients.
- integer, dimension(:), allocatable, public irn
    line index of nonzero coefficients
- integer, dimension(:), allocatable, public jcn
    column index of nonzero coefficients
- real(dbl), dimension(:), allocatable, public coef
    value of nonzero coefficients

5.27.1 Detailed Description

This module assemles the system including the coefficient matrix (jacobian matrix) and the right hand side (negative of the equation values) ∗.

5.27.2 Function/Subroutine Documentation

5.27.2.1 assemblejacobian()

```
subroutine, public system::assemblejacobian (
            integer, intent(in) ndof_el,
            integer, intent(in) niter,
            type (memberinf), dimension(:), intent(in) memb_info,
            real(dbl), dimension(:), intent(in) v_root_a,
            real(dbl), dimension(:), intent(in) omega_a,
            integer, dimension(:,:), intent(in) member,
            character(*), intent(out) error,
            integer, intent(in) ncond_mb,
            type(prescriinf), dimension(:), intent(in) mb_condition,
            real(dbl), dimension(:,:), intent(in) distr_fun,
            integer, dimension(:)  dof_con,
            real(dbl), dimension(:), intent(in) x,
            integer, intent(in) aero_flag,
            integer, intent(in) grav_flag,
            real(dbl), dimension(:,:), intent(in), optional init_cond )
```

Assemble the coefficient matrix of the beam system ∗.

Parameters

in	*ndof_el*	ioaero::ndof_el
in	*niter*	ioaero::niter
in	*aero_flag*	ioaero::aero_flag
in	*grav_flag*	ioaero::grav_flag
in	*v_root_a*	linear velocity of frame a
in	*omega_a*	angular velocity of frame a
in	*memb_info*	contains the member parameters of the whole structure
in	*distr_fun*	ioaero::distr_fun
in	*member*	ioaero::member
in	*ncond_mb*	ioaero::ncond_mb
	dof_con	note dof_con is passed by value, hence what is changed in this subroutine will not affect the original value.
in	*x*	solution vector
in	*mb_condition*	ioaero::mb_condition
out	*error*	ioaero::error
in	*init_cond*	ioaero::init_cond

Definition at line 48 of file System.f90.

5.27.2.2 assemblerhs()

```
subroutine, public system::assemblerhs (
             integer, intent(in) ndof_el,
             type (memberinf), dimension(:), intent(in) memb_info,
             real(dbl), dimension(:), intent(in) v_root_a,
             real(dbl), dimension(:), intent(in) omega_a,
             integer, dimension(:,:), intent(in) member,
             character(*), intent(out) error,
             integer, intent(in) ncond_mb,
             type(prescriinf), dimension(:), intent(in) mb_condition,
             real(dbl), dimension(:,:), intent(in) distr_fun,
             integer, dimension(:)  dof_con,
             real(dbl), dimension(:), intent(in) x,
             real(dbl), dimension(:), intent(out) rhs,
             integer, intent(in) aero_flag,
             integer, intent(in) grav_flag,
             real(dbl), dimension(:,:), intent(in), optional init_cond )
```

Assemble the right hand side.

Parameters

in	*ndof_el*	ioaero::ndof_el
in	*aero_flag*	ioaero::aero_flag
in	*grav_flag*	ioaero::grav_flag
in	*memb_info*	contains the member parameters of the whole structure
in	*v_root_a*	linear velocity of frame a
in	*omega_a*	angular velocity of frame a

Parameters

in	*distr_fun*	ioaero::distr_fun
in	*member*	ioaero::member
in	*ncond_mb*	ioaero::ncond_mb
in	*x*	solution vector
in	*mb_condition*	ioaero::mb_condition
in	*init_cond*	ioaero::init_cond

Definition at line 321 of file System.f90.

5.27.2.3 pointfollowerj()

```
subroutine system::pointfollowerj (
            integer, intent(in) flag,
            integer, intent(in) nrow,
            integer, intent(in) ncol,
            real(dbl), dimension(:,:,:), intent(in) eCTtheta )  [private]
```

Add the contribution to Jacobian matrix due to follower point force or moments.

Definition at line 502 of file System.f90.

5.27.3 Variable Documentation

5.27.3.1 coef

```
real(dbl), dimension(:), allocatable, public system::coef
```

value of nonzero coefficients

Definition at line 35 of file System.f90.

5.27.3.2 irn

```
integer, dimension(:), allocatable, public system::irn
```

line index of nonzero coefficients

Definition at line 33 of file System.f90.

5.27.3.3 jcn

```
integer, dimension(:), allocatable, public system::jcn
```

column index of nonzero coefficients

Definition at line 34 of file System.f90.

5.27.3.4 kp_cond

```
real(dbl), dimension(nstrn) system::kp_cond [private]
```

prescribed value

Definition at line 29 of file System.f90.

5.27.3.5 kp_dof

```
integer, dimension(nstrn) system::kp_dof [private]
```

prescribed dof

Definition at line 28 of file System.f90.

5.27.3.6 kp_follower

```
integer, dimension(nstrn) system::kp_follower [private]
```

follower condition

Definition at line 30 of file System.f90.

5.27.3.7 ne

```
integer, public system::ne
```

Number of nonzero coefficients.

Definition at line 32 of file System.f90.

5.27.3.8 x_pt

```
real(dbl), dimension(nstrn) system::x_pt  [private]
```

the solution from the previous step for a key point.

Definition at line 27 of file System.f90.

5.28 timefunctionmodule Module Reference

A module for defining time functions needed for both prescribed concentrated and distributed conditions.

Data Types

- type timefunction

Functions/Subroutines

- real(dbl) function, public gettimefunction (tf, t)

    get the time function value for any arbitrary time from a piecewise linear function or harmonic function decide where t is located, then interpret the value
- elemental type(timefunction) function, public inittf ()

    Initialize the time function.
- type(timefunction) function, public inputechotimefunctions (IN, EIN, error)

    Input and echo Time Functions.
- real(dbl) function, dimension(size(vec)), public currentvalues (vec, vec_tf, time_fun, time_current)

    Evaluate current function based on magnitude and time.

5.28.1 Detailed Description

A module for defining time functions needed for both prescribed concentrated and distributed conditions.

5.28.2 Function/Subroutine Documentation

5.28.2.1 currentvalues()

```
real(dbl) function, dimension(size(vec)), public timefunctionmodule::currentvalues (
            real(dbl), dimension(:), intent(in) vec,
            integer, dimension(:), intent(in) vec_tf,
            type (timefunction), dimension(:), intent(in) time_fun,
            real(dbl), intent(in) time_current )
```

Evaluate current function based on magnitude and time.

Definition at line 201 of file TimeFunction.f90.

5.28.2.2 gettimefunction()

```
real(dbl) function, public timefunctionmodule::gettimefunction (
            type (timefunction), intent(in) tf,
            real(dbl), intent(in) t )
```

get the time function value for any arbitrary time from a piecewise linear function or harmonic function decide where t is located, then interpret the value

Definition at line 43 of file TimeFunction.f90.

5.28.2.3 inittf()

```
elemental type (timefunction) function, public timefunctionmodule::inittf ( )
```

Initialize the time function.

Definition at line 103 of file TimeFunction.f90.

5.28.2.4 inputechotimefunctions()

```
type (timefunction) function, public timefunctionmodule::inputechotimefunctions (
            integer, intent(in) IN,
            integer, intent(in) EIN,
            character(*), intent(out) error )
```

Input and echo Time Functions.

Parameters

in	*ein*	file units for input anf echo files, respectively

Definition at line 123 of file TimeFunction.f90.

Chapter 6

Data Type Documentation

6.1 gebtaero.CompositeBox.CompositeBox Class Reference

Class interfacing the solver with 3D FEM calculix computation to obtain the cross section parameter from a composite box.

Public Member Functions

- def __init__ (self, Left, Right, Up, Down, Width, Height, OffsetY=0., OffsetZ=0.)
- def GetOffsets (self)
- def GetWidth (self)
- def GetHeight (self)
- def CreateFbdFile (self, TypeElem, NbElemX, NbElemYZ, NbElemPly)
- def CreateInpFile (self, Stress=False, PlaneSection=False, Disp=0)
- def ComputeMassMatrix (self, OffsetY=0., OffsetZ=0.)
- def CreatePeriodicEq (self)
- def DisplaySectionDeformation (self, TypeElem, NbElemX, NbElemY, NbElemPly, DefType, Plane↩ Section=False)

Public Attributes

- Left
- Right
- Up
- Down
- Width
- Height
- OffsetY
- OffsetZ

6.1.1 Detailed Description

Class interfacing the solver with 3D FEM calculix computation to obtain the cross section parameter from a composite box.

```
Class interfacing the solver with 3D FEM calculix computation to obtain stiffness matrix
from a composite box
```

Definition at line 9 of file CompositeBox.py.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 __init__()

```
def gebtaero.CompositeBox.CompositeBox.__init__ (
              self,
              Left,
              Right,
              Up,
              Down,
              Width,
              Height,
              OffsetY = 0.,
              OffsetZ = 0. )
```

Definition at line 14 of file CompositeBox.py.

6.1.3 Member Function Documentation

6.1.3.1 ComputeMassMatrix()

```
def gebtaero.CompositeBox.CompositeBox.ComputeMassMatrix (
              self,
              OffsetY = 0.,
              OffsetZ = 0. )
```

Definition at line 288 of file CompositeBox.py.

6.1.3.2 CreateFbdFile()

```
def gebtaero.CompositeBox.CompositeBox.CreateFbdFile (
              self,
              TypeElem,
              NbElemX,
              NbElemYZ,
              NbElemPly )
```

Definition at line 37 of file CompositeBox.py.

6.1.3.3 CreateInpFile()

```
def gebtaero.CompositeBox.CompositeBox.CreateInpFile (
              self,
              Stress = False,
              PlaneSection = False,
              Disp = 0 )
```

Definition at line 162 of file CompositeBox.py.

6.1.3.4 CreatePeriodicEq()

```
def gebtaero.CompositeBox.CompositeBox.CreatePeriodicEq (
              self )
```

Definition at line 317 of file CompositeBox.py.

6.1.3.5 DisplaySectionDeformation()

```
def gebtaero.CompositeBox.CompositeBox.DisplaySectionDeformation (
              self,
              TypeElem,
              NbElemX,
              NbElemY,
              NbElemPly,
              DefType,
              PlaneSection = False )
```

Definition at line 320 of file CompositeBox.py.

6.1.3.6 GetHeight()

```
def gebtaero.CompositeBox.CompositeBox.GetHeight (
              self )
```

Definition at line 34 of file CompositeBox.py.

6.1.3.7 GetOffsets()

```
def gebtaero.CompositeBox.CompositeBox.GetOffsets (
              self )
```

Definition at line 28 of file CompositeBox.py.

6.1.3.8 GetWidth()

```
def gebtaero.CompositeBox.CompositeBox.GetWidth (
              self )
```

Definition at line 31 of file CompositeBox.py.

6.1.4 Member Data Documentation

6.1.4.1 Down

```
gebtaero.CompositeBox.CompositeBox.Down
```

Definition at line 18 of file CompositeBox.py.

6.1.4.2 Height

```
gebtaero.CompositeBox.CompositeBox.Height
```

Definition at line 20 of file CompositeBox.py.

6.1.4.3 Left

```
gebtaero.CompositeBox.CompositeBox.Left
```

Definition at line 15 of file CompositeBox.py.

6.1.4.4 OffsetY

```
gebtaero.CompositeBox.CompositeBox.OffsetY
```

Definition at line 21 of file CompositeBox.py.

6.1.4.5 OffsetZ

```
gebtaero.CompositeBox.CompositeBox.OffsetZ
```

Definition at line 22 of file CompositeBox.py.

6.1.4.6 Right

```
gebtaero.CompositeBox.CompositeBox.Right
```

Definition at line 16 of file CompositeBox.py.

6.1.4.7 Up

```
gebtaero.CompositeBox.CompositeBox.Up
```

Definition at line 17 of file CompositeBox.py.

6.1.4.8 Width

```
gebtaero.CompositeBox.CompositeBox.Width
```

Definition at line 19 of file CompositeBox.py.

The documentation for this class was generated from the following file:

- /home/bertrand/these/logiciels/programme/interface/src/gebtaero/CompositeBox.py

6.2 gebtaero.CompositePlate.CompositePlate Class Reference

Public Member Functions

- def __init__ (self, Chord=1, OffsetY=0., OffsetZ=0.)
- def AppendPly (self, Ply)
- def GetLayup (self)
- def GetTotThickness (self)
- def GetOffsets (self)
- def CreateFbdFile (self, TypeElem, NbElemX, NbElemY, NbElemPly)
- def CreateInpFile (self, Stress=False, PlaneSection=False, Disp=0)
- def ComputeMassMatrix (self, OffsetY=0., OffsetZ=0.)
- def CreatePeriodicEq (self)
- def DisplaySectionDeformation (self, TypeElem, NbElemX, NbElemY, NbElemPly, DefType, Plane↩Section=False)

Public Attributes

- Chord
- Layup
- Materials
- Orientations
- TotThickness
- OffsetY
- OffsetZ

6.2.1 Detailed Description

```
Class interfacing the solver with 3D FEM calculix computation to obtain stiffness matrix
from a composite plate
```

Definition at line 8 of file CompositePlate.py.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 __init__()

```
def gebtaero.CompositePlate.CompositePlate.__init__ (
              self,
              Chord = 1,
              OffsetY = 0.,
              OffsetZ = 0.  )
```

Definition at line 13 of file CompositePlate.py.

6.2.3 Member Function Documentation

6.2.3.1 AppendPly()

```
def gebtaero.CompositePlate.CompositePlate.AppendPly (
              self,
              Ply )
```

Definition at line 22 of file CompositePlate.py.

6.2.3.2 ComputeMassMatrix()

```
def gebtaero.CompositePlate.CompositePlate.ComputeMassMatrix (
              self,
              OffsetY = 0.,
              OffsetZ = 0.  )
```

Definition at line 158 of file CompositePlate.py.

6.2.3.3 CreateFbdFile()

```
def gebtaero.CompositePlate.CompositePlate.CreateFbdFile (
              self,
              TypeElem,
              NbElemX,
              NbElemY,
              NbElemPly )
```

Definition at line 39 of file CompositePlate.py.

6.2.3.4 CreateInpFile()

```
def gebtaero.CompositePlate.CompositePlate.CreateInpFile (
              self,
              Stress = False,
              PlaneSection = False,
              Disp = 0 )
```

Definition at line 86 of file CompositePlate.py.

6.2.3.5 CreatePeriodicEq()

```
def gebtaero.CompositePlate.CompositePlate.CreatePeriodicEq (
              self )
```

Definition at line 185 of file CompositePlate.py.

6.2.3.6 DisplaySectionDeformation()

```
def gebtaero.CompositePlate.CompositePlate.DisplaySectionDeformation (
              self,
              TypeElem,
              NbElemX,
              NbElemY,
              NbElemPly,
              DefType,
              PlaneSection = False )
```

Definition at line 188 of file CompositePlate.py.

6.2.3.7 GetLayup()

```
def gebtaero.CompositePlate.CompositePlate.GetLayup (
              self )
```

Definition at line 30 of file CompositePlate.py.

6.2.3.8 GetOffsets()

```
def gebtaero.CompositePlate.CompositePlate.GetOffsets (
              self )
```

Definition at line 36 of file CompositePlate.py.

6.2.3.9 GetTotThickness()

```
def gebtaero.CompositePlate.CompositePlate.GetTotThickness (
              self )
```

Definition at line 33 of file CompositePlate.py.

6.2.4 Member Data Documentation

6.2.4.1 Chord

```
gebtaero.CompositePlate.CompositePlate.Chord
```

Definition at line 14 of file CompositePlate.py.

6.2.4.2 Layup

```
gebtaero.CompositePlate.CompositePlate.Layup
```

Definition at line 15 of file CompositePlate.py.

6.2.4.3 Materials

`gebtaero.CompositePlate.CompositePlate.Materials`

Definition at line 16 of file CompositePlate.py.

6.2.4.4 OffsetY

`gebtaero.CompositePlate.CompositePlate.OffsetY`

Definition at line 19 of file CompositePlate.py.

6.2.4.5 OffsetZ

`gebtaero.CompositePlate.CompositePlate.OffsetZ`

Definition at line 20 of file CompositePlate.py.

6.2.4.6 Orientations

`gebtaero.CompositePlate.CompositePlate.Orientations`

Definition at line 17 of file CompositePlate.py.

6.2.4.7 TotThickness

`gebtaero.CompositePlate.CompositePlate.TotThickness`

Definition at line 18 of file CompositePlate.py.

The documentation for this class was generated from the following file:

- /home/bertrand/these/logiciels/programme/interface/src/gebtaero/CompositePlate.py

6.3 gebtaero.CompositePly.CompositePly Class Reference

Public Member Functions

- def __init__ (self, Material, Thickness, Orientation)
- def GetMaterial (self)
- def GetThickness (self)
- def GetOrientation (self)

Public Attributes

- Material
- Thickness
- Orientation

6.3.1 Detailed Description

```
This class defined a laminated composite ply with an orthotropic material,
a thickness (m) and a fiber orientation (°)
```

Definition at line 1 of file CompositePly.py.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 __init__()

```
def gebtaero.CompositePly.CompositePly.__init__ (
              self,
              Material,
              Thickness,
              Orientation )
```

Definition at line 6 of file CompositePly.py.

6.3.3 Member Function Documentation

6.3.3.1 GetMaterial()

```
def gebtaero.CompositePly.CompositePly.GetMaterial (
              self )
```

Definition at line 11 of file CompositePly.py.

6.3.3.2 GetOrientation()

```
def gebtaero.CompositePly.CompositePly.GetOrientation (
              self )
```

Definition at line 17 of file CompositePly.py.

6.3.3.3 GetThickness()

```
def gebtaero.CompositePly.CompositePly.GetThickness (
              self )
```

Definition at line 14 of file CompositePly.py.

6.3.4 Member Data Documentation

6.3.4.1 Material

```
gebtaero.CompositePly.CompositePly.Material
```

Definition at line 7 of file CompositePly.py.

6.3.4.2 Orientation

```
gebtaero.CompositePly.CompositePly.Orientation
```

Definition at line 9 of file CompositePly.py.

6.3.4.3 Thickness

```
gebtaero.CompositePly.CompositePly.Thickness
```

Definition at line 8 of file CompositePly.py.

The documentation for this class was generated from the following file:

- /home/bertrand/these/logiciels/programme/interface/src/gebtaero/CompositePly.py

6.4 gebtaero.CrossSection.CrossSection Class Reference

Public Member Functions

- def __init__ (self)
- def SetMassMatrix (self, Mu, I22, I33, I23, Nu, Zcg=0.)
- def SetMassMatrixByPlate (self, Plate)
- def SetMassMatrixByBox (self, Box)
- def SetMassMatrixByRectBeamValues (self, h, L, Mat, NeglectI22=False)
- def SetMassMatrixByMesh (self, Mesh, AtElasticCenter=False, SymY=False, ChordScale=False, verbosity=0)
- def GetMassMatrix (self)
- def SetFlexibilityMatrixByRectBeamValues (self, h, L, Mat, RigidEIg3=False)
- def SetFlexibilityMatrixByIsotropicValues (self, EIg2, EIg3, GJ)
- def SetFlexibilityMatrixByPlate (self, Plate, TypeElem, NbElemX, NbElemY, NbElemPly, RigidX=False, RigidZ=False)
- def SetFlexibilityMatrixByBox (self, Box, TypeElem, NbElemX, NbElemY, NbElemPly, RigidX=False, RigidZ=False)
- def SetFlexibilityMatrixByMesh (self, Mesh, PlaneSection=False, AtElasticCenter=False, RigidX=False, RigidZ=False, ChordScale=False)
- def GetFlexibilityMatrix (self)

Public Attributes

- MassMatrix
- FlexibilityMatrix
- ElasticCenter

6.4.1 Detailed Description

```
class containing the flexibility and mass matrix of a wing section
Mass matrix is defined analytically, flexibility matrix is defined either analytically
or using 3D FEM solver Calculix
```

Definition at line 6 of file CrossSection.py.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 __init__()

```
def gebtaero.CrossSection.CrossSection.__init__ (
              self )
```

Definition at line 12 of file CrossSection.py.

6.4.3 Member Function Documentation

6.4.3.1 GetFlexibilityMatrix()

```
def gebtaero.CrossSection.CrossSection.GetFlexibilityMatrix (
              self )
```

Definition at line 218 of file CrossSection.py.

6.4.3.2 GetMassMatrix()

```
def gebtaero.CrossSection.CrossSection.GetMassMatrix (
              self )
```

Definition at line 105 of file CrossSection.py.

6.4.3.3 SetFlexibilityMatrixByBox()

```
def gebtaero.CrossSection.CrossSection.SetFlexibilityMatrixByBox (
              self,
              Box,
              TypeElem,
              NbElemX,
              NbElemY,
              NbElemPly,
              RigidX = False,
              RigidZ = False )
```

```
Set the flexibility matrix using homogeneisation process.
First a 4*4 Stiffness matrix is computed, then coefficients are
placed in the final 6*6 Flexibility Matrix and scaled using the box width
```

Definition at line 171 of file CrossSection.py.

6.4.3.4 SetFlexibilityMatrixByIsotropicValues()

```
def gebtaero.CrossSection.CrossSection.SetFlexibilityMatrixByIsotropicValues (
              self,
              EIg2,
              EIg3,
              GJ )
```

```
Set the flexibility matrix using isotropic vlaues namely the bending stiffness
spanwise EIg2, the bending stiffness chordwise EIg3 and the torsionnal stiffness
GJ.
@param float EIg2 : bending stiffness spanwise
@param float EIg3 : bending stiffness chordwise
@param float GJ : torsionnal stiffness
```

Definition at line 126 of file CrossSection.py.

6.4.3.5 SetFlexibilityMatrixByMesh()

```
def gebtaero.CrossSection.CrossSection.SetFlexibilityMatrixByMesh (
              self,
              Mesh,
              PlaneSection = False,
              AtElasticCenter = False,
              RigidX = False,
              RigidZ = False,
              ChordScale = False )
```

Definition at line 192 of file CrossSection.py.

6.4.3.6 SetFlexibilityMatrixByPlate()

```
def gebtaero.CrossSection.CrossSection.SetFlexibilityMatrixByPlate (
        self,
        Plate,
        TypeElem,
        NbElemX,
        NbElemY,
        NbElemPly,
        RigidX = False,
        RigidZ = False )
```

```
Set the flexibility matrix using homogeneisation process.
First a 4*4 Stiffness matrix is computed, then inverted and coefficients are
placed in the final 6*6 Flexibility Matrix
```

Definition at line 150 of file CrossSection.py.

6.4.3.7 SetFlexibilityMatrixByRectBeamValues()

```
def gebtaero.CrossSection.CrossSection.SetFlexibilityMatrixByRectBeamValues (
        self,
        h,
        L,
        Mat,
        RigidEIg3 = False )
```

Definition at line 108 of file CrossSection.py.

6.4.3.8 SetMassMatrix()

```
def gebtaero.CrossSection.CrossSection.SetMassMatrix (
        self,
        Mu,
        I22,
        I33,
        I23,
        Nu,
        Zcg = 0. )
```

```
@param float Mu : Mass per unit length
@param float I11 : Mass moment of inertia about wingspan axis
@param float Nu : Distance between moment calculation point and center of gravity. Positif is the CG is near l
```

Definition at line 17 of file CrossSection.py.

6.4.3.9 SetMassMatrixByBox()

```
def gebtaero.CrossSection.CrossSection.SetMassMatrixByBox (
              self,
              Box )
```

Definition at line 46 of file CrossSection.py.

6.4.3.10 SetMassMatrixByMesh()

```
def gebtaero.CrossSection.CrossSection.SetMassMatrixByMesh (
              self,
              Mesh,
              AtElasticCenter = False,
              SymY = False,
              ChordScale = False,
              verbosity = 0 )
```

Definition at line 66 of file CrossSection.py.

6.4.3.11 SetMassMatrixByPlate()

```
def gebtaero.CrossSection.CrossSection.SetMassMatrixByPlate (
              self,
              Plate )
```

Definition at line 41 of file CrossSection.py.

6.4.3.12 SetMassMatrixByRectBeamValues()

```
def gebtaero.CrossSection.CrossSection.SetMassMatrixByRectBeamValues (
              self,
              h,
              L,
              Mat,
              NeglectI22 = False )
```

Definition at line 51 of file CrossSection.py.

6.4.4 Member Data Documentation

6.4.4.1 ElasticCenter

`gebtaero.CrossSection.CrossSection.ElasticCenter`

Definition at line 15 of file CrossSection.py.

6.4.4.2 FlexibilityMatrix

`gebtaero.CrossSection.CrossSection.FlexibilityMatrix`

Definition at line 14 of file CrossSection.py.

6.4.4.3 MassMatrix

`gebtaero.CrossSection.CrossSection.MassMatrix`

Definition at line 13 of file CrossSection.py.

The documentation for this class was generated from the following file:

- /home/bertrand/these/logiciels/programme/interface/src/gebtaero/CrossSection.py

6.5 prescribedcondition::distriload Type Reference

Define the distributed load condition.

Private Attributes

- real(dbl), dimension(nstrn) value

  the current functional value of the load
- real(dbl), dimension(nstrn, nstrn) distr_fun

  the distribution function for each load
- integer, dimension(nstrn) follower

  whether the force vector/moment vector are follower quantities

6.5.1 Detailed Description

Define the distributed load condition.

Definition at line 41 of file PrescribedCondition.f90.

6.5.2 Member Data Documentation

6.5.2.1 distr_fun

`real(dbl), dimension(nstrn,nstrn) prescribedcondition::distriload::distr_fun [private]`

the distribution function for each load

Definition at line 44 of file PrescribedCondition.f90.

6.5.2.2 follower

`integer, dimension(nstrn) prescribedcondition::distriload::follower [private]`

whether the force vector/moment vector are follower quantities

Definition at line 45 of file PrescribedCondition.f90.

6.5.2.3 value

`real(dbl), dimension(nstrn) prescribedcondition::distriload::value [private]`

the current functional value of the load

Definition at line 43 of file PrescribedCondition.f90.

The documentation for this type was generated from the following file:

- /home/bertrand/these/logiciels/programme/src/PrescribedCondition.f90

6.6 gebtaero.ExternalMesh.ExternalMesh Class Reference

Public Member Functions

- def __init__ (self, MeshFile, OffsetY=0., OffsetZ=0., UnvConv=False, Chord=1.)
- def CreatePeriodicEq (self)
- def GetMeshFile (self)
- def AppendComponent (self, Name, Material, Orientation=None)
- def CreateInpFile (self, Stress=False, PlaneSection=False, Disp=0)
- def ComputeElementSurfAndCG (self, nodes, element, x0)
- def ComputeMassMatrixFromMesh (self, nodes, x0, elements)
- def DisplaySectionDeformation (self, DefType)

Public Attributes

- MeshFile
- Components
- Materials
- Orientations
- TotThickness
- OffsetY
- OffsetZ
- Chord
- nstrain
- ncurv
- Lx
- nodes
- x0
- elements

6.6.1 Detailed Description

```
This class allow to use a mesh created without cgx, the elset name must correspond to the component name.
The input mesh format has to be inp
```

Definition at line 10 of file ExternalMesh.py.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 __init__()

```
def gebtaero.ExternalMesh.ExternalMesh.__init__ (
              self,
              MeshFile,
              OffsetY = 0.,
              OffsetZ = 0.,
              UnvConv = False,
              Chord = 1.  )
```

Definition at line 15 of file ExternalMesh.py.

6.6.3 Member Function Documentation

6.6.3.1 AppendComponent()

```
def gebtaero.ExternalMesh.ExternalMesh.AppendComponent (
              self,
              Name,
              Material,
              Orientation = None )
```

Definition at line 46 of file ExternalMesh.py.

6.6.3.2 ComputeElementSurfAndCG()

```
def gebtaero.ExternalMesh.ExternalMesh.ComputeElementSurfAndCG (
              self,
              nodes,
              element,
              x0 )
```

Definition at line 126 of file ExternalMesh.py.

6.6.3.3 ComputeMassMatrixFromMesh()

```
def gebtaero.ExternalMesh.ExternalMesh.ComputeMassMatrixFromMesh (
              self,
              nodes,
              x0,
              elements )
```

Definition at line 160 of file ExternalMesh.py.

6.6.3.4 CreateInpFile()

```
def gebtaero.ExternalMesh.ExternalMesh.CreateInpFile (
              self,
              Stress = False,
              PlaneSection = False,
              Disp = 0 )
```

Definition at line 57 of file ExternalMesh.py.

6.6.3.5 CreatePeriodicEq()

```
def gebtaero.ExternalMesh.ExternalMesh.CreatePeriodicEq (
              self )
```

Definition at line 40 of file ExternalMesh.py.

6.6.3.6 DisplaySectionDeformation()

```
def gebtaero.ExternalMesh.ExternalMesh.DisplaySectionDeformation (
              self,
              DefType )
```

Definition at line 198 of file ExternalMesh.py.

6.6.3.7 GetMeshFile()

```
def gebtaero.ExternalMesh.ExternalMesh.GetMeshFile (
              self )
```

Definition at line 43 of file ExternalMesh.py.

6.6.4 Member Data Documentation

6.6.4.1 Chord

```
gebtaero.ExternalMesh.ExternalMesh.Chord
```

Definition at line 30 of file ExternalMesh.py.

6.6.4.2 Components

```
gebtaero.ExternalMesh.ExternalMesh.Components
```

Definition at line 24 of file ExternalMesh.py.

6.6.4.3 elements

`gebtaero.ExternalMesh.ExternalMesh.elements`

Definition at line 38 of file ExternalMesh.py.

6.6.4.4 Lx

`gebtaero.ExternalMesh.ExternalMesh.Lx`

Definition at line 35 of file ExternalMesh.py.

6.6.4.5 Materials

`gebtaero.ExternalMesh.ExternalMesh.Materials`

Definition at line 25 of file ExternalMesh.py.

6.6.4.6 MeshFile

`gebtaero.ExternalMesh.ExternalMesh.MeshFile`

Definition at line 21 of file ExternalMesh.py.

6.6.4.7 ncurv

`gebtaero.ExternalMesh.ExternalMesh.ncurv`

Definition at line 34 of file ExternalMesh.py.

6.6.4.8 nodes

`gebtaero.ExternalMesh.ExternalMesh.nodes`

Definition at line 36 of file ExternalMesh.py.

6.6.4.9 nstrain

`gebtaero.ExternalMesh.ExternalMesh.nstrain`

Definition at line 33 of file ExternalMesh.py.

6.6.4.10 OffsetY

`gebtaero.ExternalMesh.ExternalMesh.OffsetY`

Definition at line 28 of file ExternalMesh.py.

6.6.4.11 OffsetZ

`gebtaero.ExternalMesh.ExternalMesh.OffsetZ`

Definition at line 29 of file ExternalMesh.py.

6.6.4.12 Orientations

`gebtaero.ExternalMesh.ExternalMesh.Orientations`

Definition at line 26 of file ExternalMesh.py.

6.6.4.13 TotThickness

`gebtaero.ExternalMesh.ExternalMesh.TotThickness`

Definition at line 27 of file ExternalMesh.py.

6.6.4.14 x0

`gebtaero.ExternalMesh.ExternalMesh.x0`

Definition at line 37 of file ExternalMesh.py.

The documentation for this class was generated from the following file:

- /home/bertrand/these/logiciels/programme/interface/src/gebtaero/ExternalMesh.py

6.7 gebtaero.Frame.Frame Class Reference

Public Member Functions

- def __init__ (self, Axis, Twist)
- def GetFrameMatrix (self)
- def GetAxis (self)
- def GetTwist (self)

Public Attributes

- Axis
- Twist
- FrameMatrix

6.7.1 Detailed Description

```
class containing the matrix Cab of the solver ie the local frame of the undeformed wing
```

Definition at line 5 of file Frame.py.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 __init__()

```
def gebtaero.Frame.Frame.__init__ (
              self,
              Axis,
              Twist )
```

Definition at line 9 of file Frame.py.

6.7.3 Member Function Documentation

6.7.3.1 GetAxis()

```
def gebtaero.Frame.Frame.GetAxis (
              self )
```

Definition at line 59 of file Frame.py.

6.7.3.2 GetFrameMatrix()

```
def gebtaero.Frame.Frame.GetFrameMatrix (
              self )
```

Definition at line 56 of file Frame.py.

6.7.3.3 GetTwist()

```
def gebtaero.Frame.Frame.GetTwist (
              self )
```

Definition at line 62 of file Frame.py.

6.7.4 Member Data Documentation

6.7.4.1 Axis

```
gebtaero.Frame.Frame.Axis
```

Definition at line 10 of file Frame.py.

6.7.4.2 FrameMatrix

```
gebtaero.Frame.Frame.FrameMatrix
```

Definition at line 12 of file Frame.py.

6.7.4.3 Twist

```
gebtaero.Frame.Frame.Twist
```

Definition at line 11 of file Frame.py.

The documentation for this class was generated from the following file:

- /home/bertrand/these/logiciels/programme/interface/src/gebtaero/Frame.py

6.8 gebtaero.GebtPlot.GebtPlot Class Reference

Public Member Functions

- def EigenFreqDamping (Velocity, Modes, Style=None, ReducedDamping=True, DampAxis=False, Damp←Scale=0.01)
- def EigenFreqDampingUnsorted (Velocity, Modes, Style=None, DampAxis=False, DampScale=0.01)
- def WriteParaviewScript (VtkFolder, VtkName, AirfoilPath="/opt/gebtaero/airfoil/airfoil_default.vtk")
- def ParaviewOutput (self, VtkFolder, VtkName, AirfoilPath="/opt/gebtaero/airfoil/airfoil_default.vtk")

6.8.1 Detailed Description

```
Class containing ploting routines
```

Definition at line 7 of file GebtPlot.py.

6.8.2 Member Function Documentation

6.8.2.1 EigenFreqDamping()

```
def gebtaero.GebtPlot.GebtPlot.EigenFreqDamping (
              Velocity,
              Modes,
              Style = None,
              ReducedDamping = True,
              DampAxis = False,
              DampScale = 0.01 )
```

```
Plot frequencies and Damping of a set of computed modes
```

Definition at line 12 of file GebtPlot.py.

6.8.2.2 EigenFreqDampingUnsorted()

```
def gebtaero.GebtPlot.GebtPlot.EigenFreqDampingUnsorted (
              Velocity,
              Modes,
              Style = None,
              DampAxis = False,
              DampScale = 0.01 )
```

```
Plot frequencies and Damping of a set of computed modes
```

Definition at line 65 of file GebtPlot.py.

6.8.2.3 ParaviewOutput()

```
def gebtaero.GebtPlot.GebtPlot.ParaviewOutput (
              self,
              VtkFolder,
              VtkName,
              AirfoilPath = "/opt/gebtaero/airfoil/airfoil_default.vtk" )
```

Definition at line 125 of file GebtPlot.py.

6.8.2.4 WriteParaviewScript()

```
def gebtaero.GebtPlot.GebtPlot.WriteParaviewScript (
              VtkFolder,
              VtkName,
              AirfoilPath = "/opt/gebtaero/airfoil/airfoil_default.vtk" )
```

Definition at line 94 of file GebtPlot.py.

The documentation for this class was generated from the following file:

- /home/bertrand/these/logiciels/programme/interface/src/gebtaero/GebtPlot.py

6.9 gebtaero.InputFile.InputFile Class Reference

Public Member Functions

- def __init__ (self, Name, AnalysisFlag, AeroFlag, GravFlag, Niter, Nstep, Nvtk, Nev, ACOmegaa, A↩ COmegaaTFNumber, ACVa, ACVaTFNumber, Wing, Vinf, Rho, AlphaAC, BetaAC, SimuStart, SimuEnd, Xcg=0.)
- def GetName (self)
- def GetFileName (self)
- def GetAnalysisFlag (self)
- def AppendTimeFunction (self, TimeFunction)
- def GetTimeFunction (self, index)
- def WriteInputFile (self)
- def RemoveInputFile (self)
- def WriteInitFile (self)

Public Attributes

- Name
- FileName
- AnalysisFlag
- AeroFlag
- GravFlag
- Niter
- Nstep
- Nvtk
- Nev
- ACOmegaa
- ACOmegaaTFNumber
- ACVa
- ACVaTFNumber
- Wing
- Vinf
- Rho
- AlphaAC
- BetaAC
- SimuStart
- SimuEnd
- Xcg
- TimeFunctions

6.9.1 Detailed Description

```
This class is a mirror of the input data file of the solver
a method is implemented to write the input file using the class argument
```

Definition at line 4 of file InputFile.py.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 __init__()

```
def gebtaero.InputFile.InputFile.__init__ (
              self,
              Name,
              AnalysisFlag,
              AeroFlag,
              GravFlag,
              Niter,
              Nstep,
              Nvtk,
              Nev,
              ACOmegaa,
              ACOmegaaTFNumber,
              ACVa,
```

```
              ACVaTFNumber,
              Wing,
              Vinf,
              Rho,
              AlphaAC,
              BetaAC,
              SimuStart,
              SimuEnd,
              Xcg = 0.  )
```

Definition at line 9 of file InputFile.py.

6.9.3 Member Function Documentation

6.9.3.1 AppendTimeFunction()

```
def gebtaero.InputFile.InputFile.AppendTimeFunction (
              self,
              TimeFunction )
```

Definition at line 44 of file InputFile.py.

6.9.3.2 GetAnalysisFlag()

```
def gebtaero.InputFile.InputFile.GetAnalysisFlag (
              self )
```

Definition at line 41 of file InputFile.py.

6.9.3.3 GetFileName()

```
def gebtaero.InputFile.InputFile.GetFileName (
              self )
```

Definition at line 38 of file InputFile.py.

6.9.3.4 GetName()

```
def gebtaero.InputFile.InputFile.GetName (
              self )
```

Definition at line 35 of file InputFile.py.

6.9.3.5 GetTimeFunction()

```
def gebtaero.InputFile.InputFile.GetTimeFunction (
              self,
              index )
```

Definition at line 47 of file InputFile.py.

6.9.3.6 RemoveInputFile()

```
def gebtaero.InputFile.InputFile.RemoveInputFile (
              self )
```

Definition at line 207 of file InputFile.py.

6.9.3.7 WriteInitFile()

```
def gebtaero.InputFile.InputFile.WriteInitFile (
              self )
```

Definition at line 211 of file InputFile.py.

6.9.3.8 WriteInputFile()

```
def gebtaero.InputFile.InputFile.WriteInputFile (
              self )
```

```
This method intend to write the input data file of the solver using the class attributes
```

Definition at line 50 of file InputFile.py.

6.9.4 Member Data Documentation

6.9.4.1 ACOmegaa

```
gebtaero.InputFile.InputFile.ACOmegaa
```

Definition at line 19 of file InputFile.py.

6.9.4.2 ACOmegaaTFNumber

`gebtaero.InputFile.InputFile.ACOmegaaTFNumber`

Definition at line 20 of file InputFile.py.

6.9.4.3 ACVa

`gebtaero.InputFile.InputFile.ACVa`

Definition at line 21 of file InputFile.py.

6.9.4.4 ACVaTFNumber

`gebtaero.InputFile.InputFile.ACVaTFNumber`

Definition at line 22 of file InputFile.py.

6.9.4.5 AeroFlag

`gebtaero.InputFile.InputFile.AeroFlag`

Definition at line 13 of file InputFile.py.

6.9.4.6 AlphaAC

`gebtaero.InputFile.InputFile.AlphaAC`

Definition at line 26 of file InputFile.py.

6.9.4.7 AnalysisFlag

`gebtaero.InputFile.InputFile.AnalysisFlag`

Definition at line 12 of file InputFile.py.

6.9.4.8 BetaAC

`gebtaero.InputFile.InputFile.BetaAC`

Definition at line 27 of file InputFile.py.

6.9.4.9 FileName

`gebtaero.InputFile.InputFile.FileName`

Definition at line 11 of file InputFile.py.

6.9.4.10 GravFlag

`gebtaero.InputFile.InputFile.GravFlag`

Definition at line 14 of file InputFile.py.

6.9.4.11 Name

`gebtaero.InputFile.InputFile.Name`

Definition at line 10 of file InputFile.py.

6.9.4.12 Nev

`gebtaero.InputFile.InputFile.Nev`

Definition at line 18 of file InputFile.py.

6.9.4.13 Niter

`gebtaero.InputFile.InputFile.Niter`

Definition at line 15 of file InputFile.py.

6.9.4.14 Nstep

`gebtaero.InputFile.InputFile.Nstep`

Definition at line 16 of file InputFile.py.

6.9.4.15 Nvtk

`gebtaero.InputFile.InputFile.Nvtk`

Definition at line 17 of file InputFile.py.

6.9.4.16 Rho

`gebtaero.InputFile.InputFile.Rho`

Definition at line 25 of file InputFile.py.

6.9.4.17 SimuEnd

`gebtaero.InputFile.InputFile.SimuEnd`

Definition at line 29 of file InputFile.py.

6.9.4.18 SimuStart

`gebtaero.InputFile.InputFile.SimuStart`

Definition at line 28 of file InputFile.py.

6.9.4.19 TimeFunctions

`gebtaero.InputFile.InputFile.TimeFunctions`

Definition at line 33 of file InputFile.py.

6.9.4.20 Vinf

```
gebtaero.InputFile.InputFile.Vinf
```

Definition at line 24 of file InputFile.py.

6.9.4.21 Wing

```
gebtaero.InputFile.InputFile.Wing
```

Definition at line 23 of file InputFile.py.

6.9.4.22 Xcg

```
gebtaero.InputFile.InputFile.Xcg
```

Definition at line 30 of file InputFile.py.

The documentation for this class was generated from the following file:

- /home/bertrand/these/logiciels/programme/interface/src/gebtaero/InputFile.py

6.10 gebtaero.IsoMaterial.IsoMaterial Class Reference

Public Member Functions

- def __init__ (self, E, Nu, Rho=None)
- def GetIso (self)
- def GetDensity (self)

Public Attributes

- E
- Nu
- Rho

6.10.1 Detailed Description

```
This class contains the isotropic material characteristics
```

Definition at line 1 of file IsoMaterial.py.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 __init__()

```
def gebtaero.IsoMaterial.IsoMaterial.__init__ (
              self,
              E,
              Nu,
              Rho = None )
```

Definition at line 5 of file IsoMaterial.py.

6.10.3 Member Function Documentation

6.10.3.1 GetDensity()

```
def gebtaero.IsoMaterial.IsoMaterial.GetDensity (
              self )
```

Definition at line 13 of file IsoMaterial.py.

6.10.3.2 GetIso()

```
def gebtaero.IsoMaterial.IsoMaterial.GetIso (
              self )
```

Definition at line 10 of file IsoMaterial.py.

6.10.4 Member Data Documentation

6.10.4.1 E

```
gebtaero.IsoMaterial.IsoMaterial.E
```

Definition at line 6 of file IsoMaterial.py.

6.10.4.2 Nu

```
gebtaero.IsoMaterial.IsoMaterial.Nu
```

Definition at line 7 of file IsoMaterial.py.

6.10.4.3 Rho

```
gebtaero.IsoMaterial.IsoMaterial.Rho
```

Definition at line 8 of file IsoMaterial.py.

The documentation for this class was generated from the following file:

- /home/bertrand/these/logiciels/programme/interface/src/gebtaero/IsoMaterial.py

6.11 internaldata::memberinf Type Reference

structure containing the caracteritics of a finite element.

Public Attributes

- integer ndiv

  number of divisions
- integer ncol_memb

  total number of columns of the member
- real(dbl) dl

  length of each division
- real(dbl), dimension(:,:,:), allocatable mate

  mate(ndiv,12,6): flexibity and mass properties for each division
- real(dbl), dimension(:,:,:), allocatable triad

  triad(ndiv,3,3): cab for each division, evaluated at the middle point of the division
- real(dbl), dimension(:), allocatable le

  Le(ndiv): ending arc length for each division.
- real(dbl), dimension(:,:), allocatable coordinate

  coordinate(ndiv,3) coordinate for the middle of the division
- real(dbl), dimension(:,:), allocatable aerodyn_coef

  aerodynamic coefficients

6.11.1 Detailed Description

structure containing the caracteritics of a finite element.

Definition at line 52 of file InternalData.f90.

6.11.2 Member Data Documentation

6.11.2.1 aerodyn_coef

```
real(dbl), dimension(:,:), allocatable internaldata::memberinf::aerodyn_coef
```

aerodynamic coefficients

Definition at line 60 of file InternalData.f90.

6.11.2.2 coordinate

```
real(dbl), dimension(:,:), allocatable internaldata::memberinf::coordinate
```

coordinate(ndiv,3) coordinate for the middle of the division

Definition at line 59 of file InternalData.f90.

6.11.2.3 dl

```
real(dbl) internaldata::memberinf::dl
```

length of each division

Definition at line 55 of file InternalData.f90.

6.11.2.4 le

```
real(dbl), dimension(:), allocatable internaldata::memberinf::le
```

Le(ndiv): ending arc length for each division.

Definition at line 58 of file InternalData.f90.

6.11.2.5 mate

```
real(dbl), dimension(:,:,:), allocatable internaldata::memberinf::mate
```

mate(ndiv,12,6): flexibity and mass properties for each division

Definition at line 56 of file InternalData.f90.

6.11.2.6 ncol_memb

```
integer internaldata::memberinf::ncol_memb
```

total number of columns of the member

Definition at line 54 of file InternalData.f90.

6.11.2.7 ndiv

```
integer internaldata::memberinf::ndiv
```

number of divisions

Definition at line 53 of file InternalData.f90.

6.11.2.8 triad

```
real(dbl), dimension(:,:,:), allocatable internaldata::memberinf::triad
```

triad(ndiv,3,3): cab for each division, evaluated at the middle point of the division

Definition at line 57 of file InternalData.f90.

The documentation for this type was generated from the following file:

- /home/bertrand/these/logiciels/programme/src/InternalData.f90

6.12 gebtaero.OrthoMaterial.OrthoMaterial Class Reference

Public Member Functions

- def __init__ (self, El, Et, Nult, Glt, Rho=None)
- def GetOrtho (self)
- def GetDensity (self)

Public Attributes

- El
- Et
- Nult
- Glt
- Rho

6.12.1 Detailed Description

```
This class contains the orthotropic material characteristics
```

Definition at line 1 of file OrthoMaterial.py.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 __init__()

```
def gebtaero.OrthoMaterial.OrthoMaterial.__init__ (
              self,
              El,
              Et,
              Nult,
              Glt,
              Rho = None )
```

Definition at line 5 of file OrthoMaterial.py.

6.12.3 Member Function Documentation

6.12.3.1 GetDensity()

```
def gebtaero.OrthoMaterial.OrthoMaterial.GetDensity (
              self )
```

Definition at line 15 of file OrthoMaterial.py.

6.12.3.2 GetOrtho()

```
def gebtaero.OrthoMaterial.OrthoMaterial.GetOrtho (
              self )
```

Definition at line 12 of file OrthoMaterial.py.

6.12.4 Member Data Documentation

6.12.4.1 El

`gebtaero.OrthoMaterial.OrthoMaterial.El`

Definition at line 6 of file OrthoMaterial.py.

6.12.4.2 Et

`gebtaero.OrthoMaterial.OrthoMaterial.Et`

Definition at line 7 of file OrthoMaterial.py.

6.12.4.3 Glt

`gebtaero.OrthoMaterial.OrthoMaterial.Glt`

Definition at line 9 of file OrthoMaterial.py.

6.12.4.4 Nult

`gebtaero.OrthoMaterial.OrthoMaterial.Nult`

Definition at line 8 of file OrthoMaterial.py.

6.12.4.5 Rho

`gebtaero.OrthoMaterial.OrthoMaterial.Rho`

Definition at line 10 of file OrthoMaterial.py.

The documentation for this class was generated from the following file:

- /home/bertrand/these/logiciels/programme/interface/src/gebtaero/OrthoMaterial.py

6.13 prescribedcondition::prescriinf Type Reference

Define the prescribed condition.

Private Attributes

- integer id

  where it is applied, could be a node number or member number
- integer, dimension(nstrn) dof

  maximum 6 degrees of freedom can be prescribed, for distributed loads, it is used to denote the distribution function no
- real(dbl), dimension(nstrn) value

  the magnitude of the prescribed values
- integer, dimension(nstrn) time_fun_no

  which time function is used
- integer, dimension(nstrn) follower

  whether the prescribed quantity is a follower or not: 1 is a follower; 0 is not
- real(dbl), dimension(nstrn) value_current

  indicate the current functional value updated by time steps, calculated internally

6.13.1 Detailed Description

Define the prescribed condition.

Definition at line 28 of file PrescribedCondition.f90.

6.13.2 Member Data Documentation

6.13.2.1 dof

`integer, dimension(nstrn) prescribedcondition::prescriinf::dof [private]`

maximum 6 degrees of freedom can be prescribed, for distributed loads, it is used to denote the distribution function no

Definition at line 31 of file PrescribedCondition.f90.

6.13.2.2 follower

`integer, dimension(nstrn) prescribedcondition::prescriinf::follower [private]`

whether the prescribed quantity is a follower or not: 1 is a follower; 0 is not

Definition at line 35 of file PrescribedCondition.f90.

6.13.2.3 id

`integer prescribedcondition::prescriinf::id [private]`

where it is applied, could be a node number or member number

Definition at line 30 of file PrescribedCondition.f90.

6.13.2.4 time_fun_no

`integer, dimension(nstrn) prescribedcondition::prescriinf::time_fun_no [private]`

which time function is used

Definition at line 34 of file PrescribedCondition.f90.

6.13.2.5 value

`real(dbl), dimension(nstrn) prescribedcondition::prescriinf::value [private]`

the magnitude of the prescribed values

Definition at line 33 of file PrescribedCondition.f90.

6.13.2.6 value_current

`real(dbl), dimension(nstrn) prescribedcondition::prescriinf::value_current [private]`

indicate the current functional value updated by time steps, calculated internally

Definition at line 36 of file PrescribedCondition.f90.

The documentation for this type was generated from the following file:

- /home/bertrand/these/logiciels/programme/src/PrescribedCondition.f90

6.14 gebtaero.Simulation.Simulation Class Reference

Public Member Functions

- def __init__ (self, Wing)
- def GetWing (self)
- def Eigenvalues (self, Vinf, Rho, AlphaAC, BetaAC, AeroFlag, NumberOfModes, GravFlag=0, verbosity=0, arpack=0, vtk=0)
- def StaticLoads (self, Vinf, Rho, AlphaAC, BetaAC, GravFlag=0, verbosity=0)
- def ModalFlutterSpeed (self, Rho, Vmin, Vmax, Vstep, DeltaV, AeroFlag, FreqLim, AlphaAC, BetaAC, Ksi↩ Obj=1e-6, GravFlag=0, verbosity=0, arpack=1, ModesToCompute=20)
- def ModalCriticalSpeed (self, Rho, Vmin, Vmax, Vstep, DeltaV, AeroFlag, AlphaAC, BetaAC, GravFlag=0, verbosity=0, mode=0)
- def ModalDivergenceSpeed (self, Rho, Vmin, Vmax, Vstep, DeltaV, AeroFlag, AlphaAC, BetaAC, Norm↩ Lim=1., KsiObj=1e-6, GravFlag=0, verbosity=0)
- def ModalFlutterSpeedSorted (self, Rho, Vmax, DeltaV, AeroFlag, ModesToCompute, ModesToPlot, Alpha↩ AC, BetaAC, KsiObj=1e-6, CorrCoef=0.9, GravFlag=0, verbosity=0, arpack=0)
- def ModalDivergenceSpeedSorted (self, Rho, Vmax, DeltaV, AeroFlag, ModesToCompute, ModesToPlot, AlphaAC, BetaAC, CorrCoef=0.95, GravFlag=0, verbosity=0)
- def EigenTabSorted (self, Rho, Vmax, DeltaV, Nstep, AeroFlag, ModesToCompute, ModesToPlot, AlphaAC, BetaAC, CorrCoef=0.9, GravFlag=0, verbosity=0)
- def EigenTab (self, Rho, Vmin, Vmax, Nstep, AeroFlag, ModesToPlot, ModesToCompute, AlphaAC, BetaAC, GravFlag=0, verbosity=0)
- def DeformedModalFlutterSpeed (self, Rho, Vmin, Vmax, DeltaV, AeroFlag, FreqLim, NumberOfModes, Ksitol, Lifttol, BetaAC, verbosity=0)
- def DeformedModalFlutterSpeedSorted (self, Rho, Vmax, DeltaV, AeroFlag, ModesToCompute, ModesTo↩ Plot, Lifttol, BetaAC, KsiObj=1e-6, verbosity=0)
- def EquilibriumAoA (self, Rho, Vinf, BetaAC, tolerance, verbosity=0)
- def TemporalFlutterSpeed (self, Rho, Vmin, Vmax, DeltaV, AeroFlag, ModesToPlot, AlphaAC, BetaAC, Ksitol, NbPeriod, StepByPeriod, CoefPerturb, GravFlag=0, verbosity=0)
- def FlutterVtk (self, Rho, Vmin, Vmax, DeltaV, AeroFlag, AlphaAC, BetaAC, NbPeriod, StepByPeriod, Coef↩ Perturb, CoefVinf, Nvtk, FlutterLimit=0.4, GravFlag=0, verbosity=0)
- def TemporalDynamic (self, Rho, Vinf, AeroFlag, AlphaAC, BetaAC, Time, Nstep, FreqPerturb, TimePerturb, CoefPerturb, CoefVinf, Nvtk, GravFlag=0, verbosity=0)
- def TemporalDivergenceSpeed (self, Rho, Vmin, Vmax, Vstep, DeltaV, AeroFlag, BetaAC, GravFlag=0, verbosity=0)

Public Attributes

- Wing
- Input

6.14.1 Detailed Description

```
This class contains the methods design to find eigenvalues, static shape, flutter speed
(modal or temporal), divergence speed of a Wing in a particular configuration (AoA, Vinf,...)
```

Definition at line 12 of file Simulation.py.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 __init__()

```
def gebtaero.Simulation.Simulation.__init__ (
              self,
              Wing )
```

Definition at line 17 of file Simulation.py.

6.14.3 Member Function Documentation

6.14.3.1 DeformedModalFlutterSpeed()

```
def gebtaero.Simulation.Simulation.DeformedModalFlutterSpeed (
              self,
              Rho,
              Vmin,
              Vmax,
              DeltaV,
              AeroFlag,
              FreqLim,
              NumberOfModes,
              Ksitol,
              Lifttol,
              BetaAC,
              verbosity = 0 )
```

Definition at line 717 of file Simulation.py.

6.14.3.2 DeformedModalFlutterSpeedSorted()

```
def gebtaero.Simulation.Simulation.DeformedModalFlutterSpeedSorted (
              self,
              Rho,
              Vmax,
              DeltaV,
              AeroFlag,
              ModesToCompute,
              ModesToPlot,
              Lifttol,
              BetaAC,
              KsiObj = 1e-6,
              verbosity = 0 )
```

Definition at line 740 of file Simulation.py.

6.14.3.3 EigenTab()

```
def gebtaero.Simulation.Simulation.EigenTab (
              self,
              Rho,
              Vmin,
              Vmax,
              Nstep,
              AeroFlag,
              ModesToPlot,
              ModesToCompute,
              AlphaAC,
              BetaAC,
              GravFlag = 0,
              verbosity = 0 )
```

Definition at line 673 of file Simulation.py.

6.14.3.4 EigenTabSorted()

```
def gebtaero.Simulation.Simulation.EigenTabSorted (
              self,
              Rho,
              Vmax,
              DeltaV,
              Nstep,
              AeroFlag,
              ModesToCompute,
              ModesToPlot,
              AlphaAC,
              BetaAC,
              CorrCoef = 0.9,
              GravFlag = 0,
              verbosity = 0 )
```

Definition at line 542 of file Simulation.py.

6.14.3.5 Eigenvalues()

```
def gebtaero.Simulation.Simulation.Eigenvalues (
              self,
              Vinf,
              Rho,
              AlphaAC,
              BetaAC,
              AeroFlag,
              NumberOfModes,
              GravFlag = 0,
              verbosity = 0,
              arpack = 0,
              vtk = 0 )
```

Definition at line 23 of file Simulation.py.

6.14.3.6 EquilibriumAoA()

```
def gebtaero.Simulation.Simulation.EquilibriumAoA (
              self,
              Rho,
              Vinf,
              BetaAC,
              tolerance,
              verbosity = 0 )
```

Definition at line 763 of file Simulation.py.

6.14.3.7 FlutterVtk()

```
def gebtaero.Simulation.Simulation.FlutterVtk (
              self,
              Rho,
              Vmin,
              Vmax,
              DeltaV,
              AeroFlag,
              AlphaAC,
              BetaAC,
              NbPeriod,
              StepByPeriod,
              CoefPerturb,
              CoefVinf,
              Nvtk,
              FlutterLimit = 0.4,
              GravFlag = 0,
              verbosity = 0 )
```

Definition at line 837 of file Simulation.py.

6.14.3.8 GetWing()

```
def gebtaero.Simulation.Simulation.GetWing (
              self )
```

Definition at line 20 of file Simulation.py.

6.14.3.9 ModalCriticalSpeed()

```
def gebtaero.Simulation.Simulation.ModalCriticalSpeed (
              self,
              Rho,
              Vmin,
              Vmax,
              Vstep,
              DeltaV,
              AeroFlag,
              AlphaAC,
              BetaAC,
              GravFlag = 0,
              verbosity = 0,
              mode = 0 )
```

Definition at line 124 of file Simulation.py.

6.14.3.10 ModalDivergenceSpeed()

```
def gebtaero.Simulation.Simulation.ModalDivergenceSpeed (
              self,
              Rho,
              Vmin,
              Vmax,
              Vstep,
              DeltaV,
              AeroFlag,
              AlphaAC,
              BetaAC,
              NormLim = 1.,
              KsiObj = 1e-6,
              GravFlag = 0,
              verbosity = 0 )
```

Definition at line 240 of file Simulation.py.

6.14.3.11 ModalDivergenceSpeedSorted()

```
def gebtaero.Simulation.Simulation.ModalDivergenceSpeedSorted (
              self,
              Rho,
              Vmax,
              DeltaV,
              AeroFlag,
              ModesToCompute,
              ModesToPlot,
              AlphaAC,
              BetaAC,
              CorrCoef = 0.95,
              GravFlag = 0,
              verbosity = 0 )
```

Definition at line 427 of file Simulation.py.

6.14.3.12 ModalFlutterSpeed()

```
def gebtaero.Simulation.Simulation.ModalFlutterSpeed (
              self,
              Rho,
              Vmin,
              Vmax,
              Vstep,
              DeltaV,
              AeroFlag,
              FreqLim,
              AlphaAC,
              BetaAC,
              KsiObj = 1e-6,
              GravFlag = 0,
              verbosity = 0,
              arpack = 1,
              ModesToCompute = 20 )
```

Definition at line 56 of file Simulation.py.

6.14.3.13 ModalFlutterSpeedSorted()

```
def gebtaero.Simulation.Simulation.ModalFlutterSpeedSorted (
              self,
              Rho,
              Vmax,
              DeltaV,
              AeroFlag,
              ModesToCompute,
              ModesToPlot,
              AlphaAC,
              BetaAC,
              KsiObj = 1e-6,
              CorrCoef = 0.9,
              GravFlag = 0,
              verbosity = 0,
              arpack = 0 )
```

Definition at line 299 of file Simulation.py.

6.14.3.14 StaticLoads()

```
def gebtaero.Simulation.Simulation.StaticLoads (
              self,
              Vinf,
              Rho,
              AlphaAC,
              BetaAC,
              GravFlag = 0,
              verbosity = 0 )
```

Definition at line 40 of file Simulation.py.

6.14.3.15 TemporalDivergenceSpeed()

```
def gebtaero.Simulation.Simulation.TemporalDivergenceSpeed (
              self,
              Rho,
              Vmin,
              Vmax,
              Vstep,
              DeltaV,
              AeroFlag,
              BetaAC,
              GravFlag = 0,
              verbosity = 0 )
```

Definition at line 928 of file Simulation.py.

6.14.3.16 TemporalDynamic()

```
def gebtaero.Simulation.Simulation.TemporalDynamic (
              self,
              Rho,
              Vinf,
              AeroFlag,
              AlphaAC,
              BetaAC,
              Time,
              Nstep,
              FreqPerturb,
              TimePerturb,
              CoefPerturb,
              CoefVinf,
              Nvtk,
              GravFlag = 0,
              verbosity = 0 )
```

Definition at line 896 of file Simulation.py.

6.14.3.17 TemporalFlutterSpeed()

```
def gebtaero.Simulation.Simulation.TemporalFlutterSpeed (
              self,
              Rho,
              Vmin,
              Vmax,
              DeltaV,
              AeroFlag,
              ModesToPlot,
              AlphaAC,
              BetaAC,
              Ksitol,
```

```
            NbPeriod,
            StepByPeriod,
            CoefPerturb,
            GravFlag = 0,
            verbosity = 0 )
```

Definition at line 781 of file Simulation.py.

6.14.4 Member Data Documentation

6.14.4.1 Input

```
gebtaero.Simulation.Simulation.Input
```

Definition at line 26 of file Simulation.py.

6.14.4.2 Wing

```
gebtaero.Simulation.Simulation.Wing
```

Definition at line 18 of file Simulation.py.

The documentation for this class was generated from the following file:

- /home/bertrand/these/logiciels/programme/interface/src/gebtaero/Simulation.py

6.15 timefunctionmodule::timefunction Type Reference

Private Attributes

- integer fun_type
  function type: 0, user defined; 1, harmonic
- real(dbl) ts
- real(dbl) te
  starting and ending time
- integer entries
  number of entries
- real(dbl), dimension(:), pointer time_val
  the ith time, in increasing order; the amplitude of the harmonic
- real(dbl), dimension(:), pointer fun_val
  the ith functional value; the period of the harmonic
- real(dbl), dimension(:), pointer phase_val
  the phase value of the harmonic

6.15.1 Detailed Description

Definition at line 23 of file TimeFunction.f90.

6.15.2 Member Data Documentation

6.15.2.1 entries

`integer timefunctionmodule::timefunction::entries [private]`

number of entries

Definition at line 27 of file TimeFunction.f90.

6.15.2.2 fun_type

`integer timefunctionmodule::timefunction::fun_type [private]`

function type: 0, user defined; 1, harmonic

Definition at line 25 of file TimeFunction.f90.

6.15.2.3 fun_val

`real(dbl), dimension(:), pointer timefunctionmodule::timefunction::fun_val [private]`

the ith functional value; the period of the harmonic

Definition at line 29 of file TimeFunction.f90.

6.15.2.4 phase_val

`real(dbl), dimension(:), pointer timefunctionmodule::timefunction::phase_val [private]`

the phase value of the harmonic

Definition at line 30 of file TimeFunction.f90.

6.15.2.5 te

```
real(dbl) timefunctionmodule::timefunction::te  [private]
```

starting and ending time

Definition at line 26 of file TimeFunction.f90.

6.15.2.6 time_val

```
real(dbl), dimension(:), pointer timefunctionmodule::timefunction::time_val  [private]
```

the ith time, in increasing order; the amplitude of the harmonic

Definition at line 28 of file TimeFunction.f90.

6.15.2.7 ts

```
real(dbl) timefunctionmodule::timefunction::ts  [private]
```

Definition at line 26 of file TimeFunction.f90.

The documentation for this type was generated from the following file:

- /home/bertrand/these/logiciels/programme/src/TimeFunction.f90

6.16 gebtaero.TimeFunction.TimeFunction Class Reference

Public Member Functions

- def __init__ (self, FunctionType, FunctionStart, FunctionEnd)
- def GetFunctionType (self)
- def GetFunctionStart (self)
- def GetFunctionEnd (self)
- def AppendFunctionEntriePieceWise (self, time, value)
- def AppendFunctionEntrieHarmonic (self, amplitude, period, phase)
- def GetFunctionEntrie (self, index)
- def GetFunctionEntries (self)

Public Attributes

- FunctionType
- FunctionStart
- FunctionEnd
- FunctionEntries

6.16.1 Detailed Description

```
Time function object as defined in gebt documentation
```

Definition at line 3 of file TimeFunction.py.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 __init__()

```
def gebtaero.TimeFunction.TimeFunction.__init__ (
              self,
              FunctionType,
              FunctionStart,
              FunctionEnd )
```

Definition at line 7 of file TimeFunction.py.

6.16.3 Member Function Documentation

6.16.3.1 AppendFunctionEntrieHarmonic()

```
def gebtaero.TimeFunction.TimeFunction.AppendFunctionEntrieHarmonic (
              self,
              amplitude,
              period,
              phase )
```

Definition at line 30 of file TimeFunction.py.

6.16.3.2 AppendFunctionEntriePieceWise()

```
def gebtaero.TimeFunction.TimeFunction.AppendFunctionEntriePieceWise (
              self,
              time,
              value )
```

Definition at line 24 of file TimeFunction.py.

6.16.3.3 GetFunctionEnd()

```
def gebtaero.TimeFunction.TimeFunction.GetFunctionEnd (
              self )
```

Definition at line 21 of file TimeFunction.py.

6.16.3.4 GetFunctionEntrie()

```
def gebtaero.TimeFunction.TimeFunction.GetFunctionEntrie (
              self,
              index )
```

Definition at line 36 of file TimeFunction.py.

6.16.3.5 GetFunctionEntries()

```
def gebtaero.TimeFunction.TimeFunction.GetFunctionEntries (
              self )
```

Definition at line 39 of file TimeFunction.py.

6.16.3.6 GetFunctionStart()

```
def gebtaero.TimeFunction.TimeFunction.GetFunctionStart (
              self )
```

Definition at line 18 of file TimeFunction.py.

6.16.3.7 GetFunctionType()

```
def gebtaero.TimeFunction.TimeFunction.GetFunctionType (
              self )
```

Definition at line 15 of file TimeFunction.py.

6.16.4 Member Data Documentation

6.16.4.1 FunctionEnd

`gebtaero.TimeFunction.TimeFunction.FunctionEnd`

Definition at line 10 of file TimeFunction.py.

6.16.4.2 FunctionEntries

`gebtaero.TimeFunction.TimeFunction.FunctionEntries`

Definition at line 13 of file TimeFunction.py.

6.16.4.3 FunctionStart

`gebtaero.TimeFunction.TimeFunction.FunctionStart`

Definition at line 9 of file TimeFunction.py.

6.16.4.4 FunctionType

`gebtaero.TimeFunction.TimeFunction.FunctionType`

Definition at line 8 of file TimeFunction.py.

The documentation for this class was generated from the following file:

- /home/bertrand/these/logiciels/programme/interface/src/gebtaero/TimeFunction.py

6.17 gebtaero.Wing.Wing Class Reference

Public Member Functions

- def __init__ (self, Name, WingRootPosition)
- def GetWingRootPosition (self)
- def GetName (self)
- def AppendWingSection (self, WingSection)
- def GetWingSections (self)
- def GetCrossSections (self)
- def GetFrames (self)
- def GetKpList (self)
- def GetWeight (self)
- def GetSurface (self)

Public Attributes

- Name
- WingRootPosition
- WingSections
- Frames
- CrossSections
- KpList

6.17.1 Detailed Description

```
Wing compose by one ore more wing section
:version: 12/02/18
:author: Bertrand Kirsch

ATTRIBUTES

A list containing the wing sections
WingSections (private) : WingSection[]

A vector containing the wing root position
WingRootPosition (private) : np.array[3]

A vector containing the direction of the wing at the root
WingRootAxis (private) : np.array[3]

The wing twist at the root (rad)
WingRootTwist (private) : float

The wing frame to be input in the solver regarding the wing axis and the wing twist
WingFrame (private) : np.array[3][3]
```

Definition at line 4 of file Wing.py.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 __init__()

```
def gebtaero.Wing.Wing.__init__ (
              self,
              Name,
              WingRootPosition )
```

Definition at line 27 of file Wing.py.

6.17.3 Member Function Documentation

6.17.3.1 AppendWingSection()

```
def gebtaero.Wing.Wing.AppendWingSection (
              self,
              WingSection )
```

```
Add a section to the current wing, first section is at the wing root, last section at the wing tip.
```

Definition at line 44 of file Wing.py.

6.17.3.2 GetCrossSections()

```
def gebtaero.Wing.Wing.GetCrossSections (
              self )
```

Definition at line 63 of file Wing.py.

6.17.3.3 GetFrames()

```
def gebtaero.Wing.Wing.GetFrames (
              self )
```

Definition at line 66 of file Wing.py.

6.17.3.4 GetKpList()

```
def gebtaero.Wing.Wing.GetKpList (
              self )
```

Definition at line 69 of file Wing.py.

6.17.3.5 GetName()

```
def gebtaero.Wing.Wing.GetName (
              self )
```

Definition at line 41 of file Wing.py.

6.17.3.6 GetSurface()

```
def gebtaero.Wing.Wing.GetSurface (
              self )
```

Definition at line 80 of file Wing.py.

6.17.3.7 GetWeight()

```
def gebtaero.Wing.Wing.GetWeight (
              self )
```

Definition at line 72 of file Wing.py.

6.17.3.8 GetWingRootPosition()

```
def gebtaero.Wing.Wing.GetWingRootPosition (
              self )
```

Definition at line 38 of file Wing.py.

6.17.3.9 GetWingSections()

```
def gebtaero.Wing.Wing.GetWingSections (
              self )
```

Definition at line 60 of file Wing.py.

6.17.4 Member Data Documentation

6.17.4.1 CrossSections

```
gebtaero.Wing.Wing.CrossSections
```

Definition at line 34 of file Wing.py.

6.17.4.2 Frames

`gebtaero.Wing.Wing.Frames`

Definition at line 33 of file Wing.py.

6.17.4.3 KpList

`gebtaero.Wing.Wing.KpList`

Definition at line 35 of file Wing.py.

6.17.4.4 Name

`gebtaero.Wing.Wing.Name`

Definition at line 28 of file Wing.py.

6.17.4.5 WingRootPosition

`gebtaero.Wing.Wing.WingRootPosition`

Definition at line 29 of file Wing.py.

6.17.4.6 WingSections

`gebtaero.Wing.Wing.WingSections`

Definition at line 32 of file Wing.py.

The documentation for this class was generated from the following file:

- /home/bertrand/these/logiciels/programme/interface/src/gebtaero/Wing.py

6.18 gebtaero.WingSection.WingSection Class Reference

Public Member Functions

- def __init__ (self, Chord, ParameterA, NumberOfElements, SectionLength, CrossSection, Frame)
- def SetParameterA (self, ParameterA)
- def GetParameterA (self)
- def SetChord (self, Chord)
- def GetChord (self)
- def GetHalfChord (self)
- def SetNumberOfElements (self, NumberOfElements)
- def GetNumberOfElements (self)
- def SetSectionLength (self, SectionLength)
- def GetSectionLength (self)
- def GetCrossSection (self)
- def GetFrame (self)

Public Attributes

- Chord
- ParameterA
- NumberOfElements
- SectionLength
- HalfChord
- CrossSection
- Frame

6.18.1 Detailed Description

```
Section of a wing. A section is defined for each changing parameter (dihedral,
flexibility,...)

:version: 12/02/18
:author: Bertrand Kirsch

ATTRIBUTES

The loccal chord of the wing section (m)
Chord  (private) : float

Distance between the moment calculation point and the half chord. For isotropic
wing the moment calculation point is placed on the elastic center
ParameterA  (private) : float

Half of the local wing chord used in the aerodynamic model
HalfChord  (private) : float

Number of FE elements of the wing section
NumberOfElements  (private) : int

Length of the wing section
SectionLength (private) : float
```

Definition at line 3 of file WingSection.py.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 __init__()

```
def gebtaero.WingSection.WingSection.__init__ (
              self,
              Chord,
              ParameterA,
              NumberOfElements,
              SectionLength,
              CrossSection,
              Frame )
```

Definition at line 30 of file WingSection.py.

6.18.3 Member Function Documentation

6.18.3.1 GetChord()

```
def gebtaero.WingSection.WingSection.GetChord (
              self )
```

Definition at line 49 of file WingSection.py.

6.18.3.2 GetCrossSection()

```
def gebtaero.WingSection.WingSection.GetCrossSection (
              self )
```

Definition at line 69 of file WingSection.py.

6.18.3.3 GetFrame()

```
def gebtaero.WingSection.WingSection.GetFrame (
              self )
```

Definition at line 72 of file WingSection.py.

6.18.3.4 GetHalfChord()

```
def gebtaero.WingSection.WingSection.GetHalfChord (
              self )
```

Definition at line 52 of file WingSection.py.

6.18.3.5 GetNumberOfElements()

```
def gebtaero.WingSection.WingSection.GetNumberOfElements (
              self )
```

Definition at line 59 of file WingSection.py.

6.18.3.6 GetParameterA()

```
def gebtaero.WingSection.WingSection.GetParameterA (
              self )
```

Definition at line 42 of file WingSection.py.

6.18.3.7 GetSectionLength()

```
def gebtaero.WingSection.WingSection.GetSectionLength (
              self )
```

Definition at line 66 of file WingSection.py.

6.18.3.8 SetChord()

```
def gebtaero.WingSection.WingSection.SetChord (
              self,
              Chord )
```

Definition at line 45 of file WingSection.py.

6.18.3.9 SetNumberOfElements()

```
def gebtaero.WingSection.WingSection.SetNumberOfElements (
              self,
              NumberOfElements )
```

Definition at line 56 of file WingSection.py.

6.18.3.10 SetParameterA()

```
def gebtaero.WingSection.WingSection.SetParameterA (
              self,
              ParameterA )
```

Definition at line 39 of file WingSection.py.

6.18.3.11 SetSectionLength()

```
def gebtaero.WingSection.WingSection.SetSectionLength (
              self,
              SectionLength )
```

Definition at line 63 of file WingSection.py.

6.18.4 Member Data Documentation

6.18.4.1 Chord

```
gebtaero.WingSection.WingSection.Chord
```

Definition at line 31 of file WingSection.py.

6.18.4.2 CrossSection

```
gebtaero.WingSection.WingSection.CrossSection
```

Definition at line 36 of file WingSection.py.

6.18.4.3 Frame

`gebtaero.WingSection.WingSection.Frame`

Definition at line 37 of file WingSection.py.

6.18.4.4 HalfChord

`gebtaero.WingSection.WingSection.HalfChord`

Definition at line 35 of file WingSection.py.

6.18.4.5 NumberOfElements

`gebtaero.WingSection.WingSection.NumberOfElements`

Definition at line 33 of file WingSection.py.

6.18.4.6 ParameterA

`gebtaero.WingSection.WingSection.ParameterA`

Definition at line 32 of file WingSection.py.

6.18.4.7 SectionLength

`gebtaero.WingSection.WingSection.SectionLength`

Definition at line 34 of file WingSection.py.

The documentation for this class was generated from the following file:

- /home/bertrand/these/logiciels/programme/interface/src/gebtaero/WingSection.py

6.19 globaldatafun::writevec Interface Reference

Private Member Functions

- subroutine writeintvector (file_unit, vec)

  Write an integer vector to the file_unit ∗.
- subroutine writerealvector (file_unit, vec)

  Write a real vector to the file_unit ∗.

6.19.1 Detailed Description

Definition at line 105 of file GlobalDataFun.f90.

6.19.2 Member Function/Subroutine Documentation

6.19.2.1 writeintvector()

```
subroutine globaldatafun::writevec::writeintvector (
            integer, intent(in) file_unit,
            integer, dimension(:), intent(in) vec )  [private]
```

Write an integer vector to the file_unit *.

Parameters

in	*file_unit*	File unit to write the vector

Definition at line 375 of file GlobalDataFun.f90.

6.19.2.2 writerealvector()

```
subroutine globaldatafun::writevec::writerealvector (
            integer, intent(in) file_unit,
            real(dbl), dimension(:), intent(in) vec )  [private]
```

Write a real vector to the file_unit *.

Parameters

in	*file_unit*	File unit to write the vector

Definition at line 392 of file GlobalDataFun.f90.

The documentation for this interface was generated from the following file:

- /home/bertrand/these/logiciels/programme/src/GlobalDataFun.f90

Chapter 7

File Documentation

7.1 /home/bertrand/logiciels/gebtaero_frama/README.md File Reference

7.2 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/__init__.py File Reference

Namespaces

- gebtaero

7.3 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/CompositeBox.py File Reference

Classes

- class gebtaero.CompositeBox.CompositeBox

  Class interfacing the solver with 3D FEM calculix computation to obtain the cross section parameter from a composite box.

Namespaces

- gebtaero.CompositeBox

7.4 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/CompositePlate.py File Reference

Classes

- class gebtaero.CompositePlate.CompositePlate

Namespaces

- gebtaero.CompositePlate

7.5 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/CompositePly.py File Reference

Classes

- class gebtaero.CompositePly.CompositePly

Namespaces

- gebtaero.CompositePly

7.6 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/CrossSection.py File Reference

Classes

- class gebtaero.CrossSection.CrossSection

Namespaces

- gebtaero.CrossSection

7.7 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/ExternalMesh.py File Reference

Classes

- class gebtaero.ExternalMesh.ExternalMesh

Namespaces

- gebtaero.ExternalMesh

7.8 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/Frame.py File Reference

Classes

- class gebtaero.Frame.Frame

Namespaces

- gebtaero.Frame

7.9 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/GebtPlot.py File Reference

Classes

- class gebtaero.GebtPlot.GebtPlot

Namespaces

- gebtaero.GebtPlot

7.10 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/InputFile.py File Reference

Classes

- class gebtaero.InputFile.InputFile

Namespaces

- gebtaero.InputFile

7.11 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/IsoMaterial.py File Reference

Classes

- class gebtaero.IsoMaterial.IsoMaterial

Namespaces

- gebtaero.IsoMaterial

7.12 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/OrthoMaterial.py File Reference

Classes

- class gebtaero.OrthoMaterial.OrthoMaterial

Namespaces

- gebtaero.OrthoMaterial

7.13 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/Simulation.py File Reference

Classes

- class gebtaero.Simulation.Simulation

Namespaces

- gebtaero.Simulation

7.14 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/TimeFunction.py File Reference

Classes

- class gebtaero.TimeFunction.TimeFunction

Namespaces

- gebtaero.TimeFunction

7.15 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/utils.py File Reference

Namespaces

- gebtaero.utils

Functions

- def gebtaero.utils.CreatePeriodicEq (MeshFile, OffsetY=0., OffsetZ=0.)
- def gebtaero.utils.RunFbdFile (FileName)
- def gebtaero.utils.RunInpFile (FileName)
- def gebtaero.utils.RunFrdFile (FileName)
- def gebtaero.utils.RunParaviewScript (FileName)
- def gebtaero.utils.RunUnvConv (FileName, FileOut, Reduced='R')
- def gebtaero.utils.RemoveFiles ()
- def gebtaero.utils.RemoveMeshFiles ()
- def gebtaero.utils.ReadNodesField (self, FileName, FieldName)
- def gebtaero.utils.ReadFlexibilityFromDisp (FileName, nstrain, ncurv, Lx, tol, RigidX=False, RigidZ=False)
- def gebtaero.utils.ReadEigenVec (FileName)
- def gebtaero.utils.CorrelateTab (Tab1, Tab2, Index)
- def gebtaero.utils.ComputeElasticCenterFromFlexMat (FlexMat)
- def gebtaero.utils.ReadFromPipe (Command)
- def gebtaero.utils.ReadModesFromPipe (Command, output="modes")
- def gebtaero.utils.ReadLoadsFromPipe (Command, output="static")

7.16 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/Wing.py File Reference

Classes

- class gebtaero.Wing.Wing

Namespaces

- gebtaero.Wing

7.17 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/WingSection.py File Reference

Classes

- class gebtaero.WingSection.WingSection

Namespaces

- gebtaero.WingSection

7.18 /home/bertrand/these/logiciels/programme/src/Analysis.f90 File Reference

Functions/Subroutines

- subroutine analysis (nkp, nelem, ndof_el, nmemb, ncond_pt, nmate, nframe, ndistrfun, ncurv, coord, member, pt_condition, material, aerodyn_coef, niter, nstep, sol_pt, sol_mb, error, ncond_mb, ntimefun, frame, mb↩_condition, distr_fun, curvature, omega_a0, omega_a_tf, v_root_a0, v_root_a_tf, simu_time, time_function, analysis_flag, init_cond, nev, eigen_val, eigen_vec_pt, eigen_vec_mb, aero_flag, grav_flag)

  The file contains the analysis subroutine called by the main program.

7.18.1 Function/Subroutine Documentation

7.18.1.1 analysis()

```
subroutine analysis (
            integer, intent(in) nkp,
            integer, intent(in) nelem,
            integer, intent(in) ndof_el,
            integer, intent(in) nmemb,
            integer, intent(in) ncond_pt,
            integer, intent(in) nmate,
            integer, intent(in) nframe,
            integer, intent(in) ndistrfun,
            integer, intent(in) ncurv,
            real(dbl), dimension(nkp,ndim), intent(in) coord,
            integer, dimension(nmemb,memb_const), intent(in) member,
            type(prescriinf), dimension(ncond_pt), intent(inout) pt_condition,
            real(dbl), dimension(nmate,nstrn+nstrn,nstrn), intent(inout) material,
            real(dbl), dimension(nmate,8), intent(in) aerodyn_coef,
            integer, intent(in) niter,
            integer, intent(in) nstep,
            real(dbl), dimension(nstep,nkp,ndim+ndof_nd), intent(out) sol_pt,
            real(dbl), dimension(nstep,nelem,ndim+ndof_el), intent(out) sol_mb,
            character(*), intent(out) error,
            integer, intent(in) ncond_mb,
            integer, intent(in) ntimefun,
            real(dbl), dimension(nframe,ndim,ndim), intent(in) frame,
            type(prescriinf), dimension(ncond_mb), intent(inout) mb_condition,
            real(dbl), dimension(ndistrfun,nstrn), intent(in) distr_fun,
            real(dbl), dimension(ncurv,ndim), intent(in) curvature,
            real(dbl), dimension(ndim), intent(in) omega_a0,
            integer, dimension(ndim), intent(in) omega_a_tf,
            real(dbl), dimension(ndim), intent(in) v_root_a0,
            integer, dimension(ndim), intent(in) v_root_a_tf,
            real(dbl), dimension(2), intent(in) simu_time,
            type(timefunction), dimension(ntimefun), intent(in) time_function,
            integer, intent(in) analysis_flag,
            real(dbl), dimension(nelem,12+nstates), intent(inout) init_cond,
            integer, intent(inout) nev,
            real(dbl), dimension(2,nev+1), intent(out) eigen_val,
            real(dbl), dimension(nev+1,nkp,ndim+ndof_nd), intent(out) eigen_vec_pt,
            real(dbl), dimension(nev+1,nelem,ndim+ndof_el), intent(out) eigen_vec_mb,
            integer, intent(in) aero_flag,
            integer, intent(in) grav_flag )
```

The file contains the analysis subroutine called by the main program.

Parameters

in	*nkp*	ioaero::nkp
in	*nelem*	ioaero::nelem
in	*ndof_el*	ioaero::ndof_el
in	*nmemb*	ioaero::nmemb
in	*ncond_pt*	ioaero::ncond_pt
in	*nmate*	ioaero::nmate
in	*nframe*	ioaero::nframe
in	*ndistrfun*	ioaero::ndistrfun
in	*ncurv*	ioaero::ncurv
in	*coord*	ioaero::coord

Parameters

in	*member*	ioaero::member
in,out	*pt_condition*	ioaero::pt_condition
in,out	*material*	ioaero::material
in	*aerodyn_coef*	ioaero::aerodyn_coef
in	*niter*	ioaero::niter
in	*nstep*	ioaero::nstep
out	*sol_pt*	ioaero::sol_pt
out	*sol_mb*	ioaero::sol_mb
out	*error*	ioaero::error
in	*ncond_mb*	ioaero::ncond_mb
in	*ntimefun*	ioaero::ntimefun
in	*frame*	ioaero::frame
in,out	*mb_condition*	ioaero::mb_condition
in	*distr_fun*	ioaero::distr_fun
in	*curvature*	ioaero::curvature
in	*omega_a0*	ioaero::omega_a0
in	*v_root_a0*	ioaero::v_root_a0
in	*omega_a_tf*	ioaero::omega_a_tf
in	*v_root_a_tf*	ioaero::v_root_a_tf
in	*simu_time*	ioaero::simu_time
in	*time_function*	ioaero::time_function
in,out	*init_cond*	ioaero::init_cond
in	*analysis_flag*	ioaero::analysis_flag
in,out	*nev*	ioaero::nev
out	*eigen_val*	ioaero::eigen_val
out	*eigen_vec_pt*	ioaero::eigen_vec_pt
out	*eigen_vec_mb*	ioaero::eigen_vec_mb
in	*aero_flag*	ioaero::aero_flag
in	*grav_flag*	ioaero::grav_flag

Definition at line 7 of file Analysis.f90.

7.19 /home/bertrand/these/logiciels/programme/src/CPUtime.f90 File Reference

Modules

- module cputime

  A module use to calculate the computation time.

Functions/Subroutines

- subroutine, public cputime::tic

  Start the timer.
- real function, public cputime::toc ()

  Stop the timer.

Variables

- integer(8) cputime::start
- integer(8) cputime::rate
- integer(8) cputime::finish

7.20 /home/bertrand/these/logiciels/programme/src/EigenSolveMumps.f90 File Reference

Modules

- module eigenmumps

  This module contains the main routines needed for eigen value analysis and allow to call Arpack and MUMPS library.

Functions/Subroutines

- subroutine, public eigenmumps::eigensolvemumps (ndof_el, memb_info, v_root_a, omega_a, member, pt←_condition, niter, error, ncond_mb, mb_condition, distr_fun, dof_con, x, nev, eigen_val, eigen_vec, aero_flag, grav_flag)

  this routine solve the eigenproblem by linearising about a steady state x
- subroutine eigenmumps::arpack (nev, ncv, er, ei, vector, error)

  this subroutine make the interface with the arpack sparse eigensolver library
- subroutine eigenmumps::aw (u, nzM, irnM, jcnM, coef_mass, w)

  Solve the linear system $K*V = Z$ to avoid matrix inversion with solver MUMPS.

Variables

- type(dmumps_struc) eigenmumps::mumps_par

  an array containing the configuration parameters of MUMPS solver
- integer eigenmumps::ierr

  error code of the MUMPS solver

7.21 /home/bertrand/these/logiciels/programme/src/Element.f90 File Reference

Modules

- module element

  This module contains information and calculation for an element within a member.

Functions/Subroutines

- real(dbl) function, dimension(ndof_el+ndof_nd), public element::elemeqn (ndof_el)

  Compute the value of the Right Hand Side for a finite element.
- subroutine, public element::elemjacobian (ndof_el, niter, elemJac)

  Caculate the Jacobian matrix for each element.
- subroutine, public element::elemmass (ndof_el, elemM)

  Caculate the mass matrix for each element.
- subroutine, public element::extractelementproperties (elem_no, memb_info_i, x_elem, v_root_a, omega_a, ndof_el, init_elem, aero_flag, grav_flag)

  Extract element properties needed for element assembly.

Variables

- real(dbl) element::dl

  length of the element
- real(dbl) element::le

  the ending arc length of the current element
- real(dbl), dimension(ndim, ndim) element::ecab

  direction cosine matrix of the undeformed element
- real(dbl), dimension(nstrn, nstrn) element::eflex

  flexibility matrix of the elment
- real(dbl), dimension(nstrn, nstrn) element::emass

  inverse of the mass matrix of the element
- type(distriload), public element::load

  distributed load
- logical, public element::exist_load
- logical, public element::follower_load

  flags to indicate whether distributed load exist and whether they are follower forces
- real(dbl), dimension(ndim) element::ui
- real(dbl), dimension(ndim) element::theta
- real(dbl), dimension(ndim) element::fi
- real(dbl), dimension(ndim) element::mi
- real(dbl), dimension(ndim) element::e1gammad
- real(dbl), dimension(ndim) element::kappa
- real(dbl), dimension(ndim) element::epi
- real(dbl), dimension(ndim) element::hi
- real(dbl), dimension(ndim) element::vi
- real(dbl), dimension(ndim) element::omegai
- real(dbl), dimension(ndim) element::ev_i

  initial velocity of the mid point of the element
- real(dbl), dimension(ndim) element::eomega_a

  initial angular velocity of the element
- real(dbl), dimension(ndim, ndim) element::ect

  the transpose of the direction cosine matrix corresponding to elastic rotation
- real(dbl), dimension(ndim, ndim) element::ectcab

  eCT.Cab
- real(dbl), dimension(ndim, ndim) element::ecabhalfl

  eCab∗dL/2
- real(dbl), dimension(ndim, ndim) element::ectcabhalfl

  eCTCab∗dL/2
- real(dbl), dimension(ndim) element::uidot
- real(dbl), dimension(ndim) element::thetadot
- real(dbl), dimension(ndim) element::ctcabpdot
- real(dbl), dimension(ndim) element::ctcabhdot
- real(dbl) element::rho
- real(dbl) element::chord
- real(dbl) element::x_cg
- real(dbl) element::alpha
- real(dbl) element::alphadot
- real(dbl) element::hdot
- real(dbl) element::aw
- real(dbl) element::bw
- real(dbl) element::u
- real(dbl) element::hdotdot

- real(dbl) element::alphadotdot
- real(dbl) element::alpha_ac
- real(dbl) element::beta_ac
- real(dbl) element::beta
- integer element::a_flag
- integer element::g_flag
- real(dbl), dimension(nstates) element::lambda
- real(dbl), dimension(nstates) element::lambdadot
- real(dbl), dimension(nstates, 2 ∗nstates+2) element::p
- real(dbl), dimension(nstates, nstates) element::ident
- real(dbl) element::lambda0
- real(dbl), dimension(nstates, nstates) element::a
- real(dbl), dimension(nstates) element::b
- real(dbl), dimension(nstates) element::c
- real(dbl), dimension(ndim) element::dir_moment
- real(dbl), dimension(ndim) element::dir_lift
- real(dbl), dimension(ndim, ndim) element::jdir_moment
- real(dbl), dimension(ndim, ndim) element::jdir_lift
- real(dbl), dimension(ndim) element::jalpha_theta
- real(dbl), dimension(ndim) element::jalphadot_theta
- real(dbl), dimension(ndim) element::jhdot_theta
- real(dbl), dimension(ndim+ndim) element::jalphadot_ph
- real(dbl), dimension(ndim+ndim) element::jhdot_ph
- real(dbl), dimension(ndim+ndim) element::jalphadotdot_phdot
- real(dbl), dimension(ndim+ndim) element::jhdotdot_phdot
- real(dbl), dimension(ndim+ndim) element::jalphadotdot_ph
- real(dbl), dimension(ndim+ndim) element::jhdotdot_ph
- real(dbl), dimension(ndim+ndim) element::jlambda0_phdot
- real(dbl), dimension(nstates) element::jlambda0_lambda
- real(dbl), dimension(3, 18+nstates) element::jlift
- real(dbl), dimension(3, 18+nstates) element::jmoment
- real(dbl), dimension(ndim, ndim) element::ecaf
- real(dbl), dimension(ndim) element::wind

7.22 /home/bertrand/these/logiciels/programme/src/GlobalDataFun.f90 File Reference

Data Types

- interface globaldatafun::writevec

Modules

- module globaldatafun

  This module contains general-purpose global constants,I/O functions/subroutines and math functions/subroutines.

Functions/Subroutines

- logical function, public globaldatafun::fileopen (file_unit, file_name, sta_type, rw_type, error)

  To open an old or new file for reading or writing ∗.
- logical function, public globaldatafun::ioerror (message, error)

  Check the error of I/O processing ∗.
- character(20) function globaldatafun::itochar (n)

  Convert an integer to character ∗.
- logical function, public globaldatafun::memoryerror (vari_name, error)

  Check the error of memory allocation ∗.
- subroutine, public globaldatafun::titleprint (file_unit, title)

  To print a title for a block of data ∗.
- subroutine, public globaldatafun::writeerror (EIN, error)

  Write error to the echo file ∗.
- subroutine globaldatafun::writeintvector (file_unit, vec)

  Write an integer vector to the file_unit ∗.
- subroutine globaldatafun::writerealvector (file_unit, vec)

  Write a real vector to the file_unit ∗.
- subroutine, public globaldatafun::ct_theta (theta, eCT, ekttek, eCTtheta)

  Calculate eC^T derivative w.r.t theta ∗ return derivative and ekttek ∗.
- real(dbl) function, dimension(3, 3), public globaldatafun::ct_theta_t (theta, eCT, x)

  Calculate $e\dot{C}^T.x$ *derivative w.r.t* $\dot{theta}$ *∗.*
- real(dbl) function, dimension(3, 3), public globaldatafun::dircosinetrodrigues (theta)

  Calculate the transpose of the direction cosine in ∗ terms of rodrigues parameters ∗.
- subroutine, public globaldatafun::invert (matrix_in, matrix, vari_name, error)

  Invert a small square matrix ∗.
- real(dbl) function, dimension(size(mat, 1), size(mat, 2)), public globaldatafun::matmul3 (mat, vec)

  Multiply a rank 3 matrix with a vector with every colum ∗ of the resulting matrix is equal to the multiplication ∗.
- real(dbl) function, public globaldatafun::norm (vector)

  Calculate the L2 norm of a real vector ∗.
- real(dbl) function, dimension(size(vec1), size(vec2)), public globaldatafun::outerproduct (vec1, vec2)

  Calculate the outer product of two vectors ∗.
- real(dbl) function, dimension(3, 3), public globaldatafun::tilde (vect)

  Carry out the tilde operation for a real vector ∗.
- real(dbl) function, dimension(3), public globaldatafun::crossproduct (a, b)

  Carry out cross product of two real vectors ∗.
- subroutine, public globaldatafun::insert1delement (nz, tmpR, irn, jcn, elemCoef1D, str_r1, str_c1, str_r2, str_c2, str_r3, str_c3, str_r4, str_c4)

  Insert a real matrix into the 1D coefficient matrix.
- subroutine, public globaldatafun::extract2delement (nz, irn, jcn, elemCoef1D, tmpR, str_r1, str_c1)

  Back a 2D array from the 1D coefficient matrix.
- real(dbl) function, dimension(nsize), public globaldatafun::matmul_sparse (vector, nsize, ne, irn, jcn, matrix1D)

  Matmul(vector, matrix) with matrix stored in a spare format.
- real(dbl) function, dimension(n, 2 ∗n+2), public globaldatafun::peters (n)

  Functions used for the Finite State Unsteady Thin Airfoil Theory of Peters see Introduction to Structural Dynamics and Aeroelasticity by Hodges p 139.
- real(dbl) function, dimension(n, n), public globaldatafun::mata (n)

  Compute the Peters A matrix.
- real(dbl) function, dimension(n, n) globaldatafun::matd (n)

  Compute the Peters D matrix.

- real(dbl) function, dimension(n) globaldatafun::vecb (n)

    Compute the Peters B vector.
- real(dbl) function, dimension(n) globaldatafun::vecc (n)

    Compute Peters C vector.
- real(dbl) function, dimension(n) globaldatafun::vecd (n)

    Compute Peters D vector.
- real(dbl) function, dimension(n, n) globaldatafun::prod (Vec1, Vec2, n)

    Compute the product of two square matrix.
- real(dbl) function globaldatafun::factoriel (n)

    Compute the value of factoriel(n)

Variables

- integer, parameter, public globaldatafun::ndim =3

    All the beams could behavior in the 3D space.
- integer, parameter, public globaldatafun::ndof_nd =12

    degrees of freedom per node/element is 12.
- integer, parameter, public globaldatafun::nstrn =6

    Number of strain measures/dofs in Timoshenko model.
- integer, parameter, public globaldatafun::memb_const =7

    Number of labels needed for member properties.
- integer, parameter, public globaldatafun::nstates = 6

    Number of induces-flow states in Peters theory.
- integer, parameter, public globaldatafun::dbl =SELECTED_REAL_KIND(15, 307)
- real(dbl), parameter, public globaldatafun::pi = 3.1415926535897932D0
- real(dbl), parameter, public globaldatafun::deg_2_rad = 1.7453292519943296D-2

    the ratio between radians and degrees
- real(dbl), parameter, public globaldatafun::rad_2_deg = 5.7295779513082321D1

    convert radian to degree
- real(dbl), parameter, public globaldatafun::tolerance = EPSILON(1.0_DBL)

    a smart number of the double precision real number
- real(dbl), parameter, public globaldatafun::grav = 9.81

    gravity acceleration
- real(dbl), dimension(3, 3), parameter, public globaldatafun::i3 = RESHAPE((/1.D0, 0.D0, 0.D0, 0.D0, 1.D0, 0.D0, 0.D0, 0.D0, 1.D0/), (/3,3/))

    The 3x3 identity matrix.
- real(dbl), dimension(3), parameter, public globaldatafun::e1 =(/1._DBL,0._DBL,0._DBL/)

    The e1 unit vector.
- integer, public globaldatafun::in_stat

    flag to indicate if the I/O process is successful: if positive, an error occured; if negative, an end-of-file or end-of-record condition occurred; zero, no error, end-of-file, or end-of-record condition occurred.
- integer, public globaldatafun::allo_stat

    flag to indicate status of allocating memory
- character(∗), parameter, public globaldatafun::fmt_real ='ES15.7'

    format for output real numbers
- character(∗), parameter, public globaldatafun::fmt_int ='I8'

    format for output integer numbers
- integer, public globaldatafun::runmod =0

    Define the output behavior of the program; 0: legacy mode of the computation code with output of a .out text file; 1: mode compatible with the python pre/postrpocessor (argument -p in the terminal), 2: silent mode (argument -s in the terminal)

- integer, public globaldatafun::arpack_mod =0

  parameter WHICH of arpack solver (1:LI, 2:LM, 3:LR, 4:SR, default : LM in dnaupd and LI in dneupd) =>cf Arpack doc
- integer, public globaldatafun::eigen_output =0

  define wich eigenvalue data to output (0: eigenvalues and eigenvectors, 1: eigenvalues only)
- character(10), public globaldatafun::solver ='MUMPS'

  linear solver used (HSL : ddep.f, mc19.f + ma28 or MUMPS : linux library)
- integer, public globaldatafun::flutter_flag =0

  used in temporal simulation : 0= deformation are under a "flutter" state; 1= deformation are over a "flutter" state
- real(dbl), public globaldatafun::flutter_limit

  the value of maximale angular deformaton use to trigger the flutter flag

7.23 /home/bertrand/these/logiciels/programme/src/InternalData.f90 File Reference

Data Types

- type internaldata::memberinf

  structure containing the caracteritics of a finite element.

Modules

- module internaldata

  This module contains the variables needed internally in the program. Not necessary to be defined in the outside environment.

Variables

- logical, parameter internaldata::debug =.FALSE.
- integer, parameter internaldata::iout =30
- character(64) internaldata::deb_name
- integer internaldata::nsize
- integer, dimension(:,:), allocatable internaldata::dof_all
- integer, dimension(:,:), allocatable internaldata::follower_all
- real(dbl), dimension(:,:), allocatable internaldata::cond_all
- real(dbl), dimension(:,:), allocatable internaldata::init_memb
- integer internaldata::init_flag
- real(dbl) internaldata::two_divide_dt

  2/dt
- integer internaldata::assemble_flag =0

  for the purpose to share the routines between assembly of stiffness matrix and mass matrix
- integer, dimension(:,:), allocatable internaldata::index_kp

  the starting row and column for each kp
- integer, dimension(:,:), allocatable internaldata::index_mb

  the starting row and column for each member
- real(dbl), dimension(ndim) internaldata::xyz_pt1

  the coordinate of the starting point of the first member
- integer, parameter internaldata::nzelemmax =500
- integer internaldata::nemax

7.24 /home/bertrand/these/logiciels/programme/src/IOaero.f90 File Reference

Modules

- module ioaero

  This module handle I/O of the computation code. Allow to read a .dat command file possibly with a .ini file and output a .out text output file or/and a folder with .vtk file intended to be used with paraview.

Functions/Subroutines

- subroutine, public ioaero::input
- subroutine, public ioaero::output
- subroutine, public ioaero::outputvtk

Variables

- integer, parameter, private ioaero::char_len =256
- integer, parameter ioaero::in =10
- character(char_len) ioaero::inp_name
- integer, parameter, public ioaero::ein =20

  file for echoing the inputs: inp_name.ech
- character(char_len+3) ioaero::ech_name
- integer, parameter ioaero::out =40

  file for output: inp_name.out
- character(char_len+3) ioaero::out_name
- integer, parameter ioaero::init =50

  file for initial conditions: inp_name.ini
- character(char_len+3) ioaero::init_name
- integer, public ioaero::nkp

  number of key points
- integer, public ioaero::nelem

  total number of elements
- integer, public ioaero::nmemb

  number of members
- integer, public ioaero::nmate

  number of cross-sectional properties sets
- integer, public ioaero::nframe

  number of frames
- integer, public ioaero::ncond_pt

  number of point conditions for concentrated loads and boundary conditions
- integer, public ioaero::ndistrfun

  number of distributed functions
- integer, public ioaero::ncurv

  number of initial curvatures/twists
- integer, public ioaero::analysis_flag

  0: static analysis; 1: steady state response; 2: transient analysis; 3: eigenvalue analysis
- integer, public ioaero::nev

  number of frequencies and modeshapes.
- integer, public ioaero::aero_flag

  0: no aero analasys; 1: stationary aerodynamic, 2: unsteady aerodynamic

- integer, public ioaero::grav_flag
- integer, public ioaero::ncond_mb

  number of member conditions for distributed loads
- integer, public ioaero::ntimefun

  number of time functions
- integer, public ioaero::niter

  number of maximum iterations
- integer, public ioaero::nstep

  number of time steps/load steps
- integer, public ioaero::nvtk

  number of the aerodynamic cycle
- integer, dimension(:,:), allocatable, public ioaero::member

  member property array: member(nmemb,MEMB_CONST)
- integer, public ioaero::ndof_el

  dofs per element: 12 for static analysis, 18 for dynamic analysis, +Ns if aero_flag = 3
- integer, dimension(ndim), public ioaero::omega_a_tf

  time function numbers for the angular velocity of frame a
- integer, dimension(ndim), public ioaero::v_root_a_tf

  time function numbers for the velocity of the starting point of the first member
- real(dbl), dimension(:,:), allocatable, public ioaero::coord

  nodal coordinates: coord(nkp,NDIM)
- real(dbl), dimension(:,:,:), allocatable, public ioaero::material

  flexibility matrix: (nmate,12,6)
- real(dbl), dimension(:,:), allocatable, public ioaero::aerodyn_coef

  2D aerodynamic coefficient : (nmate,:)
- real(dbl), dimension(:,:,:), allocatable, public ioaero::frame

  member frames: (nframe,3,3)
- real(dbl), dimension(:,:), allocatable, public ioaero::distr_fun

  prescribed functions: (ndistrfun,6)
- real(dbl), dimension(:,:), allocatable, public ioaero::curvature

  curvatures: (ncurv,NDIM)
- real(dbl), dimension(:,:,:), allocatable, public ioaero::sol_pt

  solutions for points sol_pt(nstep,nkp,NDIM+NDOF_ND)
- real(dbl), dimension(:,:,:), allocatable, public ioaero::sol_mb

  solutions for member sol_mb(nstep,nelem,NDIM+ndof_el): nelem: total number of elements
- real(dbl), dimension(2), public ioaero::simu_time

  start and end time of the simulation.
- real(dbl), dimension(ndim), public ioaero::omega_a0

  the magnitude of angular velocity of frame a
- real(dbl), dimension(ndim), public ioaero::v_root_a0

  the magnitude of linear velocity of the starting point of the first member
- real(dbl), dimension(:,:), allocatable, public ioaero::init_cond

  initial conditions: init_cond(nelem,12); init_cond(nelem,1:6) for initial displacements/rotations init_cond(nelem,7:12) for initial velocities init_cond(nelem,13:12+NSTATES) Peters finite state parameter at time t+dt
- real(dbl), dimension(:,:), allocatable, public ioaero::eigen_val

  arrays for holding eigenvalues and eigenvectors
- real(dbl), dimension(:,:,:), allocatable, public ioaero::eigen_vec_pt

  arrays for holding eigenvalues and eigenvectors
- real(dbl), dimension(:,:,:), allocatable, public ioaero::eigen_vec_mb

  arrays for holding eigenvalues and eigenvectors
- type(prescriinf), dimension(:), allocatable, public ioaero::pt_condition

prescribed information concentrated at nodes
- type(prescriinf), dimension(:), allocatable, public ioaero::mb_condition

    prescribed information distributed along beam members
- type(timefunction), dimension(:), allocatable, public ioaero::time_function

    time functions
- character(char_len), public ioaero::velocity_str = ''
- integer ioaero::arpack

    Dummy input variable for ARPACK_MOD.
- integer ioaero::eigenoutput

    Dummy input variable for EIGEN_OUTPUT.
- character(300), public ioaero::error

7.25 /home/bertrand/these/logiciels/programme/src/mainAero.f90 File Reference

Functions/Subroutines

- program gebt

    This program launch the aeroelastic simulation by reading the .dat file (and possibly the .ini file), execute the analysis subroutine and output the result in .out test file or/and vtk files (readable in paraview)

7.25.1 Function/Subroutine Documentation

7.25.1.1 gebt()

```
program gebt ( )
```

This program launch the aeroelastic simulation by reading the .dat file (and possibly the .ini file), execute the analysis subroutine and output the result in .out test file or/and vtk files (readable in paraview)

Definition at line 87 of file mainAero.f90.

7.26 /home/bertrand/these/logiciels/programme/src/Member.f90 File Reference

Modules

- module member

    This module assemles within a member without considering the particular conditions of the end points.

Functions/Subroutines

- subroutine, public member::assemblememberrhs (ndof_el, memb_info_i, v_root_a, omega_a, x_memb, rhs↩_memb, aero_flag, grav_flag, init_cond)

  Assemble the equations for each member ∗.
- subroutine, public member::assemblememberjacobian (ndof_el, niter, memb_info_i, v_root_a, omega_a, x↩_memb, nz_memb, irn_memb, jcn_memb, coef_memb, aero_flag, grav_flag)

  Assemble the Jacobian for each member ∗.
- subroutine, public member::extractmemberproperties (memb_no, memb_info_i, member, ncond_mb, mb_↩condition, distr_fun, error, init_cond)

  Extract member properties ∗.

Variables

- integer, public member::ndiv

  number of divisions
- integer, public member::ncol_memb

  the total number of columns of the member

7.27 /home/bertrand/these/logiciels/programme/src/Preprocess.f90 File Reference

Modules

- module prepromodule

  This module preprocess the finite element model including connectivity and member information. This information are time step indepedent.

Functions/Subroutines

- subroutine, public prepromodule::preprocess (nkp, nelem, ndof_el, member, material, frame, coord, curvature, dof_con, memb_info, error, aero_flag, grav_flag, aerodyn_coef)

  Obtaining the connection condition for each key point ∗ if a point is connected to more than one member, it is ∗ a connection point, otherwise it is a boundary point. ∗ It also calculates the size of the problem ∗.
- subroutine prepromodule::memberproperties (memb_no, ndof_el, member, material, frame, coord, curvature, memb_info_i, error, aero_flag, grav_flag, aerodyn_coef)

  Extract member properties for each division ∗.
- real(dbl) function prepromodule::curvebeamfun (kn, mL, kn2, k12, kn4, xvar)

  Function for evaluating arc length of initially curved and twisted beams.
- real(dbl) function prepromodule::rtbis (func, kn, mL, kn2, k12, kn4, x1, x2, xacc, maxit, error)

  Use biosection to find root of a function ∗ from the book of Numerical Recipes ∗.

Variables

- integer prepromodule::ndiv

  member::ndiv
- integer prepromodule::ncol_memb

  member::ncol_memb

7.28 /home/bertrand/these/logiciels/programme/src/PrescribedCondition.f90 File Reference

Data Types

- type prescribedcondition::prescriinf

    Define the prescribed condition.
- type prescribedcondition::distriload

    Define the distributed load condition.

Modules

- module prescribedcondition

    A module for defining prescribed conditions including both concentrated information and distributed information.

Functions/Subroutines

- subroutine, public prescribedcondition::existpi (location, prescri_inf, exist_pi, follower_pi)

    Determine whether prescribed condition exist, and whether any of such conditions is a follower condition.
- type(distriload) function, public prescribedcondition::getdistributedload (memb_no, mb_condition, distr_fun)

    Obtain the distributed load condition.
- real(dbl) function, dimension(nstrn), public prescribedcondition::getload (flag, dL, Le, eCT, load, follower_↩ load)

    Obtain the distributed load, transform if follower.
- real(dbl) function, dimension(nstrn, 3), public prescribedcondition::getloadj (flag, dL, Le, eCTtheta, load)

    Obtain the jacobian due to follower distributed load $*$.
- subroutine, public prescribedcondition::getprescribeddof (nkp, pt_condition, kp_dof, kp_follower)

    Obtain Prescribed dof and follower condition $*$.
- subroutine, public prescribedcondition::getprescribedval (nkp, pt_condition, kp_cond)

    Obtain Prescribed value $*$.
- elemental type(prescriinf) function, public prescribedcondition::initpi ()

    Initialize Prescribed Conditions $*$.
- type(prescriinf) function, public prescribedcondition::inputechoprescribedconditions (IN, EIN, error)

    Input and echo Prescribed Conditions $*$.
- real(dbl) function, dimension(nstrn), public prescribedcondition::updatefollower (kp_dof, kp_follower, kp_↩ cond, x_pt)

    Obtain Prescribed DOF and value needed for rhs assume only follower force/moments, and no displacements or rotations can be prescribed for follower quantities. And the first three prescribed dofs for the point with a follower component should be either 7 8 9 or 10 11 12. This assumption is made for the easiness to locate the rotation parameters.
- subroutine, public prescribedcondition::updatepi (prescri_inf, time_fun, t)

    Update the prescribed information based on the current time the value is stored in: value_current.
- real(dbl) function, dimension(size(vec), size(vec)), public prescribedcondition::followerj (follower, vec, eC↩ Ttheta)

    Calculating the Jacobian due to follower conditions $J = diffC^T.vec/diff\theta$*, return a 3x3 matrix with ith column corresponding to the derivative withe respect to* θ_i.
- real(dbl) function prescribedcondition::loadintegration (flag, dL, Le, func)

    Caculate the load using Chebychev polynomials $*$.
- real(dbl) function, dimension(size(vec)) prescribedcondition::transferfollower (follower, vec, CT)

    Transfer follower according to C^ T.vec note vec is a 3x1 vector, and whether a component is a follower or not is determined by follower.
- elemental type(prescriinf) function, public prescribedcondition::initpiaero (i)

7.29 /home/bertrand/these/logiciels/programme/src/SolveMumps.f90 File Reference

Modules

- module solvemumps

  This module contains the linear & nonlinear solver interfaced with MUMPS direct solver library.

Functions/Subroutines

- subroutine, public solvemumps::linearsolutionmumps (ndof_el, memb_info, v_root_a, omega_a, member, error, ncond_mb, mb_condition, distr_fun, dof_con, x, aero_flag, grav_flag, init_cond)

  The linear solver is basically the Newton-Raphson with ∗ initial guess equal to zero and only uses one iterations ∗.
- subroutine, public solvemumps::newtonraphsonmumps (ndof_el, memb_info, v_root_a, omega_a, member, niter, error, ncond_mb, mb_condition, distr_fun, dof_con, x, aero_flag, grav_flag, init_cond)

  Use Newton-Raphson method to solve the nonlinear system ∗.
- subroutine linesearch

  Use line search to improve the convergence of Newton Raphson method, modified from the book: Numerical Recipes∗.
- subroutine, public solvemumps::extractsolution (ndof_el, member, coord, memb_info, x, dof_con, sol_pt_i, sol_mb_i)

  The subroutine extracts the solution for each key point and each member from the solution vector ∗.
- subroutine, public solvemumps::extractelementvalues (ndof_el, member, x, sol_mb_i)

  The subroutine extracts elemental values from ∗ the solution vector ∗.
- subroutine, public solvemumps::insertelementvalues (ndof_el, member, x, init_cond)

  The subroutine insert the elemental values into the solution vector: needed for initial guess for starting time marching: replace the first six valumes for each element with given initial conditions ∗.
- real(dbl) function, dimension(size(sol_mb_i, 1), 6), public solvemumps::ctcabph (niter, member, memb_info, sol_mb_i)

  Transfer the vector PH (linear and angular momenta) form frame B to frame a.

Variables

- type(dmumps_struc) solvemumps::mumps_par

  an array containing the configuration parameters of MUMPS solver
- integer solvemumps::ierr

  error code of the MUMPS solver

7.29.1 Function/Subroutine Documentation

7.29.1.1 linesearch()

```
subroutine newtonraphsonmumps::linesearch ( )  [private]
```

Use line search to improve the convergence of Newton Raphson method, modified from the book: Numerical Recipes∗.

Definition at line 303 of file SolveMumps.f90.

7.30 /home/bertrand/these/logiciels/programme/src/System.f90 File Reference

Modules

- module system

    This module assemles the system including the coefficient matrix (jacobian matrix) and the right hand side (negative of the equation values) ∗.

Functions/Subroutines

- subroutine, public system::assemblejacobian (ndof_el, niter, memb_info, v_root_a, omega_a, member, error, ncond_mb, mb_condition, distr_fun, dof_con, x, aero_flag, grav_flag, init_cond)

    Assemble the coefficient matrix of the beam system ∗.
- subroutine assemblepointj (flag, kp, nrow_mb, ncol_mb)

    Assemble the Jacobian related with trailing points.
- subroutine, public system::assemblerhs (ndof_el, memb_info, v_root_a, omega_a, member, error, ncond_mb, mb_condition, distr_fun, dof_con, x, rhs, aero_flag, grav_flag, init_cond)

    Assemble the right hand side.
- subroutine assemblepointrhs (flag, term_pt, rhs_tm, nrow_mb)

    Assemble the Jacobian related with trailing points.
- subroutine system::pointfollowerj (flag, nrow, ncol, eCTtheta)

    Add the contribution to Jacobian matrix due to follower point force or moments.

Variables

- real(dbl), dimension(nstrn) system::x_pt

    the solution from the previous step for a key point.
- integer, dimension(nstrn) system::kp_dof

    prescribed dof
- real(dbl), dimension(nstrn) system::kp_cond

    prescribed value
- integer, dimension(nstrn) system::kp_follower

    follower condition
- integer, public system::ne

    Number of nonzero coefficients.
- integer, dimension(:), allocatable, public system::irn

    line index of nonzero coefficients
- integer, dimension(:), allocatable, public system::jcn

    column index of nonzero coefficients
- real(dbl), dimension(:), allocatable, public system::coef

    value of nonzero coefficients

7.30.1 Function/Subroutine Documentation

7.30.1.1 assemblepointj()

```
subroutine assemblejacobian::assemblepointj (
            integer, intent(in) flag,
            integer, intent(in) kp,
            integer, intent(in) nrow_mb,
            integer, intent(in) ncol_mb )  [private]
```

Assemble the Jacobian related with trailing points.

Parameters

in	*flag*	indicating whether it is the starting point or the ending point: -1-starting, 1-ending
in	*kp*	the point number

Definition at line 203 of file System.f90.

7.30.1.2 assemblepointrhs()

```
subroutine assemblerhs::assemblepointrhs (
            integer, intent(in) flag,
            integer, intent(in) term_pt,
            real(dbl), dimension(:), intent(in) rhs_tm,
            integer, intent(in) nrow_mb )  [private]
```

Assemble the Jacobian related with trailing points.

Parameters

in	*flag*	indicating whether it is the starting point or the ending point: -1-starting, 1-ending
in	*term← _pt*	the point number
in	*rhs_tm*	the right hand side of the member associated with the end point

Definition at line 418 of file System.f90.

7.31 /home/bertrand/these/logiciels/programme/src/TimeFunction.f90 File Reference

Data Types

- type timefunctionmodule::timefunction

Modules

- module timefunctionmodule

  A module for defining time functions needed for both prescribed concentrated and distributed conditions.

Functions/Subroutines

- real(dbl) function, public timefunctionmodule::gettimefunction (tf, t)

  get the time function value for any arbitrary time from a piecewise linear function or harmonic function decide where t is located, then interpret the value
- elemental type(timefunction) function, public timefunctionmodule::inittf ()

  Initialize the time function.
- type(timefunction) function, public timefunctionmodule::inputechotimefunctions (IN, EIN, error)

  Input and echo Time Functions.
- real(dbl) function, dimension(size(vec)), public timefunctionmodule::currentvalues (vec, vec_tf, time_fun, time_current)

  Evaluate current function based on magnitude and time.

Index