

GEBTAero

18.09

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>GEBTAero</b>	<b>1</b>
<b>2</b>	<b>Modules Index</b>	<b>3</b>
2.1	Modules List . . . . .	3
<b>3</b>	<b>Data Type Index</b>	<b>5</b>
3.1	Data Types List . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Module Documentation</b>	<b>9</b>
5.1	cputime Module Reference . . . . .	9
5.1.1	Detailed Description . . . . .	9
5.1.2	Function/Subroutine Documentation . . . . .	9
5.1.2.1	tic() . . . . .	9
5.1.2.2	toc() . . . . .	10
5.1.3	Variable Documentation . . . . .	10
5.1.3.1	finish . . . . .	10
5.1.3.2	rate . . . . .	10
5.1.3.3	start . . . . .	10
5.2	eigenmumps Module Reference . . . . .	10
5.2.1	Detailed Description . . . . .	11
5.2.2	Function/Subroutine Documentation . . . . .	11
5.2.2.1	arpack() . . . . .	11

5.2.2.2	<code>aw()</code>	12
5.2.2.3	<code>eigensolvemumps()</code>	12
5.2.3	Variable Documentation	13
5.2.3.1	<code>ierr</code>	13
5.2.3.2	<code>mumps_par</code>	13
5.3	element Module Reference	13
5.3.1	Detailed Description	15
5.3.2	Function/Subroutine Documentation	16
5.3.2.1	<code>elemeqn()</code>	16
5.3.2.2	<code>elemjacobian()</code>	16
5.3.2.3	<code>elemmass()</code>	16
5.3.2.4	<code>extractelementproperties()</code>	17
5.3.3	Variable Documentation	17
5.3.3.1	<code>a</code>	18
5.3.3.2	<code>a_flag</code>	18
5.3.3.3	<code>alpha</code>	18
5.3.3.4	<code>alpha_ac</code>	18
5.3.3.5	<code>alphadot</code>	18
5.3.3.6	<code>alphadotdot</code>	18
5.3.3.7	<code>aw</code>	19
5.3.3.8	<code>b</code>	19
5.3.3.9	<code>beta</code>	19
5.3.3.10	<code>beta_ac</code>	19
5.3.3.11	<code>bw</code>	19
5.3.3.12	<code>c</code>	19
5.3.3.13	<code>chord</code>	20
5.3.3.14	<code>ctcabhdot</code>	20
5.3.3.15	<code>ctcabpdot</code>	20
5.3.3.16	<code>dir_lift</code>	20
5.3.3.17	<code>dir_moment</code>	20

5.3.3.18	dl	20
5.3.3.19	e1gammad	21
5.3.3.20	ecab	21
5.3.3.21	ecabhalf	21
5.3.3.22	ecaf	21
5.3.3.23	ect	21
5.3.3.24	ectcab	22
5.3.3.25	ectcabhalf	22
5.3.3.26	eflex	22
5.3.3.27	emass	22
5.3.3.28	eomega_a	22
5.3.3.29	epi	23
5.3.3.30	ev_i	23
5.3.3.31	exist_load	23
5.3.3.32	fi	23
5.3.3.33	follower_load	23
5.3.3.34	g_flag	23
5.3.3.35	hdot	24
5.3.3.36	hdotdot	24
5.3.3.37	hi	24
5.3.3.38	ident	24
5.3.3.39	jalpha_theta	24
5.3.3.40	jalphadot_ph	24
5.3.3.41	jalphadot_theta	25
5.3.3.42	jalphadotdot_ph	25
5.3.3.43	jalphadotdot_phdot	25
5.3.3.44	jdir_lift	25
5.3.3.45	jdir_moment	25
5.3.3.46	jhdot_ph	25
5.3.3.47	jhdot_theta	26

5.3.3.48	jhdotdot_ph	26
5.3.3.49	jhdotdot_phdot	26
5.3.3.50	jlambda0_lambda	26
5.3.3.51	jlambda0_phdot	26
5.3.3.52	jlift	26
5.3.3.53	jmoment	27
5.3.3.54	kappa	27
5.3.3.55	lambda	27
5.3.3.56	lambda0	27
5.3.3.57	lambdadot	27
5.3.3.58	le	27
5.3.3.59	load	28
5.3.3.60	mi	28
5.3.3.61	omegai	28
5.3.3.62	p	28
5.3.3.63	rho	28
5.3.3.64	theta	28
5.3.3.65	thetadot	29
5.3.3.66	u	29
5.3.3.67	ui	29
5.3.3.68	uidot	29
5.3.3.69	vi	29
5.3.3.70	wind	29
5.3.3.71	x_cg	30
5.4	gebtaero Namespace Reference	30
5.5	gebtaero.CompositeBox Namespace Reference	30
5.6	gebtaero.CompositePlate Namespace Reference	30
5.7	gebtaero.CompositePly Namespace Reference	30
5.8	gebtaero.CrossSection Namespace Reference	31
5.9	gebtaero.ExternalMesh Namespace Reference	31

5.10	<a href="#">gebtaero.Frame Namespace Reference</a>	31
5.11	<a href="#">gebtaero.GebtPlot Namespace Reference</a>	31
5.12	<a href="#">gebtaero.InputFile Namespace Reference</a>	31
5.13	<a href="#">gebtaero.IsoMaterial Namespace Reference</a>	31
5.14	<a href="#">gebtaero.OrthoMaterial Namespace Reference</a>	31
5.15	<a href="#">gebtaero.Simulation Namespace Reference</a>	32
5.16	<a href="#">gebtaero.TimeFunction Namespace Reference</a>	32
5.17	<a href="#">gebtaero.utils Namespace Reference</a>	32
5.17.1	<a href="#">Function Documentation</a>	32
5.17.1.1	<a href="#">ComputeElasticCenterFromFlexMat()</a>	32
5.17.1.2	<a href="#">CorrelateTab()</a>	33
5.17.1.3	<a href="#">CreatePeriodicEq()</a>	33
5.17.1.4	<a href="#">ReadEigenVec()</a>	33
5.17.1.5	<a href="#">ReadFlexibilityFromDisp()</a>	33
5.17.1.6	<a href="#">ReadFromPipe()</a>	33
5.17.1.7	<a href="#">ReadLoadsFromPipe()</a>	34
5.17.1.8	<a href="#">ReadModesFromPipe()</a>	34
5.17.1.9	<a href="#">ReadNodesField()</a>	34
5.17.1.10	<a href="#">RemoveFiles()</a>	34
5.17.1.11	<a href="#">RemoveMeshFiles()</a>	34
5.17.1.12	<a href="#">RunFbdFile()</a>	35
5.17.1.13	<a href="#">RunFrdFile()</a>	35
5.17.1.14	<a href="#">RunInpFile()</a>	35
5.17.1.15	<a href="#">RunParaviewScript()</a>	35
5.17.1.16	<a href="#">RunUnvConv()</a>	35
5.18	<a href="#">gebtaero.Wing Namespace Reference</a>	35
5.19	<a href="#">gebtaero.WingSection Namespace Reference</a>	36
5.20	<a href="#">globaldatafun Module Reference</a>	36
5.20.1	<a href="#">Detailed Description</a>	38
5.20.2	<a href="#">Function/Subroutine Documentation</a>	38

5.20.2.1	<code>crossproduct()</code>	38
5.20.2.2	<code>ct_theta()</code>	39
5.20.2.3	<code>ct_theta_t()</code>	39
5.20.2.4	<code>dircosinetrodrigues()</code>	39
5.20.2.5	<code>extract2delement()</code>	40
5.20.2.6	<code>factoriel()</code>	40
5.20.2.7	<code>fileopen()</code>	40
5.20.2.8	<code>insert1delement()</code>	41
5.20.2.9	<code>invert()</code>	41
5.20.2.10	<code>ioerror()</code>	41
5.20.2.11	<code>itochar()</code>	42
5.20.2.12	<code>mata()</code>	42
5.20.2.13	<code>matd()</code>	42
5.20.2.14	<code>matmul3()</code>	42
5.20.2.15	<code>matmul_sparse()</code>	43
5.20.2.16	<code>memoryerror()</code>	43
5.20.2.17	<code>norm()</code>	43
5.20.2.18	<code>outerproduct()</code>	43
5.20.2.19	<code>peters()</code>	44
5.20.2.20	<code>prod()</code>	44
5.20.2.21	<code>tilde()</code>	44
5.20.2.22	<code>titleprint()</code>	44
5.20.2.23	<code>vecb()</code>	45
5.20.2.24	<code>vecc()</code>	45
5.20.2.25	<code>vecd()</code>	45
5.20.2.26	<code>writeerror()</code>	45
5.20.2.27	<code>writeintvector()</code>	46
5.20.2.28	<code>writerealvector()</code>	46
5.20.3	Variable Documentation	46
5.20.3.1	<code>allo_stat</code>	46



5.20.3.2	arpack_mod	46
5.20.3.3	dbl	47
5.20.3.4	deg_2_rad	47
5.20.3.5	e1	47
5.20.3.6	eigen_output	47
5.20.3.7	flutter_flag	47
5.20.3.8	flutter_limit	48
5.20.3.9	fmt_int	48
5.20.3.10	fmt_real	48
5.20.3.11	grav	48
5.20.3.12	i3	48
5.20.3.13	in_stat	49
5.20.3.14	memb_const	49
5.20.3.15	ndim	49
5.20.3.16	ndof_nd	49
5.20.3.17	nstates	49
5.20.3.18	nstrn	50
5.20.3.19	pi	50
5.20.3.20	rad_2_deg	50
5.20.3.21	runmod	50
5.20.3.22	solver	50
5.20.3.23	tolerance	51
5.21	internaldata Module Reference	51
5.21.1	Detailed Description	51
5.21.2	Variable Documentation	52
5.21.2.1	assemble_flag	52
5.21.2.2	cond_all	52
5.21.2.3	deb_name	52
5.21.2.4	debug	52
5.21.2.5	dof_all	52

5.21.2.6	follower_all	53
5.21.2.7	index_kp	53
5.21.2.8	index_mb	53
5.21.2.9	init_flag	53
5.21.2.10	init_memb	53
5.21.2.11	iout	53
5.21.2.12	nemax	54
5.21.2.13	nsiz	54
5.21.2.14	nzelemmax	54
5.21.2.15	two_divide_dt	54
5.21.2.16	xyz_pt1	54
5.22	ioaero Module Reference	54
5.22.1	Detailed Description	57
5.22.2	Function/Subroutine Documentation	57
5.22.2.1	input()	57
5.22.2.2	output()	57
5.22.2.3	outputvtk()	57
5.22.3	Variable Documentation	57
5.22.3.1	aero_flag	57
5.22.3.2	aerodyn_coef	58
5.22.3.3	analysis_flag	58
5.22.3.4	arpack	58
5.22.3.5	char_len	58
5.22.3.6	coord	58
5.22.3.7	curvature	59
5.22.3.8	distr_fun	59
5.22.3.9	ech_name	59
5.22.3.10	eigen_val	59
5.22.3.11	eigen_vec_mb	59
5.22.3.12	eigen_vec_pt	60

5.22.3.13 eigenoutput . . . . .	60
5.22.3.14 ein . . . . .	60
5.22.3.15 error . . . . .	60
5.22.3.16 frame . . . . .	60
5.22.3.17 grav_flag . . . . .	60
5.22.3.18 in . . . . .	61
5.22.3.19 init . . . . .	61
5.22.3.20 init_cond . . . . .	61
5.22.3.21 init_name . . . . .	61
5.22.3.22 inp_name . . . . .	62
5.22.3.23 material . . . . .	62
5.22.3.24 mb_condition . . . . .	62
5.22.3.25 member . . . . .	62
5.22.3.26 ncond_mb . . . . .	62
5.22.3.27 ncond_pt . . . . .	63
5.22.3.28 ncurv . . . . .	63
5.22.3.29 ndistrfun . . . . .	63
5.22.3.30 ndof_el . . . . .	63
5.22.3.31 nelem . . . . .	63
5.22.3.32 nev . . . . .	64
5.22.3.33 nframe . . . . .	64
5.22.3.34 niter . . . . .	64
5.22.3.35 nkp . . . . .	64
5.22.3.36 nmate . . . . .	64
5.22.3.37 nmemb . . . . .	65
5.22.3.38 nstep . . . . .	65
5.22.3.39 ntimefun . . . . .	65
5.22.3.40 nvtk . . . . .	65
5.22.3.41 omega_a0 . . . . .	66
5.22.3.42 omega_a_tf . . . . .	66

5.22.3.43 out . . . . .	66
5.22.3.44 out_name . . . . .	66
5.22.3.45 pt_condition . . . . .	66
5.22.3.46 simu_time . . . . .	67
5.22.3.47 sol_mb . . . . .	67
5.22.3.48 sol_pt . . . . .	67
5.22.3.49 time_function . . . . .	67
5.22.3.50 v_root_a0 . . . . .	67
5.22.3.51 v_root_a_tf . . . . .	68
5.22.3.52 velocity_str . . . . .	68
5.23 member Module Reference . . . . .	68
5.23.1 Detailed Description . . . . .	68
5.23.2 Function/Subroutine Documentation . . . . .	68
5.23.2.1 assemblememberjacobian() . . . . .	69
5.23.2.2 assemblememberrhs() . . . . .	69
5.23.2.3 extractmemberproperties() . . . . .	70
5.23.3 Variable Documentation . . . . .	70
5.23.3.1 ncol_memb . . . . .	71
5.23.3.2 ndiv . . . . .	71
5.24 premodule Module Reference . . . . .	71
5.24.1 Detailed Description . . . . .	71
5.24.2 Function/Subroutine Documentation . . . . .	72
5.24.2.1 curvebeamfun() . . . . .	72
5.24.2.2 memberproperties() . . . . .	72
5.24.2.3 preprocess() . . . . .	73
5.24.2.4 rtbis() . . . . .	73
5.24.3 Variable Documentation . . . . .	74
5.24.3.1 ncol_memb . . . . .	74
5.24.3.2 ndiv . . . . .	74
5.25 prescribedcondition Module Reference . . . . .	74

5.25.1 Detailed Description . . . . .	75
5.25.2 Function/Subroutine Documentation . . . . .	75
5.25.2.1 existpi() . . . . .	75
5.25.2.2 followerj() . . . . .	76
5.25.2.3 getdistributedload() . . . . .	76
5.25.2.4 getload() . . . . .	76
5.25.2.5 getloadj() . . . . .	77
5.25.2.6 getprescribeddof() . . . . .	77
5.25.2.7 getprescribedval() . . . . .	77
5.25.2.8 initpi() . . . . .	78
5.25.2.9 initpiaero() . . . . .	78
5.25.2.10 inutechoprescribedconditions() . . . . .	78
5.25.2.11 loadintegration() . . . . .	78
5.25.2.12 transferfollower() . . . . .	79
5.25.2.13 updatefollower() . . . . .	79
5.25.2.14 updatepi() . . . . .	79
5.26 solvemumps Module Reference . . . . .	80
5.26.1 Detailed Description . . . . .	80
5.26.2 Function/Subroutine Documentation . . . . .	80
5.26.2.1 ctcabph() . . . . .	80
5.26.2.2 extractelementvalues() . . . . .	81
5.26.2.3 extractsolution() . . . . .	81
5.26.2.4 insertelementvalues() . . . . .	82
5.26.2.5 linearsolutionmumps() . . . . .	82
5.26.2.6 newtonraphsonmumps() . . . . .	84
5.26.3 Variable Documentation . . . . .	85
5.26.3.1 ierr . . . . .	85
5.26.3.2 mumps_par . . . . .	85
5.27 system Module Reference . . . . .	85
5.27.1 Detailed Description . . . . .	86

5.27.2	Function/Subroutine Documentation	86
5.27.2.1	assemblejacobian()	86
5.27.2.2	assemblerhs()	87
5.27.2.3	pointfollowerj()	88
5.27.3	Variable Documentation	88
5.27.3.1	coef	88
5.27.3.2	irn	88
5.27.3.3	jcn	89
5.27.3.4	kp_cond	89
5.27.3.5	kp_dof	89
5.27.3.6	kp_follower	89
5.27.3.7	ne	89
5.27.3.8	x_pt	90
5.28	timefunctionmodule Module Reference	90
5.28.1	Detailed Description	90
5.28.2	Function/Subroutine Documentation	90
5.28.2.1	currentvalues()	90
5.28.2.2	gettimefunction()	91
5.28.2.3	inittf()	91
5.28.2.4	inputechotimefunctions()	91

<b>6</b>	<b>Data Type Documentation</b>	<b>93</b>
6.1	gebtaero.CompositeBox.CompositeBox Class Reference	93
6.1.1	Detailed Description	93
6.1.2	Constructor & Destructor Documentation	94
6.1.2.1	__init__()	94
6.1.3	Member Function Documentation	94
6.1.3.1	ComputeMassMatrix()	94
6.1.3.2	CreateFbdFile()	94
6.1.3.3	CreateInpFile()	95
6.1.3.4	CreatePeriodicEq()	95
6.1.3.5	DisplaySectionDeformation()	95
6.1.3.6	GetHeight()	95
6.1.3.7	GetOffsets()	95
6.1.3.8	GetWidth()	96
6.1.4	Member Data Documentation	96
6.1.4.1	Down	96
6.1.4.2	Height	96
6.1.4.3	Left	96
6.1.4.4	OffsetY	96
6.1.4.5	OffsetZ	96
6.1.4.6	Right	97
6.1.4.7	Up	97
6.1.4.8	Width	97
6.2	gebtaero.CompositePlate.CompositePlate Class Reference	97
6.2.1	Detailed Description	98
6.2.2	Constructor & Destructor Documentation	98
6.2.2.1	__init__()	98
6.2.3	Member Function Documentation	98
6.2.3.1	AppendPly()	98
6.2.3.2	ComputeMassMatrix()	98

6.2.3.3	CreateFbdFile()	99
6.2.3.4	CreateInpFile()	99
6.2.3.5	CreatePeriodicEq()	99
6.2.3.6	DisplaySectionDeformation()	99
6.2.3.7	GetLayup()	100
6.2.3.8	GetOffsets()	100
6.2.3.9	GetTotThickness()	100
6.2.4	Member Data Documentation	100
6.2.4.1	Chord	100
6.2.4.2	Layup	100
6.2.4.3	Materials	101
6.2.4.4	OffsetY	101
6.2.4.5	OffsetZ	101
6.2.4.6	Orientations	101
6.2.4.7	TotThickness	101
6.3	geptaero.CompositePly.CompositePly Class Reference	101
6.3.1	Detailed Description	102
6.3.2	Constructor & Destructor Documentation	102
6.3.2.1	__init__()	102
6.3.3	Member Function Documentation	102
6.3.3.1	GetMaterial()	102
6.3.3.2	GetOrientation()	102
6.3.3.3	GetThickness()	103
6.3.4	Member Data Documentation	103
6.3.4.1	Material	103
6.3.4.2	Orientation	103
6.3.4.3	Thickness	103
6.4	geptaero.CrossSection.CrossSection Class Reference	103
6.4.1	Detailed Description	104
6.4.2	Constructor & Destructor Documentation	104



6.4.2.1	<code>__init__()</code>	104
6.4.3	Member Function Documentation	104
6.4.3.1	<code>GetFlexibilityMatrix()</code>	104
6.4.3.2	<code>GetMassMatrix()</code>	104
6.4.3.3	<code>SetFlexibilityMatrixByBox()</code>	105
6.4.3.4	<code>SetFlexibilityMatrixByIsotropicValues()</code>	105
6.4.3.5	<code>SetFlexibilityMatrixByMesh()</code>	105
6.4.3.6	<code>SetFlexibilityMatrixByPlate()</code>	106
6.4.3.7	<code>SetFlexibilityMatrixByRectBeamValues()</code>	106
6.4.3.8	<code>SetMassMatrix()</code>	106
6.4.3.9	<code>SetMassMatrixByBox()</code>	107
6.4.3.10	<code>SetMassMatrixByMesh()</code>	107
6.4.3.11	<code>SetMassMatrixByPlate()</code>	107
6.4.3.12	<code>SetMassMatrixByRectBeamValues()</code>	107
6.4.4	Member Data Documentation	107
6.4.4.1	<code>ElasticCenter</code>	108
6.4.4.2	<code>FlexibilityMatrix</code>	108
6.4.4.3	<code>MassMatrix</code>	108
6.5	<code>prescribedcondition::distriload</code> Type Reference	108
6.5.1	Detailed Description	108
6.5.2	Member Data Documentation	109
6.5.2.1	<code>distr_fun</code>	109
6.5.2.2	<code>follower</code>	109
6.5.2.3	<code>value</code>	109
6.6	<code>gebtaero.ExternalMesh.ExternalMesh</code> Class Reference	109
6.6.1	Detailed Description	110
6.6.2	Constructor & Destructor Documentation	110
6.6.2.1	<code>__init__()</code>	110
6.6.3	Member Function Documentation	110
6.6.3.1	<code>AppendComponent()</code>	111

6.6.3.2	ComputeElementSurfAndCG()	111
6.6.3.3	ComputeMassMatrixFromMesh()	111
6.6.3.4	CreateInpFile()	111
6.6.3.5	CreatePeriodicEq()	112
6.6.3.6	DisplaySectionDeformation()	112
6.6.3.7	GetMeshFile()	112
6.6.4	Member Data Documentation	112
6.6.4.1	Chord	112
6.6.4.2	Components	112
6.6.4.3	elements	113
6.6.4.4	Lx	113
6.6.4.5	Materials	113
6.6.4.6	MeshFile	113
6.6.4.7	ncurv	113
6.6.4.8	nodes	113
6.6.4.9	nstrain	114
6.6.4.10	OffsetY	114
6.6.4.11	OffsetZ	114
6.6.4.12	Orientations	114
6.6.4.13	TotThickness	114
6.6.4.14	x0	114
6.7	gebtaero.Frame.Frame Class Reference	115
6.7.1	Detailed Description	115
6.7.2	Constructor & Destructor Documentation	115
6.7.2.1	__init__()	115
6.7.3	Member Function Documentation	115
6.7.3.1	GetAxis()	115
6.7.3.2	GetFrameMatrix()	116
6.7.3.3	GetTwist()	116
6.7.4	Member Data Documentation	116

6.7.4.1	Axis	116
6.7.4.2	FrameMatrix	116
6.7.4.3	Twist	116
6.8	geptaero.GebtPlot.GebtPlot Class Reference	117
6.8.1	Detailed Description	117
6.8.2	Member Function Documentation	117
6.8.2.1	EigenFreqDamping()	117
6.8.2.2	EigenFreqDampingUnsorted()	117
6.8.2.3	ParaviewOutput()	118
6.8.2.4	WriteParaviewScript()	118
6.9	geptaero.InputFile.InputFile Class Reference	118
6.9.1	Detailed Description	119
6.9.2	Constructor & Destructor Documentation	119
6.9.2.1	__init__()	119
6.9.3	Member Function Documentation	120
6.9.3.1	AppendTimeFunction()	120
6.9.3.2	GetAnalysisFlag()	120
6.9.3.3	GetFileName()	120
6.9.3.4	GetName()	120
6.9.3.5	GetTimeFunction()	121
6.9.3.6	RemoveInputFile()	121
6.9.3.7	WriteInitFile()	121
6.9.3.8	WriteInputFile()	121
6.9.4	Member Data Documentation	121
6.9.4.1	ACOmegaa	121
6.9.4.2	ACOmegaaTFNumber	122
6.9.4.3	ACVa	122
6.9.4.4	ACVaTFNumber	122
6.9.4.5	AeroFlag	122
6.9.4.6	AlphaAC	122

6.9.4.7	AnalysisFlag	122
6.9.4.8	BetaAC	123
6.9.4.9	FileName	123
6.9.4.10	GravFlag	123
6.9.4.11	Name	123
6.9.4.12	Nev	123
6.9.4.13	Niter	123
6.9.4.14	Nstep	124
6.9.4.15	Nvtk	124
6.9.4.16	Rho	124
6.9.4.17	SimuEnd	124
6.9.4.18	SimuStart	124
6.9.4.19	TimeFunctions	124
6.9.4.20	Vinf	125
6.9.4.21	Wing	125
6.9.4.22	Xcg	125
6.10	gebtaero.IsoMaterial.IsoMaterial Class Reference	125
6.10.1	Detailed Description	125
6.10.2	Constructor & Destructor Documentation	126
6.10.2.1	__init__()	126
6.10.3	Member Function Documentation	126
6.10.3.1	GetDensity()	126
6.10.3.2	GetIso()	126
6.10.4	Member Data Documentation	126
6.10.4.1	E	126
6.10.4.2	Nu	127
6.10.4.3	Rho	127
6.11	internaldata::memberinf Type Reference	127
6.11.1	Detailed Description	127
6.11.2	Member Data Documentation	128

6.11.2.1	<a href="#">aerodyn_coef</a>	128
6.11.2.2	<a href="#">coordinate</a>	128
6.11.2.3	<a href="#">dl</a>	128
6.11.2.4	<a href="#">le</a>	128
6.11.2.5	<a href="#">mate</a>	128
6.11.2.6	<a href="#">ncol_memb</a>	129
6.11.2.7	<a href="#">ndiv</a>	129
6.11.2.8	<a href="#">triad</a>	129
6.12	<a href="#">gebtaero.OrthoMaterial.OrthoMaterial Class Reference</a>	129
6.12.1	<a href="#">Detailed Description</a>	130
6.12.2	<a href="#">Constructor &amp; Destructor Documentation</a>	130
6.12.2.1	<a href="#">__init__()</a>	130
6.12.3	<a href="#">Member Function Documentation</a>	130
6.12.3.1	<a href="#">GetDensity()</a>	130
6.12.3.2	<a href="#">GetOrtho()</a>	130
6.12.4	<a href="#">Member Data Documentation</a>	130
6.12.4.1	<a href="#">EI</a>	131
6.12.4.2	<a href="#">Et</a>	131
6.12.4.3	<a href="#">Glt</a>	131
6.12.4.4	<a href="#">Nult</a>	131
6.12.4.5	<a href="#">Rho</a>	131
6.13	<a href="#">prescribedcondition::prescriinf Type Reference</a>	131
6.13.1	<a href="#">Detailed Description</a>	132
6.13.2	<a href="#">Member Data Documentation</a>	132
6.13.2.1	<a href="#">dof</a>	132
6.13.2.2	<a href="#">follower</a>	132
6.13.2.3	<a href="#">id</a>	133
6.13.2.4	<a href="#">time_fun_no</a>	133
6.13.2.5	<a href="#">value</a>	133
6.13.2.6	<a href="#">value_current</a>	133

6.14	gebtaero.Simulation.Simulation Class Reference	134
6.14.1	Detailed Description	134
6.14.2	Constructor & Destructor Documentation	134
6.14.2.1	__init__()	135
6.14.3	Member Function Documentation	135
6.14.3.1	DeformedModalFlutterSpeed()	135
6.14.3.2	DeformedModalFlutterSpeedSorted()	135
6.14.3.3	EigenTab()	136
6.14.3.4	EigenTabSorted()	136
6.14.3.5	Eigenvalues()	136
6.14.3.6	EquilibriumAoA()	137
6.14.3.7	FlutterVtk()	137
6.14.3.8	GetWing()	137
6.14.3.9	ModalCriticalSpeed()	138
6.14.3.10	ModalDivergenceSpeed()	138
6.14.3.11	ModalDivergenceSpeedSorted()	138
6.14.3.12	ModalFlutterSpeed()	139
6.14.3.13	ModalFlutterSpeedSorted()	139
6.14.3.14	StaticLoads()	139
6.14.3.15	TemporalDivergenceSpeed()	140
6.14.3.16	TemporalDynamic()	140
6.14.3.17	TemporalFlutterSpeed()	140
6.14.4	Member Data Documentation	141
6.14.4.1	Input	141
6.14.4.2	Wing	141
6.15	timefunctionmodule::timefunction Type Reference	141
6.15.1	Detailed Description	142
6.15.2	Member Data Documentation	142
6.15.2.1	entries	142
6.15.2.2	fun_type	142

6.15.2.3	<a href="#">fun_val</a>	142
6.15.2.4	<a href="#">phase_val</a>	142
6.15.2.5	<a href="#">te</a>	143
6.15.2.6	<a href="#">time_val</a>	143
6.15.2.7	<a href="#">ts</a>	143
6.16	<a href="#">gebtaero.TimeFunction.TimeFunction Class Reference</a>	143
6.16.1	<a href="#">Detailed Description</a>	144
6.16.2	<a href="#">Constructor &amp; Destructor Documentation</a>	144
6.16.2.1	<a href="#">__init__()</a>	144
6.16.3	<a href="#">Member Function Documentation</a>	144
6.16.3.1	<a href="#">AppendFunctionEntrieHarmonic()</a>	144
6.16.3.2	<a href="#">AppendFunctionEntriePieceWise()</a>	144
6.16.3.3	<a href="#">GetFunctionEnd()</a>	145
6.16.3.4	<a href="#">GetFunctionEntrie()</a>	145
6.16.3.5	<a href="#">GetFunctionEntries()</a>	145
6.16.3.6	<a href="#">GetFunctionStart()</a>	145
6.16.3.7	<a href="#">GetFunctionType()</a>	145
6.16.4	<a href="#">Member Data Documentation</a>	145
6.16.4.1	<a href="#">FunctionEnd</a>	146
6.16.4.2	<a href="#">FunctionEntries</a>	146
6.16.4.3	<a href="#">FunctionStart</a>	146
6.16.4.4	<a href="#">FunctionType</a>	146
6.17	<a href="#">gebtaero.Wing.Wing Class Reference</a>	146
6.17.1	<a href="#">Detailed Description</a>	147
6.17.2	<a href="#">Constructor &amp; Destructor Documentation</a>	147
6.17.2.1	<a href="#">__init__()</a>	147
6.17.3	<a href="#">Member Function Documentation</a>	147
6.17.3.1	<a href="#">AppendWingSection()</a>	148
6.17.3.2	<a href="#">GetCrossSections()</a>	148
6.17.3.3	<a href="#">GetFrames()</a>	148

6.17.3.4	GetKpList()	148
6.17.3.5	GetName()	148
6.17.3.6	GetSurface()	149
6.17.3.7	GetWeight()	149
6.17.3.8	GetWingRootPosition()	149
6.17.3.9	GetWingSections()	149
6.17.4	Member Data Documentation	149
6.17.4.1	CrossSections	149
6.17.4.2	Frames	150
6.17.4.3	KpList	150
6.17.4.4	Name	150
6.17.4.5	WingRootPosition	150
6.17.4.6	WingSections	150
6.18	geptaero.WingSection.WingSection Class Reference	151
6.18.1	Detailed Description	151
6.18.2	Constructor & Destructor Documentation	152
6.18.2.1	__init__()	152
6.18.3	Member Function Documentation	152
6.18.3.1	GetChord()	152
6.18.3.2	GetCrossSection()	152
6.18.3.3	GetFrame()	152
6.18.3.4	GetHalfChord()	153
6.18.3.5	GetNumberOfElements()	153
6.18.3.6	GetParameterA()	153
6.18.3.7	GetSectionLength()	153
6.18.3.8	SetChord()	153
6.18.3.9	SetNumberOfElements()	154
6.18.3.10	SetParameterA()	154
6.18.3.11	SetSectionLength()	154
6.18.4	Member Data Documentation	154
6.18.4.1	Chord	154
6.18.4.2	CrossSection	154
6.18.4.3	Frame	155
6.18.4.4	HalfChord	155
6.18.4.5	NumberOfElements	155
6.18.4.6	ParameterA	155
6.18.4.7	SectionLength	155
6.19	globaldatafun::writevec Interface Reference	155
6.19.1	Detailed Description	156
6.19.2	Member Function/Subroutine Documentation	156
6.19.2.1	writeintvector()	156
6.19.2.2	writerealvector()	156



<b>7 File Documentation</b>	<b>157</b>
7.1 /home/bertrand/logiciels/gebtaero_frama/README.md File Reference	157
7.2 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/__init__.py File Reference	157
7.3 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/CompositeBox.py File Reference	157
7.4 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/CompositePlate.py File Reference	157
7.5 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/CompositePly.py File Reference	158
7.6 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/CrossSection.py File Reference	158
7.7 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/ExternalMesh.py File Reference	158
7.8 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/Frame.py File Reference	158
7.9 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/GebtPlot.py File Reference	159
7.10 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/InputFile.py File Reference	159
7.11 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/IsoMaterial.py File Reference	159
7.12 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/OrthoMaterial.py File Reference	159
7.13 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/Simulation.py File Reference	160
7.14 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/TimeFunction.py File Reference	160
7.15 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/utils.py File Reference	160
7.16 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/Wing.py File Reference	161
7.17 /home/bertrand/these/logiciels/programme/interface/src/gebtaero/WingSection.py File Reference	161
7.18 /home/bertrand/these/logiciels/programme/src/Analysis.f90 File Reference	161
7.18.1 Function/Subroutine Documentation	161
7.18.1.1 analysis()	162
7.19 /home/bertrand/these/logiciels/programme/src/CPUtime.f90 File Reference	163
7.20 /home/bertrand/these/logiciels/programme/src/EigenSolveMumps.f90 File Reference	164
7.21 /home/bertrand/these/logiciels/programme/src/Element.f90 File Reference	164
7.22 /home/bertrand/these/logiciels/programme/src/GlobalDataFun.f90 File Reference	166
7.23 /home/bertrand/these/logiciels/programme/src/InternalData.f90 File Reference	169
7.24 /home/bertrand/these/logiciels/programme/src/IOaero.f90 File Reference	170
7.25 /home/bertrand/these/logiciels/programme/src/mainAero.f90 File Reference	172
7.25.1 Function/Subroutine Documentation	172
7.25.1.1 gebt()	172
7.26 /home/bertrand/these/logiciels/programme/src/Member.f90 File Reference	172
7.27 /home/bertrand/these/logiciels/programme/src/Preprocess.f90 File Reference	173
7.28 /home/bertrand/these/logiciels/programme/src/PrescribedCondition.f90 File Reference	174
7.29 /home/bertrand/these/logiciels/programme/src/SolveMumps.f90 File Reference	175
7.29.1 Function/Subroutine Documentation	175
7.29.1.1 linesearch()	175
7.30 /home/bertrand/these/logiciels/programme/src/System.f90 File Reference	176
7.30.1 Function/Subroutine Documentation	176
7.30.1.1 assemblepointj()	176
7.30.1.2 assemblepointrhs()	177
7.31 /home/bertrand/these/logiciels/programme/src/TimeFunction.f90 File Reference	177



# 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 detailed 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:

<a href="#">cputime</a>	A module use to calculate the computation time . . . . .	9
<a href="#">eigenmumps</a>	This module contains the main routines needed for eigen value analysis and allow to call Arpack and MUMPS library . . . . .	10
<a href="#">element</a>	This module contains information and calculation for an element within a member . . . . .	13
<a href="#">gebtaero</a>	. . . . .	30
<a href="#">gebtaero.CompositeBox</a>	. . . . .	30
<a href="#">gebtaero.CompositePlate</a>	. . . . .	30
<a href="#">gebtaero.CompositePly</a>	. . . . .	30
<a href="#">gebtaero.CrossSection</a>	. . . . .	31
<a href="#">gebtaero.ExternalMesh</a>	. . . . .	31
<a href="#">gebtaero.Frame</a>	. . . . .	31
<a href="#">gebtaero.GebtPlot</a>	. . . . .	31
<a href="#">gebtaero.InputFile</a>	. . . . .	31
<a href="#">gebtaero.IsoMaterial</a>	. . . . .	31
<a href="#">gebtaero.OrthoMaterial</a>	. . . . .	31
<a href="#">gebtaero.Simulation</a>	. . . . .	32
<a href="#">gebtaero.TimeFunction</a>	. . . . .	32
<a href="#">gebtaero.utils</a>	. . . . .	32
<a href="#">gebtaero.Wing</a>	. . . . .	35
<a href="#">gebtaero.WingSection</a>	. . . . .	36
<a href="#">globaldatafun</a>	This module contains general-purpose global constants,I/O functions/subroutines and math functions/subroutines . . . . .	36
<a href="#">internaldata</a>	This module contains the variables needed internally in the program. Not necessary to be defined in the outside environment . . . . .	51
<a href="#">ioaero</a>	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 . . . . .	54
<a href="#">member</a>	This module assembles within a member without considering the particular conditions of the end points . . . . .	68

<a href="#">prepromodule</a>	
This module preprocess the finite element model including connectivity and member information.	
This information are time step independent . . . . .	71
<a href="#">prescribedcondition</a>	
A module for defining prescribed conditions including both concentrated information and distributed information . . . . .	74
<a href="#">solvemumps</a>	
This module contains the linear & nonlinear solver interfaced with MUMPS direct solver library	80
<a href="#">system</a>	
This module assembles the system including the coefficient matrix (jacobian matrix) and the right hand side (negative of the equation values) * . . . . .	85
<a href="#">timefunctionmodule</a>	
A module for defining time functions needed for both prescribed concentrated and distributed conditions . . . . .	90

## Chapter 3

# Data Type Index

### 3.1 Data Types List

Here are the data types with brief descriptions:

<a href="#">gebtaero.CompositeBox.CompositeBox</a>	
Class interfacing the solver with 3D FEM calculix computation to obtain the cross section parameter from a composite box . . . . .	93
<a href="#">gebtaero.CompositePlate.CompositePlate</a>	97
<a href="#">gebtaero.CompositePly.CompositePly</a>	101
<a href="#">gebtaero.CrossSection.CrossSection</a>	103
<a href="#">prescribedcondition::distriload</a>	
Define the distributed load condition . . . . .	108
<a href="#">gebtaero.ExternalMesh.ExternalMesh</a>	109
<a href="#">gebtaero.Frame.Frame</a>	115
<a href="#">gebtaero.GebtPlot.GebtPlot</a>	117
<a href="#">gebtaero.InputFile.InputFile</a>	118
<a href="#">gebtaero.IsoMaterial.IsoMaterial</a>	125
<a href="#">internaldata::memberinf</a>	
Structure containing the characteristics of a finite element . . . . .	127
<a href="#">gebtaero.OrthoMaterial.OrthoMaterial</a>	129
<a href="#">prescribedcondition::prescriinf</a>	
Define the prescribed condition . . . . .	131
<a href="#">gebtaero.Simulation.Simulation</a>	134
<a href="#">timefunctionmodule::timefunction</a>	141
<a href="#">gebtaero.TimeFunction.TimeFunction</a>	143
<a href="#">gebtaero.Wing.Wing</a>	146
<a href="#">gebtaero.WingSection.WingSection</a>	151
<a href="#">globaldatafun::writevec</a>	155





## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

/home/bertrand/these/logiciels/programme/interface/src/gebtaero/___init___py . . . . .	157
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/CompositeBox.py . . . . .	157
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/CompositePlate.py . . . . .	157
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/CompositePly.py . . . . .	158
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/CrossSection.py . . . . .	158
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/ExternalMesh.py . . . . .	158
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/Frame.py . . . . .	158
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/GebtPlot.py . . . . .	159
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/InputFile.py . . . . .	159
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/IsoMaterial.py . . . . .	159
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/OrthoMaterial.py . . . . .	159
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/Simulation.py . . . . .	160
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/TimeFunction.py . . . . .	160
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/Utils.py . . . . .	160
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/Wing.py . . . . .	161
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/WingSection.py . . . . .	161
/home/bertrand/these/logiciels/programme/src/Analysis.f90 . . . . .	161
/home/bertrand/these/logiciels/programme/src/CPUtime.f90 . . . . .	163
/home/bertrand/these/logiciels/programme/src/EigenSolveMumps.f90 . . . . .	164
/home/bertrand/these/logiciels/programme/src/Element.f90 . . . . .	164
/home/bertrand/these/logiciels/programme/src/GlobalDataFun.f90 . . . . .	166
/home/bertrand/these/logiciels/programme/src/InternalData.f90 . . . . .	169
/home/bertrand/these/logiciels/programme/src/IOaero.f90 . . . . .	170
/home/bertrand/these/logiciels/programme/src/mainAero.f90 . . . . .	172
/home/bertrand/these/logiciels/programme/src/Member.f90 . . . . .	172
/home/bertrand/these/logiciels/programme/src/Preprocess.f90 . . . . .	173
/home/bertrand/these/logiciels/programme/src/PrescribedCondition.f90 . . . . .	174
/home/bertrand/these/logiciels/programme/src/SolveMumps.f90 . . . . .	175
/home/bertrand/these/logiciels/programme/src/System.f90 . . . . .	176
/home/bertrand/these/logiciels/programme/src/TimeFunction.f90 . . . . .	177



## 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 [eigenmumps](#) (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	<i>nev</i>	<a href="#">ioaero::nev</a>
in	<i>ncv</i>	size of the subdomain used by the IRAM algorithm (doc Arpack)
out	<i>error</i>	<a href="#">ioaero::error</a>
out	<i>er</i>	Real part of the eigenvalue (before problem inversion)
out	<i>ei</i>	Imaginary part of the eigenvalue (before problem inversion)
out	<i>vector</i>	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	<i>nzm</i>	number of nonzero coefficient
in	<i>irnM</i>	line index of nonzero values
in	<i>jcnM</i>	column index of nonzero values
in	<i>coef_mass</i>	mass matrix sparse coefficient
in	<i>w</i>	RHS vector
out	<i>u</i>	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	<i>ndof_el</i>	<a href="#">ioaero::ndof_el</a>
in	<i>aero_flag</i>	<a href="#">ioaero::aero_flag</a>
in	<i>grav_flag</i>	<a href="#">ioaero::grav_flag</a>
in	<i>memb_info</i>	array containing the characteristics of the beam members
in	<i>v_root_a</i>	linear velocity of frame a
in	<i>omega_a</i>	angular velocity of frame a
in	<i>distr_fun</i>	<a href="#">ioaero::distr_fun</a>
in	<i>member</i>	<a href="#">ioaero::member</a>
in	<i>niter</i>	<a href="#">ioaero::niter</a>
in	<i>ncond_mb</i>	<a href="#">ioaero::ncond_mb</a>
in	<i>pt_condition</i>	<a href="#">ioaero::pt_condition</a>
in	<i>mb_condition</i>	<a href="#">ioaero::mb_condition</a>
	<i>dof_con</i>	the connecting condition for key point.
out	<i>error</i>	<a href="#">ioaero::error</a>
in	<i>x</i>	the solution vector of the linear system (steady-state)
in, out	<i>nev</i>	<a href="#">ioaero::nev</a>
out	<i>eigen_val</i>	<a href="#">ioaero::eigen_val</a>
out	<i>eigen_vec</i>	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)  
*Calculate the Jacobian matrix for each element.*
- subroutine, public [elemmass](#) (ndof\_el, elemM)  
*Calculate 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 element*
- 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) [ecabhalfI](#)  
*eCab\*dL/2*
- real(dbl), dimension(ndim, ndim) [ectcabhalfI](#)  
*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) [jalphadot\\_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	<i>ndof</i> ↔ _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 )
```

Calculate the Jacobian matrix for each element.

##### Parameters

in	<i>ndof</i> ↔ _el	ioaero::ndof_el
in	<i>niter</i>	ioaero::niter
out	<i>elemjac</i>	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 )
```

Calculate the mass matrix for each element.

## Parameters

in	<i>ndof_el</i>	<a href="#">ioaero::ndof_el</a>
out	<i>elemm</i>	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	<i>elem_no</i>	Element index
in	<i>memb_info_i</i>	the paramater of the members (see memberinf Type)
in	<i>x_elem</i>	the 12 variables of the element $u_i$ , $F_i$ , $M_i$ , for dynamic analysis, 6 more $P_i$ , $H_i$ . for initial step, $x\_elem$ contains, $CTCabPdot$ , $CTCabHdot$ , $F_i$ , $M_i$ , $P_i$ , $H_i$ . Finally for Peters aero the $N_s$ induced flow states
in	<i>v_root_a</i>	linear velocity of the root point in inertial frame
in	<i>omega_a</i>	angular velocity of the root point in inertial frame
in, out	<i>init_elem</i>	initial step: the 12 initial values of the element $u_i$ , $\theta_i$ , $\dot{u}_i$ , $\dot{\theta}_i$ . Time marching: $2/dtui + \dot{u}_i$ , $2/dt\theta_i + \dot{\theta}_i$ , $2/dtCTCabP + \dot{CTCabP}$
in	<i>ndof_el</i>	<a href="#">#ioaero::ndof_el</a>
in	<i>aero_flag</i>	<a href="#">ioaero::aero_flag</a>
in	<i>grav_flag</i>	<a href="#">ioaero::grav_flag</a>

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::elgammad [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 ecabhalf1**

```
real(dbl), dimension(ndim,ndim) element::ecabhalf1 [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 ectcabhalf1

```
real(dbl), dimension(ndim,ndim) element::ectcabhalf1 [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 geptaero.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 theta \*.*
- 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/postprocessor (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	<i>theta</i>	Rodrigues rotation parameters
in	<i>ect</i>	Direction Cosine matrix between frame b and B
out	<i>ekttek</i>	=OuterProduct(ek,theta)+OuterProduct(theta,ek)
out	<i>ecttheta</i>	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  $e\dot{C}^T \cdot x$  derivative w.r.t  $\dot{\theta}$  \*.

## Parameters

in	<i>theta</i>	Rodrigues rotation parameters
in	<i>ect</i>	Direction Cosine matrix between frame b and B
in	<i>x</i>	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_rl,
    integer, intent(in) str_cl )
```

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	<i>file_unit</i>	File Unit (see fortran IO doc)
in	<i>sta_type</i>	status type
in	<i>rw_type</i>	rewrite configuration
out	<i>error</i>	<a href="#">ioaero::error</a>

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	<i>matrix<sub>in</sub></i>	the matrix to be inverted
out	<i>matrix</i>	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	<i>message</i>	a character variable to hold error message
out	<i>error</i>	<a href="#">ioaero::error</a>

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	<i>vari_name</i>	a character variable to hold variable name
out	<i>error</i>	<a href="#">ioaero::error</a>

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	<i>n</i>	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	<i>ein</i>	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 occurred; 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/postprocessor (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 characteristics 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 analysis; 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

<i>grav_flag</i>	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 modes shapes.

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

<i>nkp</i>	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 assembles 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 assembles within a member without considering the particular conditions of the end points.

#### 5.23.2 Function/Subroutine Documentation

5.23.2.1 `assemblermemberjacobian()`

```

subroutine, public member::assemblermemberjacobian (
    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	<i>ndof_el</i>	<a href="#">ioaero::ndof_el</a>
in	<i>niter</i>	<a href="#">ioaero::niter</a>
in	<i>aero_flag</i>	<a href="#">ioaero::aero_flag</a>
in	<i>grav_flag</i>	<a href="#">ioaero::grav_flag</a>
in	<i>memb_info_i</i>	array containing the characteristics of a beam member
in	<i>v_root_a</i>	linear velocity of frame a
in	<i>omega_a</i>	angular velocity of frame a
in	<i>x_memb</i>	solution vector of a beam member
out	<i>nz_memb</i>	Number of nonzero value of the member Jacobian
out	<i>irn_memb</i>	line index of the nonzero coefficient
out	<i>jcn_memb</i>	column index of the nonzero coefficient
out	<i>coef_memb</i>	value of the nonzero coefficient

Definition at line 82 of file Member.f90.

5.23.2.2 `assemblermemberrhs()`

```

subroutine, public member::assemblermemberrhs (
    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	<i>ndof_el</i>	<a href="#">ioaero::ndof_el</a>
in	<i>aero_flag</i>	<a href="#">ioaero::aero_flag</a>
in	<i>grav_flag</i>	<a href="#">ioaero::grav_flag</a>
in	<i>memb_info_i</i>	array containing the characteristics of a beam member
in	<i>v_root_a</i>	linear velocity of frame a
in	<i>omega_a</i>	angular velocity of frame a
in	<i>x_memb</i>	solution vector of a beam member
out	<i>rhs_memb</i>	RHS vector of a beam member
in	<i>init_cond</i>	<a href="#">ioaero::init_cond</a>

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	<i>memb_no</i>	Number of the current member
in	<i>ncond_mb</i>	<a href="#">ioaero::ncond_mb</a>
in	<i>member</i>	<a href="#">ioaero::member</a>
in	<i>memb_info_i</i>	array containing the characteristics of a beam member
in	<i>distr_fun</i>	<a href="#">ioaero::distr_fun</a>
in	<i>mb_condition</i>	<a href="#">ioaero::mb_condition</a>
out	<i>error</i>	<a href="#">ioaero::error</a>
in	<i>init_cond</i>	<a href="#">ioaero::init_cond</a>

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 premodule Module Reference

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

## 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 bisection 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 independent.

## 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) kL2,
    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	<i>memb_no</i>	member index
in	<i>ndof_el</i>	<a href="#">ioaero::ndof_el</a>
in	<i>member</i>	<a href="#">ioaero::member</a>
in	<i>aero_flag</i>	<a href="#">ioaero::aero_flag</a>
in	<i>grav_flag</i>	<a href="#">ioaero::grav_flag</a>
in	<i>material</i>	<a href="#">ioaero::material</a>
in	<i>frame</i>	<a href="#">ioaero::frame</a>
in	<i>coord</i>	<a href="#">ioaero::coord</a>
in	<i>curvature</i>	<a href="#">ioaero::curvature</a>
in	<i>aerodyn_coef</i>	<a href="#">ioaero::aerodyn_coef</a>
out	<i>memb_info_i</i>	the characteristics of the ith beam member
out	<i>error</i>	<a href="#">ioaero::error</a>

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	<i>nkp</i>	<a href="#">ioaero::nkp</a>
in	<i>nelem</i>	<a href="#">ioaero::nelem</a>
in	<i>aero_flag</i>	<a href="#">ioaero::aero_flag</a>
in	<i>grav_flag</i>	<a href="#">ioaero::grav_flag</a>
in	<i>ndof_el</i>	<a href="#">ioaero::ndof_el</a>
in	<i>member</i>	<a href="#">ioaero::member</a>
in, out	<i>material</i>	<a href="#">ioaero::material</a>
in	<i>frame</i>	<a href="#">ioaero::frame</a>
in	<i>coord</i>	<a href="#">ioaero::coord</a>
in	<i>curvature</i>	<a href="#">ioaero::curvature</a>
in	<i>aerodyn_coef</i>	<a href="#">ioaero::aerodyn_coef</a>
out	<i>dof_con</i>	the connecting condition for key point.
out	<i>memb_info</i>	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([distrload](#)) 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  $J = \text{diff} C^T \cdot \text{vec} / \text{diff} \theta$ , return a 3x3 matrix with ith column corresponding to the derivative with respect to  $\theta_i$ .*
- real(dbl) function [loadintegration](#) (flag, dL, Le, func)  
*Calculate the load using Chebychev polynomials \*.*
- real(dbl) function, dimension(size(vec)) [transferfollower](#) (follower, vec, CT)  
*Transfer follower according to  $C^T \cdot \text{vec}$  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  $J = \text{diff}C^T.\text{vec}/\text{diff}\theta$ , return a 3x3 matrix with ith column corresponding to the derivative with respect to  $\theta_i$ .

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	<i>memb_no</i>	index of the beam member
in	<i>distr_fun</i>	<a href="#">ioaero::distr_fun</a>
in	<i>mb_condition</i>	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	<i>flag</i>	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	<i>nkp</i>	<a href="#">ioaero::nkp</a>
----	------------	-----------------------------

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	<i>nkp</i>	<a href="#">ioaero::nkp</a>
in	<i>pt_condition</i>	<a href="#">ioaero::pt_condition</a>

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]

Calculate the load using Chebychev polynomials \*.

#### Parameters

in	<i>flag</i>	if flag=-1, it is for the starting point, if flag=1, it is for the ending point
in	<i>dL</i>	length of the element
in	<i>le</i>	the ending point of the element at the arc length of the member
in	<i>func</i>	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  $C^T \cdot \text{vec}$  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	<i>niter</i>	<a href="#">ioaero::niter</a>
in	<i>member</i>	<a href="#">ioaero::member</a>
in	<i>sol_↔ mb_i</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	<i>ndof_el</i>	<a href="#">ioaero::ndof_el</a>
in	<i>member</i>	<a href="#">ioaero::member</a>
in	<i>x</i>	the solution vector
out	<i>sol_↔ mb_i</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	<i>ndof_el</i>	<a href="#">ioaero::ndof_el</a>
in	<i>member</i>	<a href="#">ioaero::member</a>

## Parameters

in	<i>coord</i>	<a href="#">ioaero::coord</a>
in	<i>memb_info</i>	contains the member parameters of the whole structure
in	<i>x</i>	the solution vector
in	<i>dof_con</i>	the connecting condition for key point.
out	<i>sol_pt_i</i>	solutions for points for ith step
out	<i>sol_mb_i</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	<i>ndof_el</i>	<a href="#">ioaero::ndof_el</a>
in	<i>member</i>	<a href="#">#ioaero::memeber</a>
in, out	<i>x</i>	the solution vector
in	<i>init_cond</i>	<a href="#">ioaero::init_cond</a>

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	<i>ndof_el</i>	<a href="#">ioaero::ndof_el</a>
in	<i>aero_flag</i>	<a href="#">ioaero::aero_flag</a>
in	<i>grav_flag</i>	<a href="#">ioaero::grav_flag</a>
in	<i>v_root_a</i>	linear velocity of frame a
in	<i>omega_a</i>	angular velocity of frame a
in	<i>distr_fun</i>	<a href="#">ioaero::distr_fun</a>
in	<i>member</i>	<a href="#">ioaero::member</a>
in	<i>ncond_mb</i>	<a href="#">ioaero::ncond_mb</a>
in	<i>mb_condition</i>	<a href="#">ioaero::mb_condition</a>
	<i>dof_con</i>	this array is passed by value
out	<i>error</i>	<a href="#">ioaero::error</a>
out	<i>x</i>	The solution vector
in	<i>init_cond</i>	<a href="#">ioaero::init_cond</a>

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	<i>ndof_el</i>	<a href="#">ioaero::ndof_el</a>
in	<i>aero_flag</i>	<a href="#">ioaero::aero_flag</a>
in	<i>grav_flag</i>	<a href="#">ioaero::grav_flag</a>
in	<i>memb_info</i>	contains the member parameters of the whole structure
in	<i>v_root_a</i>	linear velocity of frame a
in	<i>omega_a</i>	angular velocity of frame a
in	<i>distr_fun</i>	<a href="#">ioaero::distr_fun</a>
in	<i>member</i>	<a href="#">ioaero::member</a>
in	<i>niter</i>	<a href="#">ioaero::niter</a>

## Parameters

in	<i>ncond_mb</i>	<a href="#">ioaero::ncond_mb</a>
in	<i>mb_condition</i>	<a href="#">ioaero::mb_condition</a>
in, out	<i>x</i>	The solution vector
out	<i>error</i>	<a href="#">ioaero::error</a>
in	<i>init_cond</i>	<a href="#">ioaero::init_cond</a>

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

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	<i>ndof_el</i>	<a href="#">ioaero::ndof_el</a>
in	<i>aero_flag</i>	<a href="#">ioaero::aero_flag</a>
in	<i>grav_flag</i>	<a href="#">ioaero::grav_flag</a>
in	<i>memb_info</i>	contains the member parameters of the whole structure
in	<i>v_root_a</i>	linear velocity of frame a
in	<i>omega_a</i>	angular velocity of frame a

## Parameters

in	<i>distr_fun</i>	<a href="#">ioaero::distr_fun</a>
in	<i>member</i>	<a href="#">ioaero::member</a>
in	<i>ncond_mb</i>	<a href="#">ioaero::ncond_mb</a>
in	<i>x</i>	solution vector
in	<i>mb_condition</i>	<a href="#">ioaero::mb_condition</a>
in	<i>init_cond</i>	<a href="#">ioaero::init_cond</a>

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**

<code>in</code>	<code>ein</code>	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`, `PlaneSection`=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, RigidElg3=False)
- def [SetFlexibilityMatrixByIsotropicValues](#) (self, Elg2, Elg3, 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,
    RigidEIq3 = 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, DampScale=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 plotting 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](#), [ACOmegaaTFNumber](#), [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,
    ACOmega,
    ACOmegaTFNumber,
    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 ACOmegaTFNumber

```
gebtaero.InputFile.InputFile.ACOmegaTFNumber
```

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 characteristics 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): flexibility 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 characteristics 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): flexibility 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, [EI](#), [Et](#), [Nult](#), [Glt](#), [Rho](#)=None)
- def [GetOrtho](#) (self)
- def [GetDensity](#) (self)

### Public Attributes

- [EI](#)
- [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, KsiObj=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, NormLim=1., KsiObj=1e-6, GravFlag=0, verbosity=0)
- def [ModalFlutterSpeedSorted](#) (self, Rho, Vmax, DeltaV, AeroFlag, ModesToCompute, ModesToPlot, AlphaAC, 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, ModesToPlot, 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, CoefPerturb, 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 local 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	<i>file_unit</i>	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	<i>file_unit</i>	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	<i>nkp</i>	<a href="#">ioaero::nkp</a>
in	<i>nelem</i>	<a href="#">ioaero::nelem</a>
in	<i>ndof_el</i>	<a href="#">ioaero::ndof_el</a>
in	<i>nmemb</i>	<a href="#">ioaero::nmemb</a>
in	<i>ncond_pt</i>	<a href="#">ioaero::ncond_pt</a>
in	<i>nmate</i>	<a href="#">ioaero::nmate</a>
in	<i>nframe</i>	<a href="#">ioaero::nframe</a>
in	<i>ndistrfun</i>	<a href="#">ioaero::ndistrfun</a>
in	<i>ncurv</i>	<a href="#">ioaero::ncurv</a>
in	<i>coord</i>	<a href="#">ioaero::coord</a>

## Parameters

in	<i>member</i>	<a href="#">ioaero::member</a>
in, out	<i>pt_condition</i>	<a href="#">ioaero::pt_condition</a>
in, out	<i>material</i>	<a href="#">ioaero::material</a>
in	<i>aerodyn_coef</i>	<a href="#">ioaero::aerodyn_coef</a>
in	<i>niter</i>	<a href="#">ioaero::niter</a>
in	<i>nstep</i>	<a href="#">ioaero::nstep</a>
out	<i>sol_pt</i>	<a href="#">ioaero::sol_pt</a>
out	<i>sol_mb</i>	<a href="#">ioaero::sol_mb</a>
out	<i>error</i>	<a href="#">ioaero::error</a>
in	<i>ncond_mb</i>	<a href="#">ioaero::ncond_mb</a>
in	<i>ntimefun</i>	<a href="#">ioaero::ntimefun</a>
in	<i>frame</i>	<a href="#">ioaero::frame</a>
in, out	<i>mb_condition</i>	<a href="#">ioaero::mb_condition</a>
in	<i>distr_fun</i>	<a href="#">ioaero::distr_fun</a>
in	<i>curvature</i>	<a href="#">ioaero::curvature</a>
in	<i>omega_a0</i>	<a href="#">ioaero::omega_a0</a>
in	<i>v_root_a0</i>	<a href="#">ioaero::v_root_a0</a>
in	<i>omega_a_tf</i>	<a href="#">ioaero::omega_a_tf</a>
in	<i>v_root_a_tf</i>	<a href="#">ioaero::v_root_a_tf</a>
in	<i>simu_time</i>	<a href="#">ioaero::simu_time</a>
in	<i>time_function</i>	<a href="#">ioaero::time_function</a>
in, out	<i>init_cond</i>	<a href="#">ioaero::init_cond</a>
in	<i>analysis_flag</i>	<a href="#">ioaero::analysis_flag</a>
in, out	<i>nev</i>	<a href="#">ioaero::nev</a>
out	<i>eigen_val</i>	<a href="#">ioaero::eigen_val</a>
out	<i>eigen_vec_pt</i>	<a href="#">ioaero::eigen_vec_pt</a>
out	<i>eigen_vec_mb</i>	<a href="#">ioaero::eigen_vec_mb</a>
in	<i>aero_flag</i>	<a href="#">ioaero::aero_flag</a>
in	<i>grav_flag</i>	<a href="#">ioaero::grav_flag</a>

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)  
*Calculate the Jacobian matrix for each element.*
- subroutine, public `element::elemmass` (ndof\_el, elemM)  
*Calculate 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 element*
- real(dbl), dimension(nstrn, nstrn) `element::emass`  
*inverse of the mass matrix of the element*
- type(distribload), 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::ecabhalf`  
*eCab\*dL/2*
- real(dbl), dimension(ndim, ndim) `element::ectcabhalf`  
*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 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/postprocessor (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 modes shapes.*
- integer, public `ioaero::aero_flag`  
*0: no aero analysis; 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 assembles 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 independent.*

## 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 bisection 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  $J = \text{diff}C^T \cdot \text{vec} / \text{diff}\theta$ , return a 3x3 matrix with ith column corresponding to the derivative with respect to  $\theta_i$ .*
- real(dbl) function `prescribedcondition::loadintegration` (flag, dL, Le, func)  
*Calculate the load using Chebychev polynomials \*.*
- real(dbl) function, dimension(size(vec)) `prescribedcondition::transferfollower` (follower, vec, CT)  
*Transfer follower according to  $C^T \cdot \text{vec}$  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 [linsearch](#)  
*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 [linsearch\(\)](#)

```
subroutine newtonraphsonmumps::linsearch ( ) [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 assembles 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	<i>flag</i>	indicating whether it is the starting point or the ending point: -1-starting, 1-ending
in	<i>kp</i>	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	<i>flag</i>	indicating whether it is the starting point or the ending point: -1-starting, 1-ending
in	<i>term_pt</i>	the point number
in	<i>rhs_tm</i>	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::initff](#) ()  
*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

/home/bertrand/logiciels/gebtaero\_frama/README.md, 157  
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/CompositeBox.py, 157  
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/CompositePlate.py, 157  
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/CompositePly.py, 158  
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/CrossSection.py, 158  
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/ExternalMesh.py, 158  
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/Frame.py, 158  
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/GebtPlot.py, 159  
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/InputFile.py, 159  
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/IsoMaterial.py, 159  
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/OrthoMaterial.py, 159  
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/Simulation.py, 160  
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/TimeFunction.py, 160  
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/Wing.py, 161  
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/WingSection.py, 161  
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/\_\_\_init\_\_\_py, 157  
/home/bertrand/these/logiciels/programme/interface/src/gebtaero/ACOmegaaTFNumber.py, 160  
/home/bertrand/these/logiciels/programme/src/Analysis.f90, 161  
/home/bertrand/these/logiciels/programme/src/CP/Utime.f90, 163  
/home/bertrand/these/logiciels/programme/src/Eigen/SolveMumps.f90, 164  
/home/bertrand/these/logiciels/programme/src/Element.f90, 164  
/home/bertrand/these/logiciels/programme/src/Global/DataFun.f90, 166  
/home/bertrand/these/logiciels/programme/src/Oaero.f90, 170  
/home/bertrand/these/logiciels/programme/src/InternalData.f90, 169  
/home/bertrand/these/logiciels/programme/src/Member.f90, 172  
/home/bertrand/these/logiciels/programme/src/Preprocess.f90, 173  
/home/bertrand/these/logiciels/programme/src/PrescribedCondition.f90, 174  
/home/bertrand/these/logiciels/programme/src/SolveMumps.f90, 175  
/home/bertrand/these/logiciels/programme/src/System.f90, 176  
/home/bertrand/these/logiciels/programme/src/TimeFunction.f90, 177  
/home/bertrand/these/logiciels/programme/src/main/Aero.f90, 172  
/home/bertrand/these/logiciels/programme/src/\_\_\_init\_\_\_gebtaero::CompositeBox::CompositeBox, 94  
/home/bertrand/these/logiciels/programme/src/\_\_\_init\_\_\_gebtaero::CompositePlate::CompositePlate, 98  
/home/bertrand/these/logiciels/programme/src/\_\_\_init\_\_\_gebtaero::CompositePly::CompositePly, 102  
/home/bertrand/these/logiciels/programme/src/\_\_\_init\_\_\_gebtaero::CrossSection::CrossSection, 104  
/home/bertrand/these/logiciels/programme/src/\_\_\_init\_\_\_gebtaero::ExternalMesh::ExternalMesh, 110  
/home/bertrand/these/logiciels/programme/src/\_\_\_init\_\_\_gebtaero::Frame::Frame, 115  
/home/bertrand/these/logiciels/programme/src/\_\_\_init\_\_\_gebtaero::InputFile::InputFile, 119  
/home/bertrand/these/logiciels/programme/src/\_\_\_init\_\_\_gebtaero::IsoMaterial::IsoMaterial, 126  
/home/bertrand/these/logiciels/programme/src/\_\_\_init\_\_\_gebtaero::OrthoMaterial::OrthoMaterial, 130  
/home/bertrand/these/logiciels/programme/src/\_\_\_init\_\_\_gebtaero::Simulation::Simulation, 134  
/home/bertrand/these/logiciels/programme/src/\_\_\_init\_\_\_gebtaero::TimeFunction::TimeFunction, 144  
/home/bertrand/these/logiciels/programme/src/\_\_\_init\_\_\_gebtaero::Wing::Wing, 147  
/home/bertrand/these/logiciels/programme/src/\_\_\_init\_\_\_gebtaero::WingSection::WingSection, 152  
a  
a\_element, 17  
a\_flag  
a\_element, 18  
ACOmegaa  
gebtaero::InputFile::InputFile, 121  
ACOmegaaTFNumber  
gebtaero::InputFile::InputFile, 121  
ACVa  
gebtaero::InputFile::InputFile, 122  
ACVaTFNumber  
gebtaero::InputFile::InputFile, 122  
aero\_flag  
ioaero, 57  
AeroFlag  
gebtaero::InputFile::InputFile, 122  
aerodyn\_coef  
internaldata::memberinf, 128  
ioaero, 57  
allo\_stat  
globaldatafun, 46  
alpha

- element, [18](#)
- alpha\_ac
  - element, [18](#)
- AlphaAC
  - gebtaero::InputFile::InputFile, [122](#)
- alphadot
  - element, [18](#)
- alphadotdot
  - element, [18](#)
- analysis
  - Analysis.f90, [161](#)
- Analysis.f90
  - analysis, [161](#)
- analysis\_flag
  - ioaero, [58](#)
- AnalysisFlag
  - gebtaero::InputFile::InputFile, [122](#)
- AppendComponent
  - gebtaero::ExternalMesh::ExternalMesh, [110](#)
- AppendFunctionEntrieHarmonic
  - gebtaero::TimeFunction::TimeFunction, [144](#)
- AppendFunctionEntriePieceWise
  - gebtaero::TimeFunction::TimeFunction, [144](#)
- AppendPly
  - gebtaero::CompositePlate::CompositePlate, [98](#)
- AppendTimeFunction
  - gebtaero::InputFile::InputFile, [120](#)
- AppendWingSection
  - gebtaero::Wing::Wing, [147](#)
- arpack
  - eigenmumps, [11](#)
  - ioaero, [58](#)
- arpack\_mod
  - globaldatafun, [46](#)
- assemble\_flag
  - internaldata, [52](#)
- assemblejacobian
  - system, [86](#)
- assemblenemberjacobian
  - member, [68](#)
- assemblenemberrhs
  - member, [69](#)
- assemblepointj
  - System.f90, [176](#)
- assemblepointrhs
  - System.f90, [177](#)
- assemblerhs
  - system, [87](#)
- aw
  - eigenmumps, [12](#)
  - element, [18](#)
- Axis
  - gebtaero::Frame::Frame, [116](#)
- b
  - element, [19](#)
- beta
  - element, [19](#)
- beta\_ac
  - element, [19](#)
- BetaAC
  - gebtaero::InputFile::InputFile, [122](#)
- bw
  - element, [19](#)
- c
  - element, [19](#)
- char\_len
  - ioaero, [58](#)
- Chord
  - gebtaero::CompositePlate::CompositePlate, [100](#)
  - gebtaero::ExternalMesh::ExternalMesh, [112](#)
  - gebtaero::WingSection::WingSection, [154](#)
- chord
  - element, [19](#)
- coef
  - system, [88](#)
- Components
  - gebtaero::ExternalMesh::ExternalMesh, [112](#)
- ComputeElasticCenterFromFlexMat
  - gebtaero::utils, [32](#)
- ComputeElementSurfAndCG
  - gebtaero::ExternalMesh::ExternalMesh, [111](#)
- ComputeMassMatrix
  - gebtaero::CompositeBox::CompositeBox, [94](#)
  - gebtaero::CompositePlate::CompositePlate, [98](#)
- ComputeMassMatrixFromMesh
  - gebtaero::ExternalMesh::ExternalMesh, [111](#)
- cond\_all
  - internaldata, [52](#)
- coord
  - ioaero, [58](#)
- coordinate
  - internaldata::memberinf, [128](#)
- CorrelateTab
  - gebtaero::utils, [32](#)
- cputime, [9](#)
  - finish, [10](#)
  - rate, [10](#)
  - start, [10](#)
  - tic, [9](#)
  - toc, [9](#)
- CreateFbdFile
  - gebtaero::CompositeBox::CompositeBox, [94](#)
  - gebtaero::CompositePlate::CompositePlate, [98](#)
- CreateInpFile
  - gebtaero::CompositeBox::CompositeBox, [94](#)
  - gebtaero::CompositePlate::CompositePlate, [99](#)
  - gebtaero::ExternalMesh::ExternalMesh, [111](#)
- CreatePeriodicEq
  - gebtaero::CompositeBox::CompositeBox, [95](#)
  - gebtaero::CompositePlate::CompositePlate, [99](#)
  - gebtaero::ExternalMesh::ExternalMesh, [111](#)
  - gebtaero::utils, [33](#)
- CrossSection
  - gebtaero::WingSection::WingSection, [154](#)
- CrossSections
  - gebtaero::Wing::Wing, [149](#)

- crossproduct
  - globaldatafun, 38
- ct\_theta
  - globaldatafun, 38
- ct\_theta\_t
  - globaldatafun, 39
- ctcabhdot
  - element, 20
- ctcabpdot
  - element, 20
- ctcabph
  - solvemumps, 80
- currentvalues
  - timefunctionmodule, 90
- curvature
  - ioaero, 58
- curvebeamfun
  - prepromodule, 72
- dbl
  - globaldatafun, 46
- deb\_name
  - internaldata, 52
- debug
  - internaldata, 52
- DeformedModalFlutterSpeed
  - gebtaero::Simulation::Simulation, 135
- DeformedModalFlutterSpeedSorted
  - gebtaero::Simulation::Simulation, 135
- deg\_2\_rad
  - globaldatafun, 47
- dir\_lift
  - element, 20
- dir\_moment
  - element, 20
- dircosinetrodrigues
  - globaldatafun, 39
- DisplaySectionDeformation
  - gebtaero::CompositeBox::CompositeBox, 95
  - gebtaero::CompositePlate::CompositePlate, 99
  - gebtaero::ExternalMesh::ExternalMesh, 112
- distr\_fun
  - ioaero, 59
  - prescribedcondition::distriload, 109
- dl
  - element, 20
  - internaldata::memberinf, 128
- dof
  - prescribedcondition::prescriinf, 132
- dof\_all
  - internaldata, 52
- Down
  - gebtaero::CompositeBox::CompositeBox, 96
- E
  - gebtaero::IsoMaterial::IsoMaterial, 126
- e1
  - globaldatafun, 47
- e1gammad
  - element, 20
- ecab
  - element, 21
- ecabhalf1
  - element, 21
- ecaf
  - element, 21
- ech\_name
  - ioaero, 59
- ect
  - element, 21
- ectcab
  - element, 21
- ectcabhalf1
  - element, 22
- eflex
  - element, 22
- eigen\_output
  - globaldatafun, 47
- eigen\_val
  - ioaero, 59
- eigen\_vec\_mb
  - ioaero, 59
- eigen\_vec\_pt
  - ioaero, 59
- EigenFreqDamping
  - gebtaero::GebtPlot::GebtPlot, 117
- EigenFreqDampingUnsorted
  - gebtaero::GebtPlot::GebtPlot, 117
- EigenTab
  - gebtaero::Simulation::Simulation, 135
- EigenTabSorted
  - gebtaero::Simulation::Simulation, 136
- eigenmumps, 10
  - arpack, 11
  - aw, 12
  - eigensolvemumps, 12
  - ierr, 13
  - mumps\_par, 13
- eigenoutput
  - ioaero, 60
- eigensolvemumps
  - eigenmumps, 12
- Eigenvalues
  - gebtaero::Simulation::Simulation, 136
- ein
  - ioaero, 60
- EI
  - gebtaero::OrthoMaterial::OrthoMaterial, 130
- ElasticCenter
  - gebtaero::CrossSection::CrossSection, 107
- element, 13
  - a, 17
  - a\_flag, 18
  - alpha, 18
  - alpha\_ac, 18
  - alphadot, 18
  - alphadotdot, 18

- aw, [18](#)
- b, [19](#)
- beta, [19](#)
- beta\_ac, [19](#)
- bw, [19](#)
- c, [19](#)
- chord, [19](#)
- ctcabhdot, [20](#)
- ctcabpdot, [20](#)
- dir\_lift, [20](#)
- dir\_moment, [20](#)
- dl, [20](#)
- e1gammad, [20](#)
- ecab, [21](#)
- ecabhalf, [21](#)
- ecaf, [21](#)
- ect, [21](#)
- ectcab, [21](#)
- ectcabhalf, [22](#)
- eflex, [22](#)
- elemeqn, [16](#)
- elemjacobian, [16](#)
- elemmass, [16](#)
- emass, [22](#)
- eomega\_a, [22](#)
- epi, [22](#)
- ev\_i, [23](#)
- exist\_load, [23](#)
- extractelementproperties, [17](#)
- fi, [23](#)
- follower\_load, [23](#)
- g\_flag, [23](#)
- hdot, [23](#)
- hdotdot, [24](#)
- hi, [24](#)
- ident, [24](#)
- jalpha\_theta, [24](#)
- jalphadot\_ph, [24](#)
- jalphadot\_theta, [24](#)
- jalphadotdot\_ph, [25](#)
- jalphadotdot\_phdot, [25](#)
- jdir\_lift, [25](#)
- jdir\_moment, [25](#)
- jhdot\_ph, [25](#)
- jhdot\_theta, [25](#)
- jhdotdot\_ph, [26](#)
- jhdotdot\_phdot, [26](#)
- jlambda0\_lambda, [26](#)
- jlambda0\_phdot, [26](#)
- jlift, [26](#)
- moment, [26](#)
- kappa, [27](#)
- lambda, [27](#)
- lambda0, [27](#)
- lambda0dot, [27](#)
- le, [27](#)
- load, [27](#)
- mi, [28](#)
- omegai, [28](#)
- p, [28](#)
- rho, [28](#)
- theta, [28](#)
- thetadot, [28](#)
- u, [29](#)
- ui, [29](#)
- uidot, [29](#)
- vi, [29](#)
- wind, [29](#)
- x\_cg, [29](#)
- elements
  - gebtaero::ExternalMesh::ExternalMesh, [112](#)
- elemeqn
  - element, [16](#)
- elemjacobian
  - element, [16](#)
- elemmass
  - element, [16](#)
- emass
  - element, [22](#)
- entries
  - timefunctionmodule::timefunction, [142](#)
- eomega\_a
  - element, [22](#)
- epi
  - element, [22](#)
- EquilibriumAoA
  - gebtaero::Simulation::Simulation, [136](#)
- error
  - ioaero, [60](#)
- Et
  - gebtaero::OrthoMaterial::OrthoMaterial, [131](#)
- ev\_i
  - element, [23](#)
- exist\_load
  - element, [23](#)
- existpi
  - prescribedcondition, [75](#)
- extract2delement
  - globaldatafun, [39](#)
- extractelementproperties
  - element, [17](#)
- extractelementvalues
  - solvemumps, [81](#)
- extractmemberproperties
  - member, [70](#)
- extractsolution
  - solvemumps, [81](#)
- factoriel
  - globaldatafun, [40](#)
- fi
  - element, [23](#)
- FileName
  - gebtaero::InputFile::InputFile, [123](#)
- fileopen
  - globaldatafun, [40](#)
- finish



- cputime, 10
- FlexibilityMatrix
  - gebtaero::CrossSection::CrossSection, 108
- flutter\_flag
  - globaldatafun, 47
- flutter\_limit
  - globaldatafun, 47
- FlutterVtk
  - gebtaero::Simulation::Simulation, 137
- fmt\_int
  - globaldatafun, 48
- fmt\_real
  - globaldatafun, 48
- follower
  - prescribedcondition::distriload, 109
  - prescribedcondition::prescriinf, 132
- follower\_all
  - internaldata, 52
- follower\_load
  - element, 23
- followerj
  - prescribedcondition, 75
- Frame
  - gebtaero::WingSection::WingSection, 154
- frame
  - ioaero, 60
- FrameMatrix
  - gebtaero::Frame::Frame, 116
- Frames
  - gebtaero::Wing::Wing, 149
- fun\_type
  - timefunctionmodule::timefunction, 142
- fun\_val
  - timefunctionmodule::timefunction, 142
- FunctionEnd
  - gebtaero::TimeFunction::TimeFunction, 145
- FunctionEntries
  - gebtaero::TimeFunction::TimeFunction, 146
- FunctionStart
  - gebtaero::TimeFunction::TimeFunction, 146
- FunctionType
  - gebtaero::TimeFunction::TimeFunction, 146
- g\_flag
  - element, 23
- gebt
  - mainAero.f90, 172
- gebtaero, 30
- gebtaero.CompositeBox, 30
- gebtaero.CompositeBox.CompositeBox, 93
- gebtaero.CompositePlate, 30
- gebtaero.CompositePlate.CompositePlate, 97
- gebtaero.CompositePly, 30
- gebtaero.CompositePly.CompositePly, 101
- gebtaero.CrossSection, 31
- gebtaero.CrossSection.CrossSection, 103
- gebtaero.ExternalMesh, 31
- gebtaero.ExternalMesh.ExternalMesh, 109
- gebtaero.Frame, 31
- gebtaero.Frame.Frame, 115
- gebtaero.GebtPlot, 31
- gebtaero.GebtPlot.GebtPlot, 117
- gebtaero.InputFile, 31
- gebtaero.InputFile.InputFile, 118
- gebtaero.IsoMaterial, 31
- gebtaero.IsoMaterial.IsoMaterial, 125
- gebtaero.OrthoMaterial, 31
- gebtaero.OrthoMaterial.OrthoMaterial, 129
- gebtaero.Simulation, 32
- gebtaero.Simulation.Simulation, 134
- gebtaero.TimeFunction, 32
- gebtaero.TimeFunction.TimeFunction, 143
- gebtaero.utils, 32
- gebtaero.Wing, 35
- gebtaero.Wing.Wing, 146
- gebtaero.WingSection, 36
- gebtaero.WingSection.WingSection, 151
- gebtaero::CompositeBox::CompositeBox
  - \_\_init\_\_, 94
  - ComputeMassMatrix, 94
  - CreateFbdFile, 94
  - CreateInpFile, 94
  - CreatePeriodicEq, 95
  - DisplaySectionDeformation, 95
  - Down, 96
  - GetHeight, 95
  - GetOffsets, 95
  - GetWidth, 95
  - Height, 96
  - Left, 96
  - OffsetY, 96
  - OffsetZ, 96
  - Right, 96
  - Up, 97
  - Width, 97
- gebtaero::CompositePlate::CompositePlate
  - \_\_init\_\_, 98
  - AppendPly, 98
  - Chord, 100
  - ComputeMassMatrix, 98
  - CreateFbdFile, 98
  - CreateInpFile, 99
  - CreatePeriodicEq, 99
  - DisplaySectionDeformation, 99
  - GetLayup, 99
  - GetOffsets, 100
  - GetTotThickness, 100
  - Layup, 100
  - Materials, 100
  - OffsetY, 101
  - OffsetZ, 101
  - Orientations, 101
  - TotThickness, 101
- gebtaero::CompositePly::CompositePly
  - \_\_init\_\_, 102
  - GetMaterial, 102
  - GetOrientation, 102

- GetThickness, 102
- Material, 103
- Orientation, 103
- Thickness, 103
- gebtaero::CrossSection::CrossSection
  - \_\_init\_\_, 104
  - ElasticCenter, 107
  - FlexibilityMatrix, 108
  - GetFlexibilityMatrix, 104
  - GetMassMatrix, 104
  - MassMatrix, 108
  - SetFlexibilityMatrixByBox, 104
  - SetFlexibilityMatrixByIsotropicValues, 105
  - SetFlexibilityMatrixByMesh, 105
  - SetFlexibilityMatrixByPlate, 105
  - SetFlexibilityMatrixByRectBeamValues, 106
  - SetMassMatrix, 106
  - SetMassMatrixByBox, 106
  - SetMassMatrixByMesh, 107
  - SetMassMatrixByPlate, 107
  - SetMassMatrixByRectBeamValues, 107
- gebtaero::ExternalMesh::ExternalMesh
  - \_\_init\_\_, 110
  - AppendComponent, 110
  - Chord, 112
  - Components, 112
  - ComputeElementSurfAndCG, 111
  - ComputeMassMatrixFromMesh, 111
  - CreateInpFile, 111
  - CreatePeriodicEq, 111
  - DisplaySectionDeformation, 112
  - elements, 112
  - GetMeshFile, 112
  - Lx, 113
  - Materials, 113
  - MeshFile, 113
  - ncurv, 113
  - nodes, 113
  - nstrain, 113
  - OffsetY, 114
  - OffsetZ, 114
  - Orientations, 114
  - TotThickness, 114
  - x0, 114
- gebtaero::Frame::Frame
  - \_\_init\_\_, 115
  - Axis, 116
  - FrameMatrix, 116
  - GetAxis, 115
  - GetFrameMatrix, 115
  - GetTwist, 116
  - Twist, 116
- gebtaero::GebtPlot::GebtPlot
  - EigenFreqDamping, 117
  - EigenFreqDampingUnsorted, 117
  - ParaviewOutput, 117
  - WriteParaviewScript, 118
- gebtaero::InputFile::InputFile
  - \_\_init\_\_, 119
  - ACOmega, 121
  - ACOmegaTFNumber, 121
  - ACVa, 122
  - ACVaTFNumber, 122
  - AeroFlag, 122
  - AlphaAC, 122
  - AnalysisFlag, 122
  - AppendTimeFunction, 120
  - BetaAC, 122
  - FileName, 123
  - GetAnalysisFlag, 120
  - GetFileName, 120
  - GetName, 120
  - GetTimeFunction, 120
  - GravFlag, 123
  - Name, 123
  - Nev, 123
  - Niter, 123
  - Nstep, 123
  - Nvtk, 124
  - RemoveInputFile, 121
  - Rho, 124
  - SimuEnd, 124
  - SimuStart, 124
  - TimeFunctions, 124
  - Vinf, 124
  - Wing, 125
  - WriteInpFile, 121
  - WriteInputFile, 121
  - Xcg, 125
- gebtaero::IsoMaterial::IsoMaterial
  - \_\_init\_\_, 126
  - E, 126
  - GetDensity, 126
  - GetIso, 126
  - Nu, 126
  - Rho, 127
- gebtaero::OrthoMaterial::OrthoMaterial
  - \_\_init\_\_, 130
  - EI, 130
  - Et, 131
  - GetDensity, 130
  - GetOrtho, 130
  - Glt, 131
  - Nult, 131
  - Rho, 131
- gebtaero::Simulation::Simulation
  - \_\_init\_\_, 134
  - DeformedModalFlutterSpeed, 135
  - DeformedModalFlutterSpeedSorted, 135
  - EigenTab, 135
  - EigenTabSorted, 136
  - Eigenvalues, 136
  - EquilibriumAoA, 136
  - FlutterVtk, 137
  - GetWing, 137
  - Input, 141

- ModalCriticalSpeed, [137](#)
- ModalDivergenceSpeed, [138](#)
- ModalDivergenceSpeedSorted, [138](#)
- ModalFlutterSpeed, [138](#)
- ModalFlutterSpeedSorted, [139](#)
- StaticLoads, [139](#)
- TemporalDivergenceSpeed, [139](#)
- TemporalDynamic, [140](#)
- TemporalFlutterSpeed, [140](#)
- Wing, [141](#)
- gebtaero::TimeFunction::TimeFunction
  - \_\_init\_\_, [144](#)
  - AppendFunctionEntrieHarmonic, [144](#)
  - AppendFunctionEntriePieceWise, [144](#)
  - FunctionEnd, [145](#)
  - FunctionEntries, [146](#)
  - FunctionStart, [146](#)
  - FunctionType, [146](#)
  - GetFunctionEnd, [144](#)
  - GetFunctionEntrie, [145](#)
  - GetFunctionEntries, [145](#)
  - GetFunctionStart, [145](#)
  - GetFunctionType, [145](#)
- gebtaero::Wing::Wing
  - \_\_init\_\_, [147](#)
  - AppendWingSection, [147](#)
  - CrossSections, [149](#)
  - Frames, [149](#)
  - GetCrossSections, [148](#)
  - GetFrames, [148](#)
  - GetKpList, [148](#)
  - GetName, [148](#)
  - GetSurface, [148](#)
  - GetWeight, [149](#)
  - GetWingRootPosition, [149](#)
  - GetWingSections, [149](#)
  - KpList, [150](#)
  - Name, [150](#)
  - WingRootPosition, [150](#)
  - WingSections, [150](#)
- gebtaero::WingSection::WingSection
  - \_\_init\_\_, [152](#)
  - Chord, [154](#)
  - CrossSection, [154](#)
  - Frame, [154](#)
  - GetChord, [152](#)
  - GetCrossSection, [152](#)
  - GetFrame, [152](#)
  - GetHalfChord, [152](#)
  - GetNumberOfElements, [153](#)
  - GetParameterA, [153](#)
  - GetSectionLength, [153](#)
  - HalfChord, [155](#)
  - NumberOfElements, [155](#)
  - ParameterA, [155](#)
  - SectionLength, [155](#)
  - SetChord, [153](#)
  - SetNumberOfElements, [153](#)
  - SetParameterA, [154](#)
  - SetSectionLength, [154](#)
- gebtaero::utils
  - ComputeElasticCenterFromFlexMat, [32](#)
  - CorrelateTab, [32](#)
  - CreatePeriodicEq, [33](#)
  - ReadEigenVec, [33](#)
  - ReadFlexibilityFromDisp, [33](#)
  - ReadFromPipe, [33](#)
  - ReadLoadsFromPipe, [33](#)
  - ReadModesFromPipe, [34](#)
  - ReadNodesField, [34](#)
  - RemoveFiles, [34](#)
  - RemoveMeshFiles, [34](#)
  - RunFbdFile, [34](#)
  - RunFrdFile, [35](#)
  - RunInpFile, [35](#)
  - RunParaviewScript, [35](#)
  - RunUnvConv, [35](#)
- GetAnalysisFlag
  - gebtaero::InputFile::InputFile, [120](#)
- GetAxis
  - gebtaero::Frame::Frame, [115](#)
- GetChord
  - gebtaero::WingSection::WingSection, [152](#)
- GetCrossSection
  - gebtaero::WingSection::WingSection, [152](#)
- GetCrossSections
  - gebtaero::Wing::Wing, [148](#)
- GetDensity
  - gebtaero::IsoMaterial::IsoMaterial, [126](#)
  - gebtaero::OrthoMaterial::OrthoMaterial, [130](#)
- GetFileName
  - gebtaero::InputFile::InputFile, [120](#)
- GetFlexibilityMatrix
  - gebtaero::CrossSection::CrossSection, [104](#)
- GetFrame
  - gebtaero::WingSection::WingSection, [152](#)
- GetFrameMatrix
  - gebtaero::Frame::Frame, [115](#)
- GetFrames
  - gebtaero::Wing::Wing, [148](#)
- GetFunctionEnd
  - gebtaero::TimeFunction::TimeFunction, [144](#)
- GetFunctionEntrie
  - gebtaero::TimeFunction::TimeFunction, [145](#)
- GetFunctionEntries
  - gebtaero::TimeFunction::TimeFunction, [145](#)
- GetFunctionStart
  - gebtaero::TimeFunction::TimeFunction, [145](#)
- GetFunctionType
  - gebtaero::TimeFunction::TimeFunction, [145](#)
- GetHalfChord
  - gebtaero::WingSection::WingSection, [152](#)
- GetHeight
  - gebtaero::CompositeBox::CompositeBox, [95](#)
- GetIso
  - gebtaero::IsoMaterial::IsoMaterial, [126](#)

- GetKpList
  - gebtaero::Wing::Wing, 148
- GetLayup
  - gebtaero::CompositePlate::CompositePlate, 99
- GetMassMatrix
  - gebtaero::CrossSection::CrossSection, 104
- GetMaterial
  - gebtaero::CompositePly::CompositePly, 102
- GetMeshFile
  - gebtaero::ExternalMesh::ExternalMesh, 112
- GetName
  - gebtaero::InputFile::InputFile, 120
  - gebtaero::Wing::Wing, 148
- GetNumberOfElements
  - gebtaero::WingSection::WingSection, 153
- GetOffsets
  - gebtaero::CompositeBox::CompositeBox, 95
  - gebtaero::CompositePlate::CompositePlate, 100
- GetOrientation
  - gebtaero::CompositePly::CompositePly, 102
- GetOrtho
  - gebtaero::OrthoMaterial::OrthoMaterial, 130
- GetParameterA
  - gebtaero::WingSection::WingSection, 153
- GetSectionLength
  - gebtaero::WingSection::WingSection, 153
- GetSurface
  - gebtaero::Wing::Wing, 148
- GetThickness
  - gebtaero::CompositePly::CompositePly, 102
- GetTimeFunction
  - gebtaero::InputFile::InputFile, 120
- GetTotThickness
  - gebtaero::CompositePlate::CompositePlate, 100
- GetTwist
  - gebtaero::Frame::Frame, 116
- GetWeight
  - gebtaero::Wing::Wing, 149
- GetWidth
  - gebtaero::CompositeBox::CompositeBox, 95
- GetWing
  - gebtaero::Simulation::Simulation, 137
- GetWingRootPosition
  - gebtaero::Wing::Wing, 149
- GetWingSections
  - gebtaero::Wing::Wing, 149
- getdistributedload
  - prescribedcondition, 76
- getload
  - prescribedcondition, 76
- getloadj
  - prescribedcondition, 76
- getprescribeddof
  - prescribedcondition, 77
- getprescribedval
  - prescribedcondition, 77
- gettimefunction
  - timefunctionmodule, 90
- globaldatafun, 36
  - allo\_stat, 46
  - arpack\_mod, 46
  - crossproduct, 38
  - ct\_theta, 38
  - ct\_theta\_t, 39
  - dbl, 46
  - deg\_2\_rad, 47
  - dircosinetrodrigues, 39
  - e1, 47
  - eigen\_output, 47
  - extract2delement, 39
  - factoriel, 40
  - fileopen, 40
  - flutter\_flag, 47
  - flutter\_limit, 47
  - fmt\_int, 48
  - fmt\_real, 48
  - grav, 48
  - i3, 48
  - in\_stat, 48
  - insert1delement, 40
  - invert, 41
  - ioerror, 41
  - itochar, 42
  - mata, 42
  - matd, 42
  - matmul3, 42
  - matmul\_sparse, 42
  - memb\_const, 49
  - memoryerror, 43
  - ndim, 49
  - ndof\_nd, 49
  - norm, 43
  - nstates, 49
  - nstrn, 49
  - outerproduct, 43
  - peters, 43
  - pi, 50
  - prod, 44
  - rad\_2\_deg, 50
  - runmod, 50
  - solver, 50
  - tilde, 44
  - titleprint, 44
  - tolerance, 50
  - vecb, 44
  - vecc, 45
  - vecd, 45
  - writeerror, 45
  - writeintvector, 45
  - writerealvector, 46
- globaldatafun::writevec, 155
  - writeintvector, 156
  - writerealvector, 156
- Glt
  - gebtaero::OrthoMaterial::OrthoMaterial, 131
- grav

- globaldatafun, 48
- grav\_flag
  - ioaero, 60
- GravFlag
  - gebtaero::InputFile::InputFile, 123
- HalfChord
  - gebtaero::WingSection::WingSection, 155
- hdot
  - element, 23
- hdotdot
  - element, 24
- Height
  - gebtaero::CompositeBox::CompositeBox, 96
- hi
  - element, 24
- i3
  - globaldatafun, 48
- id
  - prescribedcondition::prescriinf, 132
- ident
  - element, 24
- ierr
  - eigenmumps, 13
  - solvemumps, 85
- in
  - ioaero, 61
- in\_stat
  - globaldatafun, 48
- index\_kp
  - internaldata, 53
- index\_mb
  - internaldata, 53
- init
  - ioaero, 61
- init\_cond
  - ioaero, 61
- init\_flag
  - internaldata, 53
- init\_memb
  - internaldata, 53
- init\_name
  - ioaero, 61
- initpi
  - prescribedcondition, 77
- initpiaero
  - prescribedcondition, 78
- inittf
  - timefunctionmodule, 91
- inp\_name
  - ioaero, 61
- Input
  - gebtaero::Simulation::Simulation, 141
- input
  - ioaero, 57
- inputechoprescribedconditions
  - prescribedcondition, 78
- inputechotimefunctions
  - timefunctionmodule, 91
- insert1delement
  - globaldatafun, 40
- insertelementvalues
  - solvemumps, 82
- internaldata, 51
  - assemble\_flag, 52
  - cond\_all, 52
  - deb\_name, 52
  - debug, 52
  - dof\_all, 52
  - follower\_all, 52
  - index\_kp, 53
  - index\_mb, 53
  - init\_flag, 53
  - init\_memb, 53
  - iout, 53
  - nemax, 53
  - nsiz, 54
  - nzelemmax, 54
  - two\_divide\_dt, 54
  - xyz\_pt1, 54
- internaldata::memberinf, 127
  - aerodyn\_coef, 128
  - coordinate, 128
  - dl, 128
  - le, 128
  - mate, 128
  - ncol\_memb, 128
  - ndiv, 129
  - triad, 129
- invert
  - globaldatafun, 41
- ioaero, 54
  - aero\_flag, 57
  - aerodyn\_coef, 57
  - analysis\_flag, 58
  - arpack, 58
  - char\_len, 58
  - coord, 58
  - curvature, 58
  - distr\_fun, 59
  - ech\_name, 59
  - eigen\_val, 59
  - eigen\_vec\_mb, 59
  - eigen\_vec\_pt, 59
  - eigenoutput, 60
  - ein, 60
  - error, 60
  - frame, 60
  - grav\_flag, 60
  - in, 61
  - init, 61
  - init\_cond, 61
  - init\_name, 61
  - inp\_name, 61
  - input, 57
  - material, 62

- mb\_condition, 62
- member, 62
- ncond\_mb, 62
- ncond\_pt, 62
- ncurv, 63
- ndistrfun, 63
- ndof\_el, 63
- nelem, 63
- nev, 63
- nframe, 64
- niter, 64
- nkp, 64
- nmate, 64
- nmemb, 65
- nstep, 65
- ntimefun, 65
- nvtk, 65
- omega\_a0, 65
- omega\_a\_tf, 66
- out, 66
- out\_name, 66
- output, 57
- outputvtk, 57
- pt\_condition, 66
- simu\_time, 66
- sol\_mb, 67
- sol\_pt, 67
- time\_function, 67
- v\_root\_a0, 67
- v\_root\_a\_tf, 67
- velocity\_str, 68
- ioerror
  - globaldatafun, 41
- iout
  - internaldata, 53
- irn
  - system, 88
- itochar
  - globaldatafun, 42
- jalpha\_theta
  - element, 24
- jalphadot\_ph
  - element, 24
- jalphadot\_theta
  - element, 24
- jalphadotdot\_ph
  - element, 25
- jalphadotdot\_phdot
  - element, 25
- jcن
  - system, 88
- jdir\_lift
  - element, 25
- jdir\_moment
  - element, 25
- jhdot\_ph
  - element, 25
- jhdot\_theta
  - element, 25
- jhdotdot\_ph
  - element, 26
- jhdotdot\_phdot
  - element, 26
- jlambda0\_lambda
  - element, 26
- jlambda0\_phdot
  - element, 26
- jlift
  - element, 26
- jmoment
  - element, 26
- kappa
  - element, 27
- kp\_cond
  - system, 89
- kp\_dof
  - system, 89
- kp\_follower
  - system, 89
- KpList
  - gebtaero::Wing::Wing, 150
- lambda
  - element, 27
- lambda0
  - element, 27
- lambdadot
  - element, 27
- Layup
  - gebtaero::CompositePlate::CompositePlate, 100
- le
  - element, 27
  - internaldata::memberinf, 128
- Left
  - gebtaero::CompositeBox::CompositeBox, 96
- linearsolutionmumps
  - solvemumps, 82
- linesearch
  - SolveMumps.f90, 175
- load
  - element, 27
- loadintegration
  - prescribedcondition, 78
- Lx
  - gebtaero::ExternalMesh::ExternalMesh, 113
- mainAero.f90
  - gebt, 172
- MassMatrix
  - gebtaero::CrossSection::CrossSection, 108
- mata
  - globaldatafun, 42
- matd
  - globaldatafun, 42
- mate
  - internaldata::memberinf, 128

- Material
  - gebtaero::CompositePly::CompositePly, 103
- material
  - ioaero, 62
- Materials
  - gebtaero::CompositePlate::CompositePlate, 100
  - gebtaero::ExternalMesh::ExternalMesh, 113
- matmul3
  - globaldatafun, 42
- matmul\_sparse
  - globaldatafun, 42
- mb\_condition
  - ioaero, 62
- memb\_const
  - globaldatafun, 49
- member, 68
  - assemblenemberjacobian, 68
  - assemblenemberrhs, 69
  - extractmemberproperties, 70
  - ioaero, 62
  - ncol\_memb, 70
  - ndiv, 71
- memberproperties
  - prepromodule, 72
- memoryerror
  - globaldatafun, 43
- MeshFile
  - gebtaero::ExternalMesh::ExternalMesh, 113
- mi
  - element, 28
- ModalCriticalSpeed
  - gebtaero::Simulation::Simulation, 137
- ModalDivergenceSpeed
  - gebtaero::Simulation::Simulation, 138
- ModalDivergenceSpeedSorted
  - gebtaero::Simulation::Simulation, 138
- ModalFlutterSpeed
  - gebtaero::Simulation::Simulation, 138
- ModalFlutterSpeedSorted
  - gebtaero::Simulation::Simulation, 139
- mumps\_par
  - eigenmumps, 13
  - solvemumps, 85
- Name
  - gebtaero::InputFile::InputFile, 123
  - gebtaero::Wing::Wing, 150
- ncol\_memb
  - internaldata::memberinf, 128
  - member, 70
  - prepromodule, 74
- ncond\_mb
  - ioaero, 62
- ncond\_pt
  - ioaero, 62
- ncurv
  - gebtaero::ExternalMesh::ExternalMesh, 113
  - ioaero, 63
- ndim
  - globaldatafun, 49
- ndistrfun
  - ioaero, 63
- ndiv
  - internaldata::memberinf, 129
  - member, 71
  - prepromodule, 74
- ndof\_el
  - ioaero, 63
- ndof\_nd
  - globaldatafun, 49
- ne
  - system, 89
- nelem
  - ioaero, 63
- nemax
  - internaldata, 53
- Nev
  - gebtaero::InputFile::InputFile, 123
- nev
  - ioaero, 63
- newtonraphsonmumps
  - solvemumps, 84
- nframe
  - ioaero, 64
- Niter
  - gebtaero::InputFile::InputFile, 123
- niter
  - ioaero, 64
- nkp
  - ioaero, 64
- nmate
  - ioaero, 64
- nmemb
  - ioaero, 65
- nodes
  - gebtaero::ExternalMesh::ExternalMesh, 113
- norm
  - globaldatafun, 43
- nsiz
  - internaldata, 54
- nstates
  - globaldatafun, 49
- Nstep
  - gebtaero::InputFile::InputFile, 123
- nstep
  - ioaero, 65
- nstrain
  - gebtaero::ExternalMesh::ExternalMesh, 113
- nstrn
  - globaldatafun, 49
- ntimefun
  - ioaero, 65
- Nu
  - gebtaero::IsoMaterial::IsoMaterial, 126
- Nult
  - gebtaero::OrthoMaterial::OrthoMaterial, 131
- NumberOfElements

- gebtaero::WingSection::WingSection, 155
- Nvtk
  - gebtaero::InputFile::InputFile, 124
- nvtk
  - ioaero, 65
- nzelemmax
  - internaldata, 54
- OffsetY
  - gebtaero::CompositeBox::CompositeBox, 96
  - gebtaero::CompositePlate::CompositePlate, 101
  - gebtaero::ExternalMesh::ExternalMesh, 114
- OffsetZ
  - gebtaero::CompositeBox::CompositeBox, 96
  - gebtaero::CompositePlate::CompositePlate, 101
  - gebtaero::ExternalMesh::ExternalMesh, 114
- omega\_a0
  - ioaero, 65
- omega\_a\_tf
  - ioaero, 66
- omegai
  - element, 28
- Orientation
  - gebtaero::CompositePly::CompositePly, 103
- Orientations
  - gebtaero::CompositePlate::CompositePlate, 101
  - gebtaero::ExternalMesh::ExternalMesh, 114
- out
  - ioaero, 66
- out\_name
  - ioaero, 66
- outerproduct
  - globaldatafun, 43
- output
  - ioaero, 57
- outputvtk
  - ioaero, 57
- p
  - element, 28
- ParameterA
  - gebtaero::WingSection::WingSection, 155
- ParaviewOutput
  - gebtaero::GebtPlot::GebtPlot, 117
- peters
  - globaldatafun, 43
- phase\_val
  - timefunctionmodule::timefunction, 142
- pi
  - globaldatafun, 50
- pointfollowerj
  - system, 88
- preprocess
  - prepromodule, 73
- prepromodule, 71
  - curvebeamfun, 72
  - memberproperties, 72
  - ncol\_mem, 74
  - ndiv, 74
  - preprocess, 73
  - rtbis, 73
- prescribedcondition, 74
  - existpi, 75
  - followerj, 75
  - getdistributedload, 76
  - getload, 76
  - getloadj, 76
  - getprescribeddof, 77
  - getprescribedval, 77
  - initpi, 77
  - initpiaero, 78
  - inputechoprescribedconditions, 78
  - loadintegration, 78
  - transferfollower, 79
  - updatefollower, 79
  - updatepi, 79
- prescribedcondition::distriload, 108
  - distr\_fun, 109
  - follower, 109
  - value, 109
- prescribedcondition::prescriinf, 131
  - dof, 132
  - follower, 132
  - id, 132
  - time\_fun\_no, 133
  - value, 133
  - value\_current, 133
- prod
  - globaldatafun, 44
- pt\_condition
  - ioaero, 66
- rad\_2\_deg
  - globaldatafun, 50
- rate
  - cputime, 10
- ReadEigenVec
  - gebtaero::utils, 33
- ReadFlexibilityFromDisp
  - gebtaero::utils, 33
- ReadFromPipe
  - gebtaero::utils, 33
- ReadLoadsFromPipe
  - gebtaero::utils, 33
- ReadModesFromPipe
  - gebtaero::utils, 34
- ReadNodesField
  - gebtaero::utils, 34
- RemoveFiles
  - gebtaero::utils, 34
- RemoveInputFile
  - gebtaero::InputFile::InputFile, 121
- RemoveMeshFiles
  - gebtaero::utils, 34
- Rho
  - gebtaero::InputFile::InputFile, 124
  - gebtaero::IsoMaterial::IsoMaterial, 127
  - gebtaero::OrthoMaterial::OrthoMaterial, 131



- rho
  - element, [28](#)
- Right
  - gebtaero::CompositeBox::CompositeBox, [96](#)
- rtbis
  - prepromodule, [73](#)
- RunFbdFile
  - gebtaero::utils, [34](#)
- RunFrdFile
  - gebtaero::utils, [35](#)
- RunInpFile
  - gebtaero::utils, [35](#)
- RunParaviewScript
  - gebtaero::utils, [35](#)
- RunUnvConv
  - gebtaero::utils, [35](#)
- runmod
  - globaldatafun, [50](#)
- SectionLength
  - gebtaero::WingSection::WingSection, [155](#)
- SetChord
  - gebtaero::WingSection::WingSection, [153](#)
- SetFlexibilityMatrixByBox
  - gebtaero::CrossSection::CrossSection, [104](#)
- SetFlexibilityMatrixByIsotropicValues
  - gebtaero::CrossSection::CrossSection, [105](#)
- SetFlexibilityMatrixByMesh
  - gebtaero::CrossSection::CrossSection, [105](#)
- SetFlexibilityMatrixByPlate
  - gebtaero::CrossSection::CrossSection, [105](#)
- SetFlexibilityMatrixByRectBeamValues
  - gebtaero::CrossSection::CrossSection, [106](#)
- SetMassMatrix
  - gebtaero::CrossSection::CrossSection, [106](#)
- SetMassMatrixByBox
  - gebtaero::CrossSection::CrossSection, [106](#)
- SetMassMatrixByMesh
  - gebtaero::CrossSection::CrossSection, [107](#)
- SetMassMatrixByPlate
  - gebtaero::CrossSection::CrossSection, [107](#)
- SetMassMatrixByRectBeamValues
  - gebtaero::CrossSection::CrossSection, [107](#)
- SetNumberOfElements
  - gebtaero::WingSection::WingSection, [153](#)
- SetParameterA
  - gebtaero::WingSection::WingSection, [154](#)
- SetSectionLength
  - gebtaero::WingSection::WingSection, [154](#)
- simu\_time
  - ioaero, [66](#)
- SimuEnd
  - gebtaero::InputFile::InputFile, [124](#)
- SimuStart
  - gebtaero::InputFile::InputFile, [124](#)
- sol\_mb
  - ioaero, [67](#)
- sol\_pt
  - ioaero, [67](#)
- SolveMumps.f90
  - linsearch, [175](#)
- solvemumps, [80](#)
  - ctcabph, [80](#)
  - extractelementvalues, [81](#)
  - extractsolution, [81](#)
  - ierr, [85](#)
  - insertelementvalues, [82](#)
  - linearsolutionmumps, [82](#)
  - mumps\_par, [85](#)
  - newtonraphsonmumps, [84](#)
- solver
  - globaldatafun, [50](#)
- start
  - cputime, [10](#)
- StaticLoads
  - gebtaero::Simulation::Simulation, [139](#)
- system, [85](#)
  - assemblejacobian, [86](#)
  - assemblerhs, [87](#)
  - coef, [88](#)
  - irn, [88](#)
  - jcn, [88](#)
  - kp\_cond, [89](#)
  - kp\_dof, [89](#)
  - kp\_follower, [89](#)
  - ne, [89](#)
  - pointfollowerj, [88](#)
  - x\_pt, [89](#)
- System.f90
  - assemblepointj, [176](#)
  - assemblepointrhs, [177](#)
- te
  - timefunctionmodule::timefunction, [142](#)
- TemporalDivergenceSpeed
  - gebtaero::Simulation::Simulation, [139](#)
- TemporalDynamic
  - gebtaero::Simulation::Simulation, [140](#)
- TemporalFlutterSpeed
  - gebtaero::Simulation::Simulation, [140](#)
- theta
  - element, [28](#)
- thetadot
  - element, [28](#)
- Thickness
  - gebtaero::CompositePly::CompositePly, [103](#)
- tic
  - cputime, [9](#)
- tilde
  - globaldatafun, [44](#)
- time\_fun\_no
  - prescribedcondition::prescriinf, [133](#)
- time\_function
  - ioaero, [67](#)
- time\_val
  - timefunctionmodule::timefunction, [143](#)
- TimeFunctions
  - gebtaero::InputFile::InputFile, [124](#)

- timefunctionmodule, 90
  - currentvalues, 90
  - gettimefunction, 90
  - inittf, 91
  - inputechotimefunctions, 91
- timefunctionmodule::timefunction, 141
  - entries, 142
  - fun\_type, 142
  - fun\_val, 142
  - phase\_val, 142
  - te, 142
  - time\_val, 143
  - ts, 143
- titleprint
  - globaldatafun, 44
- toc
  - cputime, 9
- tolerance
  - globaldatafun, 50
- TotThickness
  - gebtaero::CompositePlate::CompositePlate, 101
  - gebtaero::ExternalMesh::ExternalMesh, 114
- transferfollower
  - prescribedcondition, 79
- triad
  - internaldata::memberinf, 129
- ts
  - timefunctionmodule::timefunction, 143
- Twist
  - gebtaero::Frame::Frame, 116
- two\_divide\_dt
  - internaldata, 54
- u
  - element, 29
- ui
  - element, 29
- uidot
  - element, 29
- Up
  - gebtaero::CompositeBox::CompositeBox, 97
- updatefollower
  - prescribedcondition, 79
- updatepi
  - prescribedcondition, 79
- v\_root\_a0
  - ioaero, 67
- v\_root\_a\_tf
  - ioaero, 67
- value
  - prescribedcondition::distriload, 109
  - prescribedcondition::prescriinf, 133
- value\_current
  - prescribedcondition::prescriinf, 133
- vecb
  - globaldatafun, 44
- vecc
  - globaldatafun, 45
- vecd
  - globaldatafun, 45
- velocity\_str
  - ioaero, 68
- vi
  - element, 29
- Vinf
  - gebtaero::InputFile::InputFile, 124
- Width
  - gebtaero::CompositeBox::CompositeBox, 97
- wind
  - element, 29
- Wing
  - gebtaero::InputFile::InputFile, 125
  - gebtaero::Simulation::Simulation, 141
- WingRootPosition
  - gebtaero::Wing::Wing, 150
- WingSections
  - gebtaero::Wing::Wing, 150
- WriteInitFile
  - gebtaero::InputFile::InputFile, 121
- WriteInputFile
  - gebtaero::InputFile::InputFile, 121
- WriteParaviewScript
  - gebtaero::GebtPlot::GebtPlot, 118
- writeerror
  - globaldatafun, 45
- writeintvector
  - globaldatafun, 45
  - globaldatafun::writevec, 156
- writerealvector
  - globaldatafun, 46
  - globaldatafun::writevec, 156
- x0
  - gebtaero::ExternalMesh::ExternalMesh, 114
- x\_cg
  - element, 29
- x\_pt
  - system, 89
- Xcg
  - gebtaero::InputFile::InputFile, 125
- xyz\_pt1
  - internaldata, 54