

edos

0.1

Generated by Doxygen 1.8.6

Mon Nov 3 2014 16:03:17



# Contents

<b>1</b>	<b>Documentation.</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	General considerations about implementation. . . . .	1
1.2.1	Directory structure: . . . . .	2
1.2.1.1	Test directories: . . . . .	2
<b>2</b>	<b>Namespace Index</b>	<b>3</b>
2.1	Namespace List . . . . .	3
<b>3</b>	<b>Hierarchical Index</b>	<b>5</b>
3.1	Class Hierarchy . . . . .	5
<b>4</b>	<b>Class Index</b>	<b>7</b>
4.1	Class List . . . . .	7
<b>5</b>	<b>File Index</b>	<b>9</b>
5.1	File List . . . . .	9
<b>6</b>	<b>Namespace Documentation</b>	<b>11</b>
6.1	odes Namespace Reference . . . . .	11
6.1.1	Detailed Description . . . . .	12
6.1.2	Enumeration Type Documentation . . . . .	12
6.1.2.1	eventType . . . . .	12
6.1.3	Function Documentation . . . . .	12
6.1.3.1	dgbtrf_ . . . . .	12
6.1.3.2	dgbtrs_ . . . . .	12
6.1.3.3	dgehrd_ . . . . .	12
6.1.3.4	dgetrf_ . . . . .	12
6.1.3.5	dgetrs_ . . . . .	13
6.1.3.6	dlarnv_ . . . . .	13
6.1.3.7	dorghr_ . . . . .	13
6.1.3.8	zgbtrf_ . . . . .	13
6.1.3.9	zgbtrs_ . . . . .	13

6.1.3.10	zgetrf_ . . . . .	13
6.1.3.11	zgetrs_ . . . . .	13
<b>7</b>	<b>Class Documentation</b>	<b>15</b>
7.1	compat< full, Hessenberg > Struct Template Reference . . . . .	15
7.1.1	Member Function Documentation . . . . .	15
7.1.1.1	ct_assert . . . . .	15
7.2	odes::logger::event Struct Reference . . . . .	15
7.2.1	Constructor & Destructor Documentation . . . . .	15
7.2.1.1	event . . . . .	15
7.2.2	Member Data Documentation . . . . .	15
7.2.2.1	step . . . . .	16
7.2.2.2	TheEvent . . . . .	16
7.2.2.3	time . . . . .	16
7.2.2.4	value . . . . .	16
7.3	odes::fortranArray< n > Class Template Reference . . . . .	16
7.3.1	Detailed Description . . . . .	16
7.3.2	Constructor & Destructor Documentation . . . . .	17
7.3.2.1	fortranArray . . . . .	17
7.3.2.2	fortranArray . . . . .	17
7.3.2.3	~fortranArray . . . . .	17
7.3.3	Member Function Documentation . . . . .	17
7.3.3.1	addDiag . . . . .	17
7.3.3.2	equal_minus . . . . .	17
7.3.3.3	operator& . . . . .	17
7.3.3.4	operator() . . . . .	17
7.3.3.5	operator() . . . . .	18
7.3.3.6	print . . . . .	19
7.3.4	Member Data Documentation . . . . .	19
7.3.4.1	n2 . . . . .	19
7.3.4.2	x . . . . .	19
7.4	odes::fortranComplexArray< n > Class Template Reference . . . . .	19
7.4.1	Detailed Description . . . . .	20
7.4.2	Constructor & Destructor Documentation . . . . .	20
7.4.2.1	fortranComplexArray . . . . .	20
7.4.2.2	~fortranComplexArray . . . . .	20
7.4.3	Member Function Documentation . . . . .	20
7.4.3.1	Im . . . . .	20
7.4.3.2	operator& . . . . .	20
7.4.3.3	print . . . . .	20

7.4.3.4	Re	20
7.4.3.5	set	21
7.4.4	Member Data Documentation	21
7.4.4.1	d2	21
7.4.4.2	size	21
7.4.4.3	x	21
7.5	odes::fortranRectangularArray< n, kl, ku > Class Template Reference	21
7.5.1	Detailed Description	22
7.5.2	Constructor & Destructor Documentation	22
7.5.2.1	fortranRectangularArray	22
7.5.2.2	~fortranRectangularArray	22
7.5.3	Member Function Documentation	22
7.5.3.1	addDiag	22
7.5.3.2	equal_minus	22
7.5.3.3	operator&	22
7.5.3.4	operator()	22
7.5.3.5	operator()	23
7.5.3.6	print	23
7.5.4	Member Data Documentation	23
7.5.4.1	cc	23
7.5.4.2	klku	23
7.5.4.3	l1	23
7.5.4.4	ldab	23
7.5.4.5	size	23
7.5.4.6	x	23
7.6	odes::fortranRectangularComplexArray< n, kl, ku > Class Template Reference	24
7.6.1	Detailed Description	24
7.6.2	Constructor & Destructor Documentation	24
7.6.2.1	fortranRectangularComplexArray	24
7.6.2.2	~fortranRectangularComplexArray	24
7.6.3	Member Function Documentation	25
7.6.3.1	Im	25
7.6.3.2	operator&	25
7.6.3.3	print	25
7.6.3.4	Re	25
7.6.3.5	set	25
7.6.4	Member Data Documentation	25
7.6.4.1	cc	25
7.6.4.2	klku	25
7.6.4.3	l1	25

7.6.4.4	ldab	26
7.6.4.5	size	26
7.6.4.6	x	26
7.7	odes::fortranVector Class Reference	26
7.7.1	Detailed Description	27
7.7.2	Constructor & Destructor Documentation	27
7.7.2.1	fortranVector	27
7.7.2.2	fortranVector	27
7.7.2.3	fortranVector	27
7.7.2.4	~fortranVector	27
7.7.3	Member Function Documentation	27
7.7.3.1	get_size	27
7.7.3.2	operator&	27
7.7.3.3	operator()	27
7.7.3.4	operator()	28
7.7.3.5	operator+=	28
7.7.3.6	operator=	28
7.7.3.7	print	28
7.7.3.8	setsize	28
7.7.4	Friends And Related Function Documentation	28
7.7.4.1	fortranArray	28
7.7.4.2	fortranVectorF	28
7.7.5	Member Data Documentation	28
7.7.5.1	deletable	29
7.7.5.2	size	29
7.7.5.3	x	29
7.8	odes::fortranVectorF< n > Class Template Reference	29
7.8.1	Detailed Description	30
7.8.2	Constructor & Destructor Documentation	30
7.8.2.1	fortranVectorF	30
7.8.2.2	~fortranVectorF	30
7.8.3	Member Function Documentation	30
7.8.3.1	a_x_plus_y	30
7.8.3.2	a_Y	30
7.8.3.3	get_size	31
7.8.3.4	lc2	31
7.8.3.5	lc3	31
7.8.3.6	lc4	31
7.8.3.7	lc5	32
7.8.3.8	operator&	32

7.8.3.9	<a href="#">operator()</a>	32
7.8.3.10	<a href="#">operator()</a>	32
7.8.3.11	<a href="#">operator+=</a>	32
7.8.3.12	<a href="#">operator+=</a>	33
7.8.3.13	<a href="#">operator=</a>	33
7.8.3.14	<a href="#">operator=</a>	33
7.8.3.15	<a href="#">operator=</a>	33
7.8.3.16	<a href="#">print</a>	33
7.8.3.17	<a href="#">put</a>	33
7.8.3.18	<a href="#">setsize</a>	34
7.8.3.19	<a href="#">setsum2</a>	34
7.8.3.20	<a href="#">sum</a>	34
7.8.3.21	<a href="#">sum</a>	34
7.8.3.22	<a href="#">switchadr</a>	34
7.8.3.23	<a href="#">x_lc2</a>	35
7.8.3.24	<a href="#">x_lc3</a>	35
7.8.3.25	<a href="#">x_lc4</a>	35
7.8.4	<a href="#">Member Data Documentation</a>	35
7.8.4.1	<a href="#">deletable</a>	36
7.8.4.2	<a href="#">size</a>	36
7.8.4.3	<a href="#">x</a>	36
7.8.4.4	<a href="#">y</a>	36
7.9	<a href="#">GenericException Class Reference</a>	36
7.9.1	<a href="#">Detailed Description</a>	37
7.9.2	<a href="#">Constructor &amp; Destructor Documentation</a>	37
7.9.2.1	<a href="#">GenericException</a>	37
7.9.2.2	<a href="#">GenericException</a>	37
7.9.2.3	<a href="#">GenericException</a>	37
7.9.2.4	<a href="#">GenericException</a>	37
7.9.2.5	<a href="#">GenericException</a>	37
7.9.2.6	<a href="#">GenericException</a>	37
7.9.2.7	<a href="#">GenericException</a>	37
7.9.2.8	<a href="#">GenericException</a>	37
7.9.2.9	<a href="#">GenericException</a>	37
7.9.2.10	<a href="#">GenericException</a>	38
7.9.2.11	<a href="#">GenericException</a>	38
7.10	<a href="#">odes::logger Class Reference</a>	38
7.10.1	<a href="#">Detailed Description</a>	38
7.10.2	<a href="#">Constructor &amp; Destructor Documentation</a>	38
7.10.2.1	<a href="#">logger</a>	38

7.10.2.2	<code>~logger</code>	39
7.10.3	Member Function Documentation	39
7.10.3.1	<code>clear</code>	39
7.10.3.2	<code>print</code>	39
7.10.3.3	<code>put</code>	39
7.10.4	Member Data Documentation	39
7.10.4.1	<code>L</code>	39
7.11	<code>odes::Matrices&lt; full, Hessengerg, n, nsub, nsup &gt; Class Template Reference</code>	39
7.11.1	Detailed Description	40
7.11.2	Member Typedef Documentation	40
7.11.2.1	<code>MatrixComplex</code>	40
7.11.2.2	<code>MatrixReal</code>	40
7.11.3	Constructor & Destructor Documentation	40
7.11.3.1	<code>Matrices</code>	40
7.11.4	Member Function Documentation	40
7.11.4.1	<code>decomc</code>	40
7.11.4.2	<code>decomr</code>	41
7.11.4.3	<code>lbegin</code>	42
7.11.4.4	<code>lend</code>	42
7.11.4.5	<code>solvecomplex</code>	42
7.11.4.6	<code>solverreal</code>	42
7.11.5	Member Data Documentation	42
7.11.5.1	<code>calhes</code>	42
7.11.5.2	<code>E1</code>	42
7.11.5.3	<code>E2R</code>	42
7.11.5.4	<code>ipivc</code>	42
7.11.5.5	<code>ipivr</code>	43
7.11.5.6	<code>Z2N</code>	43
7.12	<code>odes::Matrices&lt; false, false, n, nsub, nsup &gt; Class Template Reference</code>	43
7.12.1	Detailed Description	44
7.12.2	Member Typedef Documentation	44
7.12.2.1	<code>MatrixComplex</code>	44
7.12.2.2	<code>MatrixReal</code>	44
7.12.3	Constructor & Destructor Documentation	44
7.12.3.1	<code>Matrices</code>	44
7.12.4	Member Function Documentation	44
7.12.4.1	<code>decomc</code>	44
7.12.4.2	<code>decomr</code>	44
7.12.4.3	<code>lbegin</code>	44
7.12.4.4	<code>lend</code>	44



7.12.4.5	<a href="#">solvecomplex</a>	45
7.12.4.6	<a href="#">solverreal</a>	45
7.12.5	<a href="#">Member Data Documentation</a>	45
7.12.5.1	<a href="#">calhes</a>	45
7.12.5.2	<a href="#">E1</a>	45
7.12.5.3	<a href="#">E2R</a>	45
7.12.5.4	<a href="#">ipivc</a>	45
7.12.5.5	<a href="#">ipivr</a>	45
7.12.5.6	<a href="#">ldab</a>	45
7.12.5.7	<a href="#">Z2N</a>	45
7.13	<a href="#">odes::Matrices&lt; false, true, n, nsub, nsup &gt; Class Template Reference</a>	45
7.13.1	<a href="#">Detailed Description</a>	46
7.13.2	<a href="#">Member Typedef Documentation</a>	46
7.13.2.1	<a href="#">MatrixComplex</a>	46
7.13.2.2	<a href="#">MatrixReal</a>	46
7.13.3	<a href="#">Constructor &amp; Destructor Documentation</a>	46
7.13.3.1	<a href="#">Matrices</a>	46
7.13.4	<a href="#">Member Function Documentation</a>	46
7.13.4.1	<a href="#">decomc</a>	46
7.13.4.2	<a href="#">decomr</a>	46
7.13.4.3	<a href="#">lbegin</a>	46
7.13.4.4	<a href="#">lend</a>	46
7.13.4.5	<a href="#">solvecomplex</a>	46
7.13.4.6	<a href="#">solverreal</a>	46
7.13.5	<a href="#">Member Data Documentation</a>	47
7.13.5.1	<a href="#">calhes</a>	47
7.14	<a href="#">odes::Matrices&lt; true, true, n, nsub, nsup &gt; Class Template Reference</a>	47
7.14.1	<a href="#">Detailed Description</a>	48
7.14.2	<a href="#">Member Typedef Documentation</a>	48
7.14.2.1	<a href="#">MatrixComplex</a>	48
7.14.2.2	<a href="#">MatrixReal</a>	48
7.14.3	<a href="#">Constructor &amp; Destructor Documentation</a>	48
7.14.3.1	<a href="#">Matrices</a>	48
7.14.3.2	<a href="#">~Matrices</a>	48
7.14.4	<a href="#">Member Function Documentation</a>	48
7.14.4.1	<a href="#">decomc</a>	48
7.14.4.2	<a href="#">decomr</a>	48
7.14.4.3	<a href="#">lbegin</a>	48
7.14.4.4	<a href="#">lend</a>	48
7.14.4.5	<a href="#">solvecomplex</a>	48

7.14.4.6	<a href="#">solvereal</a>	48
7.14.5	<a href="#">Member Data Documentation</a>	48
7.14.5.1	<a href="#">calhes</a>	48
7.14.5.2	<a href="#">Ci</a>	49
7.14.5.3	<a href="#">Cr</a>	49
7.14.5.4	<a href="#">E1</a>	49
7.14.5.5	<a href="#">E2I</a>	49
7.14.5.6	<a href="#">E2R</a>	49
7.14.5.7	<a href="#">Fi</a>	49
7.14.5.8	<a href="#">Fr</a>	49
7.14.5.9	<a href="#">ipi</a>	49
7.14.5.10	<a href="#">ipr</a>	49
7.14.5.11	<a href="#">lwork</a>	49
7.14.5.12	<a href="#">tau</a>	49
7.14.5.13	<a href="#">work</a>	49
7.15	<a href="#">odes::Matrixtype&lt; n, nsub, nsup &gt; Struct Template Reference</a>	49
7.15.1	<a href="#">Member Typedef Documentation</a>	49
7.15.1.1	<a href="#">Matrix</a>	49
7.15.2	<a href="#">Member Data Documentation</a>	49
7.15.2.1	<a href="#">Full</a>	50
7.16	<a href="#">odes::Radau5cc&lt; Fonct &gt; Class Template Reference</a>	50
7.16.1	<a href="#">Detailed Description</a>	53
7.16.2	<a href="#">Member Typedef Documentation</a>	54
7.16.2.1	<a href="#">MatrixComplex</a>	54
7.16.2.2	<a href="#">MatrixReal</a>	54
7.16.3	<a href="#">Member Enumeration Documentation</a>	54
7.16.3.1	<a href="#">OutNewtonLoop</a>	54
7.16.4	<a href="#">Constructor &amp; Destructor Documentation</a>	54
7.16.4.1	<a href="#">Radau5cc</a>	54
7.16.4.2	<a href="#">~Radau5cc</a>	54
7.16.5	<a href="#">Member Function Documentation</a>	54
7.16.5.1	<a href="#">ct_assert</a>	54
7.16.5.2	<a href="#">decomc</a>	54
7.16.5.3	<a href="#">decomr</a>	55
7.16.5.4	<a href="#">doNewtonMatrices</a>	56
7.16.5.5	<a href="#">doScal</a>	56
7.16.5.6	<a href="#">estrad</a>	56
7.16.5.7	<a href="#">getfirstAcceptedStep</a>	56
7.16.5.8	<a href="#">getLastTimeStep</a>	56
7.16.5.9	<a href="#">getNaccpt</a>	56

7.16.5.10	<a href="#">getNdec</a>	56
7.16.5.11	<a href="#">getNewt</a>	56
7.16.5.12	<a href="#">getNfncCalled</a>	57
7.16.5.13	<a href="#">getNJac</a>	57
7.16.5.14	<a href="#">getNrejct</a>	57
7.16.5.15	<a href="#">getNstep</a>	57
7.16.5.16	<a href="#">lbegin</a>	57
7.16.5.17	<a href="#">lend</a>	57
7.16.5.18	<a href="#">Jacobian</a>	57
7.16.5.19	<a href="#">Jacobian</a>	57
7.16.5.20	<a href="#">Newton</a>	57
7.16.5.21	<a href="#">NewtonIterationTest</a>	58
7.16.5.22	<a href="#">operator()</a>	58
7.16.5.23	<a href="#">performOnlyOneStep</a>	58
7.16.5.24	<a href="#">rhs</a>	58
7.16.5.25	<a href="#">setFacrFacI</a>	58
7.16.5.26	<a href="#">setFnewt</a>	59
7.16.5.27	<a href="#">setGustafssonTest</a>	59
7.16.5.28	<a href="#">seth</a>	59
7.16.5.29	<a href="#">setMaxIterationsNewton</a>	59
7.16.5.30	<a href="#">setNmax</a>	59
7.16.5.31	<a href="#">setRecomputeJacobianTreshold</a>	59
7.16.5.32	<a href="#">setSafe</a>	59
7.16.5.33	<a href="#">setTestPolicy</a>	60
7.16.5.34	<a href="#">slvrad</a>	60
7.16.5.35	<a href="#">solvecomplex</a>	60
7.16.5.36	<a href="#">solvereal</a>	60
7.16.6	<a href="#">Member Data Documentation</a>	60
7.16.6.1	<a href="#">ALPH</a>	60
7.16.6.2	<a href="#">atol</a>	60
7.16.6.3	<a href="#">BETA</a>	60
7.16.6.4	<a href="#">C1</a>	60
7.16.6.5	<a href="#">C1M1</a>	60
7.16.6.6	<a href="#">C1MC2</a>	61
7.16.6.7	<a href="#">C2</a>	61
7.16.6.8	<a href="#">C2M1</a>	61
7.16.6.9	<a href="#">calhes</a>	61
7.16.6.10	<a href="#">caljac</a>	61
7.16.6.11	<a href="#">cfac</a>	61
7.16.6.12	<a href="#">CNO</a>	61

7.16.6.13 ComputeJacobianNumerically . . . . .	61
7.16.6.14 CONT1 . . . . .	61
7.16.6.15 DD1 . . . . .	61
7.16.6.16 DD2 . . . . .	61
7.16.6.17 DD3 . . . . .	61
7.16.6.18 dyno . . . . .	61
7.16.6.19 dynold . . . . .	61
7.16.6.20 erracc . . . . .	61
7.16.6.21 F . . . . .	61
7.16.6.22 F1 . . . . .	61
7.16.6.23 F2 . . . . .	61
7.16.6.24 F3 . . . . .	61
7.16.6.25 faccon . . . . .	61
7.16.6.26 facI . . . . .	61
7.16.6.27 facr . . . . .	61
7.16.6.28 first . . . . .	61
7.16.6.29 firstAcceptedStep . . . . .	61
7.16.6.30 fnewt . . . . .	61
7.16.6.31 gustafssonTest . . . . .	62
7.16.6.32 h . . . . .	62
7.16.6.33 hacc . . . . .	62
7.16.6.34 Hessenberg . . . . .	62
7.16.6.35 hhfac . . . . .	62
7.16.6.36 hmax . . . . .	62
7.16.6.37 hmaxn . . . . .	62
7.16.6.38 hold . . . . .	62
7.16.6.39 hopt . . . . .	62
7.16.6.40 Jac . . . . .	62
7.16.6.41 last . . . . .	62
7.16.6.42 MatrixFull . . . . .	62
7.16.6.43 n . . . . .	62
7.16.6.44 naccpt . . . . .	62
7.16.6.45 ndec . . . . .	62
7.16.6.46 newt . . . . .	62
7.16.6.47 nfonccalled . . . . .	62
7.16.6.48 ninf . . . . .	62
7.16.6.49 nit . . . . .	62
7.16.6.50 njac . . . . .	62
7.16.6.51 nmax . . . . .	62
7.16.6.52 nreject . . . . .	62

7.16.6.53 nstep . . . . .	62
7.16.6.54 nsup . . . . .	62
7.16.6.55 onlyOneStep . . . . .	62
7.16.6.56 posneg . . . . .	62
7.16.6.57 quot1 . . . . .	62
7.16.6.58 quot2 . . . . .	63
7.16.6.59 reject . . . . .	63
7.16.6.60 rtol . . . . .	63
7.16.6.61 safe . . . . .	63
7.16.6.62 sameTestValue . . . . .	63
7.16.6.63 save1 . . . . .	63
7.16.6.64 save2 . . . . .	63
7.16.6.65 save3 . . . . .	63
7.16.6.66 scal . . . . .	63
7.16.6.67 SQ6 . . . . .	63
7.16.6.68 startn . . . . .	63
7.16.6.69 T11 . . . . .	63
7.16.6.70 T12 . . . . .	63
7.16.6.71 T13 . . . . .	63
7.16.6.72 T21 . . . . .	63
7.16.6.73 T22 . . . . .	63
7.16.6.74 T23 . . . . .	63
7.16.6.75 T31 . . . . .	63
7.16.6.76 thet . . . . .	63
7.16.6.77 theta . . . . .	63
7.16.6.78 thqold . . . . .	63
7.16.6.79 TI11 . . . . .	63
7.16.6.80 TI12 . . . . .	63
7.16.6.81 TI13 . . . . .	63
7.16.6.82 TI21 . . . . .	63
7.16.6.83 TI22 . . . . .	63
7.16.6.84 TI23 . . . . .	63
7.16.6.85 TI31 . . . . .	63
7.16.6.86 TI32 . . . . .	64
7.16.6.87 TI33 . . . . .	64
7.16.6.88 tolst . . . . .	64
7.16.6.89 U1 . . . . .	64
7.16.6.90 ound . . . . .	64
7.16.6.91 xend . . . . .	64
7.16.6.92 xold . . . . .	64

7.16.6.93 xph . . . . .	64
7.16.6.94 Z1 . . . . .	64
7.16.6.95 Z1T . . . . .	64
7.16.6.96 Z2 . . . . .	64
7.16.6.97 Z2T . . . . .	64
7.16.6.98 Z3 . . . . .	64
7.16.6.99 Z3T . . . . .	64
<b>8 File Documentation</b>	<b>65</b>
8.1 common/include/AllocateDestroyVector.hpp File Reference . . . . .	65
8.1.1 Macro Definition Documentation . . . . .	66
8.1.1.1 ASSUME_ALIGNED . . . . .	66
8.1.2 Function Documentation . . . . .	66
8.1.2.1 allocDoubleArray . . . . .	66
8.1.2.2 destroyDoubleArray . . . . .	66
8.2 common/include/compat.hpp File Reference . . . . .	66
8.2.1 Macro Definition Documentation . . . . .	67
8.2.1.1 ASSERT_CONCAT . . . . .	67
8.2.1.2 ASSERT_CONCAT_ . . . . .	67
8.2.1.3 ct_assert . . . . .	67
8.3 common/include/fortranArray.hpp File Reference . . . . .	67
8.4 common/include/fortranComplexArray.hpp File Reference . . . . .	68
8.5 common/include/fortranRectangularArray.hpp File Reference . . . . .	69
8.5.1 Macro Definition Documentation . . . . .	70
8.5.1.1 MAX . . . . .	70
8.5.1.2 MIN . . . . .	70
8.6 common/include/fortranRectangularComplexArray.hpp File Reference . . . . .	70
8.6.1 Macro Definition Documentation . . . . .	72
8.6.1.1 MAX . . . . .	72
8.6.1.2 MIN . . . . .	72
8.7 common/include/fortranVector.hpp File Reference . . . . .	72
8.8 common/include/GenericException.hpp File Reference . . . . .	73
8.9 common/include/Hessenberg.hpp File Reference . . . . .	74
8.9.1 Macro Definition Documentation . . . . .	75
8.9.1.1 ABS . . . . .	75
8.9.1.2 MIN . . . . .	75
8.9.2 Function Documentation . . . . .	75
8.9.2.1 dech . . . . .	75
8.9.2.2 dehc . . . . .	75
8.9.2.3 householder . . . . .	76

8.9.2.4	solh . . . . .	76
8.9.2.5	solhc . . . . .	76
8.10	common/include/lvdep.hpp File Reference . . . . .	77
8.11	common/include/logger.hpp File Reference . . . . .	77
8.12	common/include/MacrosForCompilers.hpp File Reference . . . . .	78
8.12.1	Macro Definition Documentation . . . . .	78
8.12.1.1	BLOCK_FACTOR . . . . .	78
8.13	common/include/Matrices.hpp File Reference . . . . .	78
8.13.1	Macro Definition Documentation . . . . .	80
8.13.1.1	MAX . . . . .	80
8.13.1.2	MIN . . . . .	80
8.14	common/include/MatrixType.hpp File Reference . . . . .	80
8.15	common/include/protos_lapack.hpp File Reference . . . . .	81
8.16	Mainpage/mainpage.dox File Reference . . . . .	81
8.17	Radau5-NG/include/Radau5cc.hpp File Reference . . . . .	82
8.17.1	Macro Definition Documentation . . . . .	82
8.17.1.1	ABS . . . . .	82
8.17.1.2	MAX . . . . .	82
8.17.1.3	MIN . . . . .	82
8.17.1.4	SIGN . . . . .	82
<b>Index</b>		<b>83</b>





# Chapter 1

## Documentation.

### 1.1 Introduction

The source directories Radau5-NG, Rock4-NG, Rock4-L, Rodas-NG, SDIRKL and SymplecticRK contains sources of different Runge-Kutta methods written in C++.

**Radau5-NG** is a rewriting of Hairer and Wanner code, optimized, but with some restrictions (see documentation).

**Rock4-NG** is a rewriting of A. Abdulle code, avoiding vector copies.

**Rock4-L** is an adaptation of Rock4 method to linear (actually affine) problems  $du/dt = Au + B$ , with  $A$  linear and  $B$  independent of  $t$ .

**Rodas-NG** is a rewriting of Hairer and Wanner code, optimized, but with some restrictions (see documentation).

**SDIRKL** is a generic implementation of SDIRK methods in the linear (affine) case. Thus, linear systems must be solved.

**SymplecticRK** is the (the family of) symplectic Gauss methods.

See documentation of individual programs.

### 1.2 General considerations about implementation.

All is written in C++, and some program must be used with recent compilers (c++-11 norm). Everything has been tested with g++, icc and clang compilers.

This is a library of templates, no intermediate binary is created.

For all these programs, a class must be created which describes the program to be solved.

### 1.2.1 Directory structure:

**common/include** contains all the classes, methods and functions common to the different programs.

for each program, the corresponding directory contains an **include** directory, which stores the different classes of the program and **test** directories.

#### 1.2.1.1 Test directories:

Each **test** directory contains a main program, all the necessary classes, and a cmake file. To compile and test the program:

Adapt CMake file if necessary.

**cd Build** directory

**cmake ..** (possibly with an option to determine the compiler you want to use); see the Cmake files provided.

**make** a **./run** file is created.

**run** to execute the program.

## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">odes</a>	Compute all matrix operations . . . . .	<a href="#">11</a>
----------------------	---	--------------------



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

compat< full, Hessenberg > . . . . .	15
compat<(Fonct::n-Fonct::nsub)==1 &&(Fonct::n-Fonct::nsup)==1, Fonct::Hessenberg > . . . . .	15
odes::Radau5cc< Fonct > . . . . .	50
odes::logger::event . . . . .	15
odes::fortranArray< n > . . . . .	16
odes::fortranComplexArray< n > . . . . .	19
odes::fortranRectangularArray< n, kl, ku > . . . . .	21
odes::fortranRectangularArray< n, nsub, nsup > . . . . .	21
odes::fortranRectangularComplexArray< n, kl, ku > . . . . .	24
odes::fortranRectangularComplexArray< n, nsub, nsup > . . . . .	24
odes::fortranVector . . . . .	26
odes::fortranVectorF< 2 *n > . . . . .	29
odes::fortranVectorF< n > . . . . .	29
GenericException . . . . .	36
odes::logger . . . . .	38
odes::Matrices< full, Hessengerg, n, nsub, nsup > . . . . .	39
odes::Matrices< false, false, n, nsub, nsup > . . . . .	43
odes::Matrices< false, true, n, nsub, nsup > . . . . .	45
odes::Matrices< MatrixFull, Hessenberg, n, ninf, nsup > . . . . .	39
odes::Matrices< true, true, n, nsub, nsup > . . . . .	47
odes::Matrices<(Fonct::n-Fonct::nsub)==1 &&(Fonct::n-Fonct::nsup)==1, Fonct::Hessenberg, Fonct::n, Fonct::nsub, Fonct::nsup > . . . . .	39
odes::Radau5cc< Fonct > . . . . .	50
odes::Matrixtype< n, nsub, nsup > . . . . .	49



## Chapter 4

# Class Index

### 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">compat&lt; full, Hessenberg &gt;</a>	15
<a href="#">odes::logger::event</a>	15
<a href="#">odes::fortranArray&lt; n &gt;</a>	
Array of double	16
<a href="#">odes::fortranComplexArray&lt; n &gt;</a>	
Array of complex.	19
<a href="#">odes::fortranRectangularArray&lt; n, kl, ku &gt;</a>	
Banded Array of double	21
<a href="#">odes::fortranRectangularComplexArray&lt; n, kl, ku &gt;</a>	
Banded Array of complex.	24
<a href="#">odes::fortranVector</a>	
Vector class (of doubles)	26
<a href="#">odes::fortranVectorF&lt; n &gt;</a>	
Vector class (of doubles)	29
<a href="#">GenericException</a>	36
<a href="#">odes::logger</a>	
For logging events	38
<a href="#">odes::Matrices&lt; full, Hessengerg, n, nsub, nsup &gt;</a>	
Full matrices	39
<a href="#">odes::Matrices&lt; false, false, n, nsub, nsup &gt;</a>	43
<a href="#">odes::Matrices&lt; false, true, n, nsub, nsup &gt;</a>	
Banded matrices, Hessenberg=true: error	45
<a href="#">odes::Matrices&lt; true, true, n, nsub, nsup &gt;</a>	
Full matrices, Hessenberg=true	47
<a href="#">odes::Matrixtype&lt; n, nsub, nsup &gt;</a>	49
<a href="#">odes::Radau5cc&lt; Fonct &gt;</a>	
A C++ implementation of Radau5	50





## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

common/include/ <a href="#">AllocateDestroyVector.hpp</a> . . . . .	65
common/include/ <a href="#">compat.hpp</a> . . . . .	66
common/include/ <a href="#">fortranArray.hpp</a> . . . . .	67
common/include/ <a href="#">fortranComplexArray.hpp</a> . . . . .	68
common/include/ <a href="#">fortranRectangularArray.hpp</a> . . . . .	69
common/include/ <a href="#">fortranRectangularComplexArray.hpp</a> . . . . .	70
common/include/ <a href="#">fortranVector.hpp</a> . . . . .	72
common/include/ <a href="#">GenericException.hpp</a> . . . . .	73
common/include/ <a href="#">Hessenberg.hpp</a> . . . . .	74
common/include/ <a href="#">lvdep.hpp</a> . . . . .	77
common/include/ <a href="#">logger.hpp</a> . . . . .	77
common/include/ <a href="#">MacrosForCompilers.hpp</a> . . . . .	78
common/include/ <a href="#">Matrices.hpp</a> . . . . .	78
common/include/ <a href="#">MatrixType.hpp</a> . . . . .	80
common/include/ <a href="#">protos_lapack.hpp</a> . . . . .	81
Radau5-NG/include/ <a href="#">Radau5cc.hpp</a> . . . . .	82



## Chapter 6

# Namespace Documentation

### 6.1 odes Namespace Reference

Compute all matrix operations.

#### Classes

- class [fortranArray](#)  
*Array of double.*
- class [fortranComplexArray](#)  
*Array of complex..*
- class [fortranRectangularArray](#)  
*Banded Array of double.*
- class [fortranRectangularComplexArray](#)  
*Banded Array of complex..*
- class [fortranVector](#)  
*Vector class (of doubles).*
- class [fortranVectorF](#)  
*Vector class (of doubles).*
- class [logger](#)  
*For logging events.*
- class [Matrices](#)  
*full matrices.*
- class [Matrices< false, false, n, nsub, nsup >](#)
- class [Matrices< true, true, n, nsub, nsup >](#)  
*Full matrices, Hessenberg=true.*
- class [Matrices< false, true, n, nsub, nsup >](#)  
*Banded matrices, Hessenberg=true: error.*
- struct [Matrixtype](#)
- class [Radau5cc](#)  
*A C++ implementation of Radau5.*

#### Enumerations

- enum [eventType](#) {  
    [success](#) =0, [rejectedStep](#), [NewtonFailed](#), [NewtonWillNotConverge](#),  
    [changedH](#), [all](#) }

## Functions

- void `dgetrf_` (int \*n, int \*m, double \*a, int \*lda, int \*ipiv, int \*info)  
*prototypes for lapack routines.*
- void `zgetrf_` (int \*n, int \*m, double \*a, int \*lda, int \*ipiv, int \*info)
- void `dgetrs_` (const char \*s, int \*N, int \*NRHS, double \*A, int \*LDA, int \*IPIV, double \*B, int \*LDB, int \*INFO)
- void `zgetrs_` (const char \*s, int \*N, int \*NRHS, double \*A, int \*LDA, int \*IPIV, double \*B, int \*LDB, int \*INFO)
- void `dgbtrf_` (int \*n, int \*m, int \*k1, int \*k2, double \*a, int \*lda, int \*ipiv, int \*info)
- void `zgbtrf_` (int \*n, int \*m, int \*k1, int \*k2, double \*a, int \*lda, int \*ipiv, int \*info)
- void `dgbtrs_` (const char \*s, int \*N, int \*k1, int \*k2, int \*NRHS, double \*A, int \*LDA, int \*IPIV, double \*B, int \*LDB, int \*INFO)
- void `zgbtrs_` (const char \*s, int \*N, int \*k1, int \*k2, int \*NRHS, double \*A, int \*LDA, int \*IPIV, double \*B, int \*LDB, int \*INFO)
- void `dlarnv_` (int \*idist, int iseed[], int \*n, double \*x)
- void `dgehrd_` (int \*n, int \*ilo, int \*ihi, double \*a, int \*lda, double tau[], double work[], int \*lwork, int \*info)
- void `dorghr_` (int \*n, int \*ilo, int \*ihi, double \*a, int \*lda, double tau[], double work[], int \*lwork, int \*info)

### 6.1.1 Detailed Description

Compute all matrix operations. determine the type of Jacobian matrices.

Compute all matrix operations: -compute Jacobian matrix -..... Factorisations.

- solve systems. Template for full matrix, with specializations for banded matrices and full matrix, transformed in Hessenberg form. Trying to instantiate a banded matrix matrix in Hessenberg form is impossible and should be captured with "compat.hpp"

Helper class, used at compile time to determine the type of Jacobian matrices.

### 6.1.2 Enumeration Type Documentation

#### 6.1.2.1 enum odes::eventType

Enumerator

**success**  
**rejectedStep**  
**NewtonFailed**  
**NewtonWillNotConverge**  
**changedH**  
**all**

### 6.1.3 Function Documentation

6.1.3.1 void odes::dgbtrf\_ ( int \* n, int \* m, int \* k1, int \* k2, double \* a, int \* lda, int \* ipiv, int \* info )

6.1.3.2 void odes::dgbtrs\_ ( const char \* s, int \* N, int \* k1, int \* k2, int \* NRHS, double \* A, int \* LDA, int \* IPIV, double \* B, int \* LDB, int \* INFO )

6.1.3.3 void odes::dgehrd\_ ( int \* n, int \* ilo, int \* ihi, double \* a, int \* lda, double tau[], double work[], int \* lwork, int \* info )

6.1.3.4 void odes::dgetrf\_ ( int \* n, int \* m, double \* a, int \* lda, int \* ipiv, int \* info )

prototypes for lapack routines.

- 6.1.3.5 void odes::dgetrs\_( const char \* *s*, int \* *N*, int \* *NRHS*, double \* *A*, int \* *LDA*, int \* *IPIV*, double \* *B*, int \* *LDB*, int \* *INFO* )
- 6.1.3.6 void odes::dlarnv\_( int \* *idist*, int *iseed*[], int \* *n*, double \* *x* )
- 6.1.3.7 void odes::dorghr\_( int \* *n*, int \* *ilo*, int \* *ihi*, double \* *a*, int \* *lda*, double *tau*[], double *work*[], int \* *lwork*, int \* *info* )
- 6.1.3.8 void odes::zgbtrf\_( int \* *n*, int \* *m*, int \* *k1*, int \* *k2*, double \* *a*, int \* *lda*, int \* *ipiv*, int \* *info* )
- 6.1.3.9 void odes::zgbtrs\_( const char \* *s*, int \* *N*, int \* *k1*, int \* *k2*, int \* *NRHS*, double \* *A*, int \* *LDA*, int \* *IPIV*, double \* *B*, int \* *LDB*, int \* *INFO* )
- 6.1.3.10 void odes::zgetrf\_( int \* *n*, int \* *m*, double \* *a*, int \* *lda*, int \* *ipiv*, int \* *info* )
- 6.1.3.11 void odes::zgetrs\_( const char \* *s*, int \* *N*, int \* *NRHS*, double \* *A*, int \* *LDA*, int \* *IPIV*, double \* *B*, int \* *LDB*, int \* *INFO* )



## Chapter 7

# Class Documentation

### 7.1 compat< full, Hessenberg > Struct Template Reference

```
#include <compat.hpp>
```

#### Public Member Functions

- [ct\\_assert](#) (full||(!full &&!Hessenberg))

#### 7.1.1 Member Function Documentation

7.1.1.1 `template<bool full, bool Hessenberg> compat< full, Hessenberg >::ct_assert ( full|| !full &&!Hessenberg )`

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

- common/include/[compat.hpp](#)

### 7.2 odes::logger::event Struct Reference

#### Public Member Functions

- [event](#) (double \_time, int \_step, [eventType](#) \_TheEvent, double \_value=0.0)

#### Public Attributes

- double [time](#)
- double [value](#)
- int [step](#)
- [eventType](#) [TheEvent](#)

#### 7.2.1 Constructor & Destructor Documentation

7.2.1.1 `odes::logger::event::event ( double _time, int _step, eventType _TheEvent, double _value = 0.0 ) [inline]`

#### 7.2.2 Member Data Documentation

7.2.2.1 `int odes::logger::event::step`

7.2.2.2 `eventType odes::logger::event::TheEvent`

7.2.2.3 `double odes::logger::event::time`

7.2.2.4 `double odes::logger::event::value`

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

- `common/include/logger.hpp`

## 7.3 `odes::fortranArray< n >` Class Template Reference

Array of double.

```
#include <fortranArray.hpp>
```

### Public Member Functions

- `fortranArray ()`  
*constructor*
- `fortranArray (const fortranArray< n > &AA)`  
*constructor (copy).*
- `~fortranArray ()`  
*destructor*
- `double operator() (int i, int j) const`
- `double & operator() (int i, int j)`
- `void equal_minus (fortranArray< n > &X)`
- `void addDiag (double v)`
- `double * operator& ()`  
*return adress of double array.*
- `void print (std::string s="") const`  
*print*

### Private Attributes

- `double * x`

### Static Private Attributes

- `static const int n2 =n*n`

#### 7.3.1 Detailed Description

```
template<int n>class odes::fortranArray< n >
```

Array of double.

Array of double, of fixed size n.n, fortran indexing.



### 7.3.2 Constructor & Destructor Documentation

7.3.2.1 `template<int n> odes::fortranArray< n >::fortranArray ( ) [inline]`

constructor

7.3.2.2 `template<int n> odes::fortranArray< n >::fortranArray ( const fortranArray< n > & AA ) [inline]`

constructor (copy).

define the ivdep pragma for different compilers.

7.3.2.3 `template<int n> odes::fortranArray< n >::~~fortranArray ( ) [inline]`

destructor

### 7.3.3 Member Function Documentation

7.3.3.1 `template<int n> void odes::fortranArray< n >::addDiag ( double v ) [inline]`

add a value to diagonal.

Parameters

<i>v</i>	the value.
----------	------------

define the ivdep pragma for different compilers.

7.3.3.2 `template<int n> void odes::fortranArray< n >::equal_minus ( fortranArray< n > & X ) [inline]`

this = -array.

Parameters

<i>X</i>	the array.
----------	------------

define the ivdep pragma for different compilers.

7.3.3.3 `template<int n> double* odes::fortranArray< n >::operator& ( ) [inline]`

return adress of double array.

7.3.3.4 `template<int n> double odes::fortranArray< n >::operator() ( int i, int j ) const [inline]`

indexing

Parameters

<i>i</i>	
<i>j</i>	

Note

fortran indexing (row major,  $i \geq 1$ ,  $j \geq 1$ ,  $i \leq n$ ,  $j \leq n$ ).

7.3.3.5 `template<int n> double& odes::fortranArray<n>::operator() ( int i, int j )` `[inline]`

indexing.

## Parameters

<i>i</i>	
<i>j</i>	

## Note

fortran indexing (row major,  $i \geq 1$ ,  $j \geq 1$ ,  $i \leq n$ ,  $j \leq n$ ).  
returns a reference, possible leftvalue.

7.3.3.6 `template<int n> void odes::fortranArray< n >::print ( std::string s = " " ) const` `[inline]`

print

## 7.3.4 Member Data Documentation

7.3.4.1 `template<int n> const int odes::fortranArray< n >::n2 = n*n` `[static]`, `[private]`

7.3.4.2 `template<int n> double* odes::fortranArray< n >::x` `[private]`

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

- [common/include/fortranArray.hpp](#)

## 7.4 odes::fortranComplexArray&lt; n &gt; Class Template Reference

Array of complex..

```
#include <fortranComplexArray.hpp>
```

## Public Member Functions

- [fortranComplexArray](#) ()  
*constructor*
- [~fortranComplexArray](#) ()  
*destructor*
- void [set](#) (int i, int j, double RealPart, double ImagPart=0)
- double & [Re](#) (int i, int j)
- double & [Im](#) (int i, int j)
- double \* [operator&](#) ()  
*return adress of double array.*
- void [print](#) (std::string s="")  
*print*

## Private Attributes

- double \* [x](#)

## Static Private Attributes

- static const int [d2](#) = 2\*n
- static const int [size](#) = [d2](#)\*n

### 7.4.1 Detailed Description

```
template<int n> class odes::fortranComplexArray< n >
```

Array of complex..

Array of Complex, of fixed size n.n, fortran indexing. Actually, we do not make use of `complex<double>` class (we just want to store coefficients for lapack zge\* routines).

### 7.4.2 Constructor & Destructor Documentation

7.4.2.1 `template<int n> odes::fortranComplexArray< n >::fortranComplexArray ( ) [inline]`

contructor

7.4.2.2 `template<int n> odes::fortranComplexArray< n >::~~fortranComplexArray ( ) [inline]`

destructor

### 7.4.3 Member Function Documentation

7.4.3.1 `template<int n> double& odes::fortranComplexArray< n >::Im ( int i, int j ) [inline]`

Indexing, return a reference to "Imaginary" part.

Parameters

<i>i</i>	
<i>j</i>	

Note

fortran indexing (row major,  $i \geq 1$ ,  $j \geq 1$ ,  $i \leq n$ ,  $j \leq n$ ).

7.4.3.2 `template<int n> double* odes::fortranComplexArray< n >::operator& ( ) [inline]`

return adress of double array.

7.4.3.3 `template<int n> void odes::fortranComplexArray< n >::print ( std::string s = " " ) [inline]`

print

7.4.3.4 `template<int n> double& odes::fortranComplexArray< n >::Re ( int i, int j ) [inline]`

Indexing, return a reference to "real" part.

Parameters

<i>i</i>	
<i>j</i>	

Note

fortran indexing (row major,  $i \geq 1$ ,  $j \geq 1$ ,  $i \leq n$ ,  $j \leq n$ ).

7.4.3.5 `template<int n> void odes::fortranComplexArray< n >::set( int i, int j, double RealPart, double ImagPart = 0 )`  
`[inline]`

#### 7.4.4 Member Data Documentation

7.4.4.1 `template<int n> const int odes::fortranComplexArray< n >::d2 = 2*n` `[static], [private]`

7.4.4.2 `template<int n> const int odes::fortranComplexArray< n >::size = d2*n` `[static], [private]`

7.4.4.3 `template<int n> double* odes::fortranComplexArray< n >::x` `[private]`

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

- [common/include/fortranComplexArray.hpp](#)

## 7.5 odes::fortranRectangularArray< n, kl, ku > Class Template Reference

Banded Array of double.

```
#include <fortranRectangularArray.hpp>
```

### Public Member Functions

- [fortranRectangularArray\(\)](#)  
*constructor*
- [~fortranRectangularArray\(\)](#)  
*destructor*
- `double operator()(int i, int j) const`
- `double & operator()(int i, int j)`
- `void equal_minus(const fortranRectangularArray< n, kl, ku > &X)`
- `void addDiag(double v)`
- `double * operator&()`  
*return adress of double array.*
- `void print(std::string s="")`  
*print*

### Private Attributes

- `double * x`

### Static Private Attributes

- `static const int ldab = 2*kl+ku+1`
- `static const int klku = kl+ku`
- `static const int size = ldab*n`
- `static const int l1 = ldab-1`
- `static const int cc = klku-ldab`

### 7.5.1 Detailed Description

`template<int n, int kl, int ku> class odes::fortranRectangularArray< n, kl, ku >`

Banded Array of double.

Banded Array of double, of fixed size n, with kl subdiagonals and ku superdiagonals, fortran indexing, like in lapack routines (see dgbtrf for example).

### 7.5.2 Constructor & Destructor Documentation

**7.5.2.1** `template<int n, int kl, int ku> odes::fortranRectangularArray< n, kl, ku >::fortranRectangularArray ( )`  
`[inline]`

constructor

**7.5.2.2** `template<int n, int kl, int ku> odes::fortranRectangularArray< n, kl, ku >::~~fortranRectangularArray ( )`  
`[inline]`

destructor

### 7.5.3 Member Function Documentation

**7.5.3.1** `template<int n, int kl, int ku> void odes::fortranRectangularArray< n, kl, ku >::addDiag ( double v )`  
`[inline]`

add a value to diagonal.

Parameters

<code>v</code>	the value.
----------------	------------

define the ivdep pragma for different compilers.

**7.5.3.2** `template<int n, int kl, int ku> void odes::fortranRectangularArray< n, kl, ku >::equal_minus ( const fortranRectangularArray< n, kl, ku > & X )` `[inline]`

this = -X.

Parameters

<code>X</code>	the array.
----------------	------------

define the ivdep pragma for different compilers.

**7.5.3.3** `template<int n, int kl, int ku> double* odes::fortranRectangularArray< n, kl, ku >::operator& ( )`  
`[inline]`

return adress of double array.

**7.5.3.4** `template<int n, int kl, int ku> double odes::fortranRectangularArray< n, kl, ku >::operator() ( int i, int j ) const`  
`[inline]`

indexing

## Parameters

<i>i</i>	
<i>j</i>	

## Note

fortran indexing (row major,  $i \geq 1$ ,  $j \geq 1$ ,  $i \leq n$ ,  $j \leq n$ ).

**7.5.3.5** `template<int n, int kl, int ku> double& odes::fortranRectangularArray< n, kl, ku >::operator() ( int i, int j )`  
`[inline]`

indexing.

## Parameters

<i>i</i>	
<i>j</i>	

## Note

fortran indexing (row major,  $i \geq 1$ ,  $j \geq 1$ ,  $i \leq n$ ,  $j \leq n$ ).  
returns a reference, possible leftvalue.

**7.5.3.6** `template<int n, int kl, int ku> void odes::fortranRectangularArray< n, kl, ku >::print ( std::string s = " " )`  
`[inline]`

print

## 7.5.4 Member Data Documentation

**7.5.4.1** `template<int n, int kl, int ku> const int odes::fortranRectangularArray< n, kl, ku >::cc =klku-ldab`  
`[static], [private]`

**7.5.4.2** `template<int n, int kl, int ku> const int odes::fortranRectangularArray< n, kl, ku >::klku =kl+ku` `[static],`  
`[private]`

**7.5.4.3** `template<int n, int kl, int ku> const int odes::fortranRectangularArray< n, kl, ku >::l1 =ldab-1` `[static],`  
`[private]`

**7.5.4.4** `template<int n, int kl, int ku> const int odes::fortranRectangularArray< n, kl, ku >::ldab =2*kl+ku+1`  
`[static], [private]`

**7.5.4.5** `template<int n, int kl, int ku> const int odes::fortranRectangularArray< n, kl, ku >::size =ldab*n`  
`[static], [private]`

**7.5.4.6** `template<int n, int kl, int ku> double* odes::fortranRectangularArray< n, kl, ku >::x` `[private]`

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

- common/include/[fortranRectangularArray.hpp](#)

## 7.6 odes::fortranRectangularComplexArray< n, kl, ku > Class Template Reference

Banded Array of complex..

```
#include <fortranRectangularComplexArray.hpp>
```

### Public Member Functions

- [fortranRectangularComplexArray](#) ()  
*constructor*
- [~fortranRectangularComplexArray](#) ()  
*destructor*
- void [set](#) (int i, int j, double RealPart, double ImagPart=0)
- double & [Re](#) (int i, int j)
- double & [Im](#) (int i, int j)
- double \* [operator&](#) ()  
*return adress of double array.*
- void [print](#) (std::string s="")  
*print*

### Private Attributes

- double \* [x](#)

### Static Private Attributes

- static const int [ldab](#) =2\*kl+ku+1
- static const int [klku](#) =kl+ku
- static const int [size](#) =2\*[ldab](#)\*n
- static const int [l1](#) =[ldab](#)-1
- static const int [cc](#) =[klku](#)-[ldab](#)

#### 7.6.1 Detailed Description

```
template<int n, int kl, int ku>class odes::fortranRectangularComplexArray< n, kl, ku >
```

Banded Array of complex..

Banded Array of Complex, of fixed size n, with kl subdiagonals and ku superdiagonals, fortran indexing, like in lapack routines (see zgbtrf for axample). Actually, we do not make use of complex<double> class (we just want to store coefficients for lapack zge\* routines).

#### 7.6.2 Constructor & Destructor Documentation

7.6.2.1 `template<int n, int kl, int ku> odes::fortranRectangularComplexArray< n, kl, ku >::fortranRectangularComplexArray ( ) [inline]`

constructor

7.6.2.2 `template<int n, int kl, int ku> odes::fortranRectangularComplexArray< n, kl, ku >::~~fortranRectangularComplexArray ( ) [inline]`

destructor



### 7.6.3 Member Function Documentation

**7.6.3.1** `template<int n, int kl, int ku> double& odes::fortranRectangularComplexArray< n, kl, ku >::Im ( int i, int j )`  
`[inline]`

Indexing, return a reference to "Imaginary" part.

Parameters

<i>i</i>	
<i>j</i>	

Note

fortran indexing (row major,  $i \geq 1$ ,  $j \geq 1$ ,  $i \leq n$ ,  $j \leq n$ ).

**7.6.3.2** `template<int n, int kl, int ku> double* odes::fortranRectangularComplexArray< n, kl, ku >::operator& ( )`  
`[inline]`

return adress of double array.

**7.6.3.3** `template<int n, int kl, int ku> void odes::fortranRectangularComplexArray< n, kl, ku >::print ( std::string s = "" )`  
`[inline]`

print

**7.6.3.4** `template<int n, int kl, int ku> double& odes::fortranRectangularComplexArray< n, kl, ku >::Re ( int i, int j )`  
`[inline]`

Indexing, return a reference to "real" part.

Parameters

<i>i</i>	
<i>j</i>	

Note

fortran indexing (row major,  $i \geq 1$ ,  $j \geq 1$ ,  $i \leq n$ ,  $j \leq n$ ).

**7.6.3.5** `template<int n, int kl, int ku> void odes::fortranRectangularComplexArray< n, kl, ku >::set ( int i, int j, double RealPart, double ImagPart = 0 )`  
`[inline]`

### 7.6.4 Member Data Documentation

**7.6.4.1** `template<int n, int kl, int ku> const int odes::fortranRectangularComplexArray< n, kl, ku >::cc = klku-ldab`  
`[static], [private]`

**7.6.4.2** `template<int n, int kl, int ku> const int odes::fortranRectangularComplexArray< n, kl, ku >::klku = kl+ku`  
`[static], [private]`

**7.6.4.3** `template<int n, int kl, int ku> const int odes::fortranRectangularComplexArray< n, kl, ku >::l1 =ldab-1`  
`[static], [private]`

7.6.4.4 `template<int n, int kl, int ku> const int odes::fortranRectangularComplexArray< n, kl, ku >::ldab =2*kl+ku+1`  
`[static], [private]`

7.6.4.5 `template<int n, int kl, int ku> const int odes::fortranRectangularComplexArray< n, kl, ku >::size =2*ldab*n`  
`[static], [private]`

7.6.4.6 `template<int n, int kl, int ku> double* odes::fortranRectangularComplexArray< n, kl, ku >::x`  
`[private]`

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

- [common/include/fortranRectangularComplexArray.hpp](#)

## 7.7 odes::fortranVector Class Reference

Vector class (of doubles).

```
#include <fortranVector.hpp>
```

### Public Member Functions

- [fortranVector](#) ()
- [fortranVector](#) (int k)
- [fortranVector](#) (double \*\_x, int \_size)
- [~fortranVector](#) ()  
*destructor*
- double [operator\(\)](#) (int i) const
- double & [operator\(\)](#) (int i)
- double \* [operator&](#) () const  
*return a pointer to the vector of double.*
- void [operator=](#) ([fortranVector](#) V)
- void [operator+=](#) ([fortranVector](#) &F)
- void [setsize](#) (unsigned int \_size)
- int [get\\_size](#) () const  
*get the size*
- void [print](#) (string s="") const  
*print*

### Protected Attributes

- double \* [x](#)
- int [size](#)
- bool [deletable](#)

### Friends

- `template<int k>`  
`class fortranArray`
- `template<int k>`  
`class fortranVectorF`

### 7.7.1 Detailed Description

Vector class (of doubles).

Vector class (of doubles). We want to use fortran indexing (from 1 to...): this is just a wrapper around an array of doubles. Note: this is not a very *safe* class!. (we are adult programmers :-) ).

### 7.7.2 Constructor & Destructor Documentation

#### 7.7.2.1 odes::fortranVector::fortranVector ( ) [inline]

constructor

Note

we construct an empty class, not usable.

#### 7.7.2.2 odes::fortranVector::fortranVector ( int *k* ) [inline]

constructor

Parameters

<i>k</i>	size of the vector
----------	--------------------

#### 7.7.2.3 odes::fortranVector::fortranVector ( double \* *\_x*, int *\_size* ) [inline]

build a [fortranVector](#).

Parameters

<i>_x</i>	a vector of double
<i>_size</i>	size of the vector

Note

we "adopt" a vector, to be able to use fortran indexing; this is not safe!

#### 7.7.2.4 odes::fortranVector::~~fortranVector ( ) [inline]

destructor

### 7.7.3 Member Function Documentation

#### 7.7.3.1 int odes::fortranVector::get\_size ( ) const [inline]

get the size

#### 7.7.3.2 double\* odes::fortranVector::operator& ( ) const [inline]

return a pointer to the vector of double.

#### 7.7.3.3 double odes::fortranVector::operator() ( int *i* ) const [inline]

operator (): indexation

## Parameters

<i>i</i>	index (between 1 and n (included)).
----------	-------------------------------------

**7.7.3.4** `double& odes::fortranVector::operator()( int i )` `[inline]`

operator (): indexation

## Parameters

<i>i</i>	index (between 1 and n (included)).
----------	-------------------------------------

## Note

returns a reference (can be used as lvalue).

**7.7.3.5** `void odes::fortranVector::operator+=( fortranVector & F )` `[inline]`

operator +=

## Parameters

<i>F</i>	to be added.
----------	--------------

define the ivdep pragma for different compilers.

**7.7.3.6** `void odes::fortranVector::operator=( fortranVector V )` `[inline]`

copy constructor

## Parameters

<i>V</i>	object to copy
----------	----------------

## Note

nothing is checked! not safe.

define the ivdep pragma for different compilers.

**7.7.3.7** `void odes::fortranVector::print( string s = " " ) const` `[inline]`

print

**7.7.3.8** `void odes::fortranVector::setsize( unsigned int _size )` `[inline]`

set size *\_param* *\_size* the size.

## 7.7.4 Friends And Related Function Documentation

**7.7.4.1** `template<int k> friend class fortranArray` `[friend]`

**7.7.4.2** `template<int k> friend class fortranVectorF` `[friend]`

## 7.7.5 Member Data Documentation

7.7.5.1 `bool odes::fortranVector::deletable` [protected]

7.7.5.2 `int odes::fortranVector::size` [protected]

7.7.5.3 `double* odes::fortranVector::x` [protected]

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

- common/include/[fortranVector.hpp](#)

## 7.8 odes::fortranVectorF< n > Class Template Reference

Vector class (of doubles).

```
#include <fortranVector.hpp>
```

### Public Member Functions

- [fortranVectorF](#) ()  
*constructor. Object created is usable.*
- [~fortranVectorF](#) ()  
*destructor*
- void [switchadr](#) ([fortranVectorF](#)< n > &v)
- void [operator=](#) (double v)
- void [operator=](#) (double tab[])
- void [setsum2](#) ([fortranVector](#) &X, [fortranVector](#) &Y)
- void [operator=](#) ([fortranVectorF](#)< n > &X)
- void [operator+=](#) ([fortranVectorF](#)< n > &F)
- void [sum](#) ([fortranVectorF](#)< n > &X, [fortranVectorF](#)< n > &Y)
- void [sum](#) ([fortranVectorF](#)< n > &X, [fortranVector](#) &Y)
- void [a\\_Y](#) (double a, const [fortranVectorF](#)< n > &Y)
- void [x\\_lc2](#) ([fortranVectorF](#)< n > &X, double a1, [fortranVectorF](#)< n > &Y1, double a2, [fortranVectorF](#)< n > &Y2)
- void [lc2](#) (double a, [fortranVectorF](#)< n > &x1, double b, [fortranVectorF](#)< n > &x2)
- void [lc3](#) (double a, [fortranVectorF](#)< n > &x1, double b, [fortranVectorF](#)< n > &x2, double c, [fortranVectorF](#)< n > &x3)
- void [lc4](#) (double a, [fortranVectorF](#)< n > &x1, double b, [fortranVectorF](#)< n > &x2, double c, [fortranVectorF](#)< n > &x3, double d, [fortranVectorF](#)< n > &x4)
- void [lc5](#) (double a, [fortranVectorF](#)< n > &x1, double b, [fortranVectorF](#)< n > &x2, double c, [fortranVectorF](#)< n > &x3, double d, [fortranVectorF](#)< n > &x4, double e, [fortranVectorF](#)< n > &x5)
- void [x\\_lc3](#) ([fortranVectorF](#)< n > &X, double a1, [fortranVectorF](#)< n > &Y1, double a2, [fortranVectorF](#)< n > &Y2, double a3, [fortranVectorF](#)< n > &Y3)
- void [x\\_lc4](#) ([fortranVectorF](#)< n > &X, double a1, [fortranVectorF](#)< n > &Y1, double a2, [fortranVectorF](#)< n > &Y2, double a3, [fortranVectorF](#)< n > &Y3, double a4, [fortranVectorF](#)< n > &Y4)
- void [a\\_x\\_plus\\_y](#) (double a, [fortranVectorF](#)< n > &X, [fortranVectorF](#)< n > &Y)
- void [put](#) (double \_x[])
- void [print](#) (string s="") const  
*print*
- double [operator\(\)](#) (int i) const
- double & [operator\(\)](#) (int i)
- double \* [operator&](#) () const  
*return a pointer to the vector of double.*
- void [operator+=](#) ([fortranVector](#) &F)

- void `setsize` (unsigned int `_size`)
- int `get_size` () const  
*get the size*

### Protected Attributes

- double \* `x`
- int `size`
- bool `deletable`

### Private Attributes

- double \* `y`

## 7.8.1 Detailed Description

`template<int n>class odes::fortranVectorF< n >`

Vector class (of doubles).

Vector class (of doubles) OF FIXED SIZE `n`. We want to use fortran indexing (from 1 to...): this is just a wrapper around an array of doubles. Note: this is not a very safe class!.

## 7.8.2 Constructor & Destructor Documentation

**7.8.2.1** `template<int n> odes::fortranVectorF< n >::fortranVectorF ( ) [inline]`

constructor. Object created is usable.

**7.8.2.2** `template<int n> odes::fortranVectorF< n >::~~fortranVectorF ( ) [inline]`

destructor

## 7.8.3 Member Function Documentation

**7.8.3.1** `template<int n> void odes::fortranVectorF< n >::a_x_plus_y ( double a, fortranVectorF< n > & X, fortranVectorF< n > & Y ) [inline]`

Parameters

<code>a</code>	coefficient
<code>X</code>	<code>fortranVectorF&lt;n&gt;</code>
<code>Y</code>	<code>fortranVectorF&lt;n&gt;</code>

define the ivdep pragma for different compilers.

**7.8.3.2** `template<int n> void odes::fortranVectorF< n >::a_Y ( double a, const fortranVectorF< n > & Y ) [inline]`

this= a\*Y

## Parameters

<i>a</i>	a double
<i>Y</i>	fortranVector<n>

define the ivdep pragma for different compilers.

**7.8.3.3** `int odes::fortranVector::get_size ( ) const` `[inline],[inherited]`

get the size

**7.8.3.4** `template<int n> void odes::fortranVectorF< n >::lc2 ( double a, fortranVectorF< n > & x1, double b, fortranVectorF< n > & x2 )` `[inline]`

this = linear combination of 2 fortranVectorF<n>

## Parameters

<i>a</i>	coefficient
<i>x1</i>	fortranVectorF<n>
<i>b</i>	coefficient
<i>x2</i>	fortranVectorF<n>

define the ivdep pragma for different compilers.

**7.8.3.5** `template<int n> void odes::fortranVectorF< n >::lc3 ( double a, fortranVectorF< n > & x1, double b, fortranVectorF< n > & x2, double c, fortranVectorF< n > & x3 )` `[inline]`

this = linear combination of 3 fortranVectorF<n>

## Parameters

<i>a</i>	coefficient
<i>x1</i>	fortranVectorF<n>
<i>b</i>	coefficient
<i>x2</i>	fortranVectorF<n>
<i>c</i>	coefficient
<i>x3</i>	fortranVectorF<n>

define the ivdep pragma for different compilers.

**7.8.3.6** `template<int n> void odes::fortranVectorF< n >::lc4 ( double a, fortranVectorF< n > & x1, double b, fortranVectorF< n > & x2, double c, fortranVectorF< n > & x3, double d, fortranVectorF< n > & x4 )` `[inline]`

this = linear combination of 4 fortranVectorF<n>

## Parameters

<i>a</i>	coefficient
<i>x1</i>	fortranVectorF<n>
<i>b</i>	coefficient
<i>x2</i>	fortranVectorF<n>
<i>c</i>	coefficient
<i>x3</i>	fortranVectorF<n>

<i>d</i>	coefficient
<i>x4</i>	fortranVectorF<n>

define the ivdep pragma for different compilers.

```
7.8.3.7  template<int n> void odes::fortranVectorF<n>::lc5 ( double a, fortranVectorF<n> & x1, double b,
    fortranVectorF<n> & x2, double c, fortranVectorF<n> & x3, double d, fortranVectorF<n> & x4,
    double e, fortranVectorF<n> & x5 )  [inline]
```

this = linear combination of 5 fortranVectorF<n>

Parameters

<i>a</i>	coefficient
<i>x1</i>	fortranVectorF<n>
<i>b</i>	coefficient
<i>x2</i>	fortranVectorF<n>
<i>c</i>	coefficient
<i>x3</i>	fortranVectorF<n>
<i>d</i>	coefficient
<i>x4</i>	fortranVectorF<n>
<i>e</i>	coefficient
<i>x5</i>	fortranVectorF<n>

define the ivdep pragma for different compilers.

```
7.8.3.8  double* odes::fortranVector::operator& ( ) const  [inline],[inherited]
```

return a pointer to the vector of double.

```
7.8.3.9  double odes::fortranVector::operator() ( int i ) const  [inline],[inherited]
```

operator (): indexation

Parameters

<i>i</i>	index (between 1 and n (included)).
----------	-------------------------------------

```
7.8.3.10 double& odes::fortranVector::operator() ( int i )  [inline],[inherited]
```

operator (): indexation

Parameters

<i>i</i>	index (between 1 and n (included)).
----------	-------------------------------------

Note

returns a reference (can be used as lvalue).

```
7.8.3.11 void odes::fortranVector::operator+= ( fortranVector & F )  [inline],[inherited]
```

operator +=



## Parameters

<i>F</i>	to be added.
----------	--------------

define the ivdep pragma for different compilers.

**7.8.3.12** `template<int n> void odes::fortranVectorF< n >::operator+=( fortranVectorF< n > & F ) [inline]`

operator +=

## Parameters

<i>F</i>	to be added.
----------	--------------

define the ivdep pragma for different compilers.

**7.8.3.13** `template<int n> void odes::fortranVectorF< n >::operator=( double v ) [inline]`

put the same value everywhere.

## Parameters

<i>v</i>	the value.
----------	------------

define the ivdep pragma for different compilers.

**7.8.3.14** `template<int n> void odes::fortranVectorF< n >::operator=( double tab[] ) [inline]`

copy an array of double.

## Parameters

<i>tab[]</i>	the array.
--------------	------------

## Note

we do not check the size of *tab[]*.

define the ivdep pragma for different compilers.

**7.8.3.15** `template<int n> void odes::fortranVectorF< n >::operator=( fortranVectorF< n > & X ) [inline]`

operator=

## Parameters

<i>X</i>	*this=X.
----------	----------

define the ivdep pragma for different compilers.

**7.8.3.16** `template<int n> void odes::fortranVectorF< n >::print ( string s = " " ) const [inline]`

print

**7.8.3.17** `template<int n> void odes::fortranVectorF< n >::put ( double _x[] ) [inline]`

copy in a vector of double.

## Parameters

<code>_x[]</code>	the vector (result).
-------------------	----------------------

## Note

we cannot check for the size of `x[]`.

define the `ivdep` pragma for different compilers.

**7.8.3.18** `void odes::fortranVector::setsize ( unsigned int _size ) [inline],[inherited]`

set size `_param _size` the size.

**7.8.3.19** `template<int n> void odes::fortranVectorF<n>::setsum2 ( fortranVector & X, fortranVector & Y ) [inline]`

this <- X+Y.

## Parameters

<code>X</code>	
<code>Y</code>	

## Note

we do not make any check on `X` and `Y`

define the `ivdep` pragma for different compilers.

**7.8.3.20** `template<int n> void odes::fortranVectorF<n>::sum ( fortranVectorF<n> & X, fortranVectorF<n> & Y ) [inline]`

this = X +Y.

## Parameters

<code>X</code>	<code>fortranVectorF&lt;n&gt;</code>
<code>Y</code>	<code>fortranVectorF&lt;n&gt;</code>

define the `ivdep` pragma for different compilers.

**7.8.3.21** `template<int n> void odes::fortranVectorF<n>::sum ( fortranVectorF<n> & X, fortranVector & Y ) [inline]`

this = X +Y.

## Parameters

<code>X</code>	<code>fortranVectorF&lt;n&gt;</code>
<code>Y</code>	<code>fortranVector&lt;n&gt;</code>

define the `ivdep` pragma for different compilers.

**7.8.3.22** `template<int n> void odes::fortranVectorF<n>::switchadr ( fortranVectorF<n> & v ) [inline]`

switch adress with an other `fortranVectorF<n>`

## Parameters

$v$	we switch with $v$ .
-----	----------------------

**7.8.3.23** `template<int n> void odes::fortranVectorF< n >::x_lc2 ( fortranVectorF< n > & X, double a1, fortranVectorF< n > & Y1, double a2, fortranVectorF< n > & Y2 ) [inline]`

this= X+ a1\*Y1 + a2\*Y2.

## Parameters

$X$	
$a1$	
$Y1$	
$a2$	
$Y2$	

define the ivdep pragma for different compilers.

**7.8.3.24** `template<int n> void odes::fortranVectorF< n >::x_lc3 ( fortranVectorF< n > & X, double a1, fortranVectorF< n > & Y1, double a2, fortranVectorF< n > & Y2, double a3, fortranVectorF< n > & Y3 ) [inline]`

this= X+ a1\*Y1 + a2\*Y2 + a3\*Y3

## Parameters

$X$	
$a1$	
$Y1$	
$a2$	
$Y2$	
$a3$	
$Y3$	

define the ivdep pragma for different compilers.

**7.8.3.25** `template<int n> void odes::fortranVectorF< n >::x_lc4 ( fortranVectorF< n > & X, double a1, fortranVectorF< n > & Y1, double a2, fortranVectorF< n > & Y2, double a3, fortranVectorF< n > & Y3, double a4, fortranVectorF< n > & Y4 ) [inline]`

this= X+ a1\*Y1 + a2\*Y2 + a3\*Y3 + a4\* Y4

## Parameters

$X$	
$a1$	
$Y1$	
$a2$	
$Y2$	
$a3$	
$Y3$	
$a4$	
$Y4$	

define the ivdep pragma for different compilers.

## 7.8.4 Member Data Documentation

7.8.4.1 `bool odes::fortranVector::deletable` `[protected],[inherited]`

7.8.4.2 `int odes::fortranVector::size` `[protected],[inherited]`

7.8.4.3 `double* odes::fortranVector::x` `[protected],[inherited]`

7.8.4.4 `template<int n> double* odes::fortranVectorF<n>::y` `[private]`

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

- [common/include/fortranVector.hpp](#)

## 7.9 GenericException Class Reference

```
#include <GenericException.hpp>
```

### Public Member Functions

- [GenericException](#) ()  
*0 argument.*
- `template<class A >`  
[GenericException](#) (A x)  
*1 argument*
- `template<class A , class B >`  
[GenericException](#) (A x, B y)  
*2 arguments*
- `template<class A , class B , class C >`  
[GenericException](#) (A x, B y, C z)  
*3 arguments*
- `template<class A , class B , class C , class D >`  
[GenericException](#) (A x, B y, C z, D a)  
*4 arguments*
- `template<class A , class B , class C , class D , class E >`  
[GenericException](#) (A x, B y, C z, D a, E b)  
*5 arguments*
- `template<class A , class B , class C , class D , class E , class F >`  
[GenericException](#) (A x, B y, C z, D a, E b, F c)  
*6 arguments*
- `template<class A , class B , class C , class D , class E , class F , class G >`  
[GenericException](#) (A x, B y, C z, D a, E b, F c, G d)  
*7 arguments:*
- `template<class A , class B , class C , class D , class E , class F , class G , class H >`  
[GenericException](#) (A x, B y, C z, D a, E b, F c, G d, H e)  
*8 arguments:*
- `template<class A , class B , class C , class D , class E , class F , class G , class H , class I >`  
[GenericException](#) (A x, B y, C z, D a, E b, F c, G d, H e, I f)  
*9 arguments:*
- `template<class A , class B , class C , class D , class E , class F , class G , class H , class I , class J >`  
[GenericException](#) (A x, B y, C z, D a, E b, F c, G d, H e, I f, J g)  
*10 arguments:*

### 7.9.1 Detailed Description

A simple exception class. Instantiate it with 1 to 6 parameters. All parameters must be printable, and will actually be printed.

### 7.9.2 Constructor & Destructor Documentation

#### 7.9.2.1 GenericException::GenericException ( ) [inline]

0 argument.

#### 7.9.2.2 template<class A > GenericException::GenericException ( A x ) [inline]

1 argument

#### 7.9.2.3 template<class A , class B > GenericException::GenericException ( A x, B y ) [inline]

2 arguments

#### 7.9.2.4 template<class A , class B , class C > GenericException::GenericException ( A x, B y, C z ) [inline]

3 arguments

#### 7.9.2.5 template<class A , class B , class C , class D > GenericException::GenericException ( A x, B y, C z, D a ) [inline]

4 arguments

#### 7.9.2.6 template<class A , class B , class C , class D , class E > GenericException::GenericException ( A x, B y, C z, D a, E b ) [inline]

5 arguments

#### 7.9.2.7 template<class A , class B , class C , class D , class E , class F > GenericException::GenericException ( A x, B y, C z, D a, E b, F c ) [inline]

6 arguments

#### 7.9.2.8 template<class A , class B , class C , class D , class E , class F , class G > GenericException::GenericException ( A x, B y, C z, D a, E b, F c, G d ) [inline]

7 arguments:

#### 7.9.2.9 template<class A , class B , class C , class D , class E , class F , class G , class H > GenericException::GenericException ( A x, B y, C z, D a, E b, F c, G d, H e ) [inline]

8 arguments:

```
7.9.2.10 template<class A , class B , class C , class D , class E , class F , class G , class H , class I >
        GenericException::GenericException ( A x, B y, C z, D a, E b, F c, G d, H e, I f ) [inline]
```

9 arguments:

```
7.9.2.11 template<class A , class B , class C , class D , class E , class F , class G , class H , class I , class J >
        GenericException::GenericException ( A x, B y, C z, D a, E b, F c, G d, H e, I f, J g ) [inline]
```

10 arguments:

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

- common/include/[GenericException.hpp](#)

## 7.10 odes::logger Class Reference

For logging events.

```
#include <logger.hpp>
```

### Classes

- struct [event](#)

### Public Member Functions

- [logger](#) ()  
*constructor*
- [~logger](#) ()  
*destructor*
- void [clear](#) ()  
*reset.*
- void [put](#) (double \_time, int \_step, [eventType](#) \_TheEvent, double \_value)
- void [print](#) ([eventType](#) E=all)

### Private Attributes

- list< [event](#) > L

#### 7.10.1 Detailed Description

For logging events.

For logging events (change of time steps, rejected steps, and so on).

#### 7.10.2 Constructor & Destructor Documentation

```
7.10.2.1 odes::logger::logger ( ) [inline]
```

constructor

7.10.2.2 odes::logger::~~logger ( ) [inline]

destructor

### 7.10.3 Member Function Documentation

7.10.3.1 void odes::logger::clear ( ) [inline]

reset.

7.10.3.2 void odes::logger::print ( eventType *E* = all ) [inline]

print

Parameters

<i>E</i>	eventype, for filtering results.
----------	----------------------------------

7.10.3.3 void odes::logger::put ( double *\_time*, int *\_step*, eventType *\_TheEvent*, double *\_value* ) [inline]

put an event

Parameters

<i>_time</i>	actual time
<i>_step</i>	number
<i>_TheEvent</i>	event type (see struct event).
<i>_value</i>	some value like new time step.

### 7.10.4 Member Data Documentation

7.10.4.1 list<event> odes::logger::L [private]

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

- common/include/[logger.hpp](#)

## 7.11 odes::Matrices< full, Hessengerg, n, nsub, nsup > Class Template Reference

full matrices.

```
#include <Matrices.hpp>
```

### Public Types

- typedef [fortranArray](#)< n > [MatrixReal](#)
- typedef [fortranComplexArray](#)< n > [MatrixComplex](#)

### Protected Member Functions

- [Matrices](#) ( )
- void [decomr](#) (double fac1, [MatrixReal](#) &Jac)

- void `decomc` (double *alpha*, double *beta*, const `MatrixReal` &*Jac*)
- void `solvereal` (`fortranVectorF`< *n* > &*Z*, const `MatrixReal` &*Jac*)
- void `solvecomplex` (`fortranVectorF`< *n* > &*Zr*, `fortranVectorF`< *n* > &*Zi*, const `fortranArray`< *n* > &*Jac*)
- int `lbegin` (int *i*) const  
*begining of line i.*
- int `lend` (int *i*) const  
*end of line i.*

## Protected Attributes

- bool `calhes`

## Private Attributes

- int `ipivr` [*n*]
- int `ipivc` [*n*]
- `MatrixReal` *E1*
- `MatrixComplex` *E2R*
- `fortranVectorF`< 2 \**n* > *Z2N*

### 7.11.1 Detailed Description

template<bool full, bool Hessengerg, int n, int nsub, int nsup>class odes::Matrices< full, Hessengerg, n, nsub, nsup >

full matrices.

### 7.11.2 Member Typedef Documentation

7.11.2.1 template<bool full, bool Hessengerg, int n, int nsub, int nsup> typedef fortranComplexArray<*n*>  
odes::Matrices< full, Hessengerg, n, nsub, nsup >::MatrixComplex

7.11.2.2 template<bool full, bool Hessengerg, int n, int nsub, int nsup> typedef fortranArray<*n*> odes::Matrices< full,  
Hessengerg, n, nsub, nsup >::MatrixReal

### 7.11.3 Constructor & Destructor Documentation

7.11.3.1 template<bool full, bool Hessengerg, int n, int nsub, int nsup> odes::Matrices< full, Hessengerg, n, nsub, nsup  
>::Matrices ( ) [inline], [protected]

### 7.11.4 Member Function Documentation

7.11.4.1 template<bool full, bool Hessengerg, int n, int nsub, int nsup> void odes::Matrices< full, Hessengerg, n, nsub,  
nsup >::decomc ( double *alpha*, double *beta*, const `MatrixReal` & *Jac* ) [inline], [protected]

build and factorize "complex" matrix

Parameters

<i>alpha</i>	: we add $\alpha \cdot I$ to the real part of the Jacobian.
--------------	---



<i>beta</i>	: we add alpha*I to the imaginary part of the Jacobian.
<i>Jac</i>	the jacobian.

define the ivdep pragma for different compilers.

define the ivdep pragma for different compilers.

```
7.11.4.2  template<bool full, bool Hessengerg, int n, int nsub, int nsup> void odes::Matrices< full, Hessengerg, n, nsub,
        nsup >::decomr ( double fac1, MatrixReal & Jac )  [inline], [protected]
```

build and factorize "real" matrix

Parameters

<i>fac1</i>	: we add fac1*I to the Jacobian.
<i>Jac</i>	the jacobian.

```
7.11.4.3  template<bool full, bool Hessengerg, int n, int nsub, int nsup> int odes::Matrices< full, Hessengerg, n, nsub, nsup
        >::lbegin ( int i ) const  [inline], [protected]
```

begining of line i.

```
7.11.4.4  template<bool full, bool Hessengerg, int n, int nsub, int nsup> int odes::Matrices< full, Hessengerg, n, nsub, nsup
        >::lend ( int i ) const  [inline], [protected]
```

end of line i.

```
7.11.4.5  template<bool full, bool Hessengerg, int n, int nsub, int nsup> void odes::Matrices< full, Hessengerg, n, nsub,
        nsup >::solvecomplex ( fortranVectorF< n > & Zr, fortranVectorF< n > & Zi, const fortranArray< n > &
        Jac )  [inline], [protected]
```

solve "imaginary" part.

Parameters

<i>Zr</i>	IN/OUT: RHS, real part (IN); result,real part (OUT).
<i>Zi</i>	IN/OUT: RHS, imag. part (IN); result,imag part (OUT).
<i>Jac</i>	the Jacobian matrix.

define the ivdep pragma for different compilers.

define the ivdep pragma for different compilers.

```
7.11.4.6  template<bool full, bool Hessengerg, int n, int nsub, int nsup> void odes::Matrices< full, Hessengerg, n, nsub,
        nsup >::solvereal ( fortranVectorF< n > & Z, const MatrixReal & Jac )  [inline], [protected]
```

solve "real" part.

Parameters

<i>Z</i>	IN/OUT: 2nd member (IN); result (OUT).
<i>Jac</i>	the Jacobian matrix.

## 7.11.5 Member Data Documentation

```
7.11.5.1  template<bool full, bool Hessengerg, int n, int nsub, int nsup> bool odes::Matrices< full, Hessengerg, n, nsub,
        nsup >::calhes  [protected]
```

- 7.11.5.2 `template<bool full, bool Hessengerg, int n, int nsub, int nsup> MatrixReal odes::Matrices< full, Hessengerg, n, nsub, nsup >::E1` [private]
- 7.11.5.3 `template<bool full, bool Hessengerg, int n, int nsub, int nsup> MatrixComplex odes::Matrices< full, Hessengerg, n, nsub, nsup >::E2R` [private]
- 7.11.5.4 `template<bool full, bool Hessengerg, int n, int nsub, int nsup> int odes::Matrices< full, Hessengerg, n, nsub, nsup >::ipivc[n]` [private]
- 7.11.5.5 `template<bool full, bool Hessengerg, int n, int nsub, int nsup> int odes::Matrices< full, Hessengerg, n, nsub, nsup >::ipivr[n]` [private]
- 7.11.5.6 `template<bool full, bool Hessengerg, int n, int nsub, int nsup> fortranVectorF<2*n> odes::Matrices< full, Hessengerg, n, nsub, nsup >::Z2N` [private]

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

- common/include/[Matrices.hpp](#)

## 7.12 `odes::Matrices< false, false, n, nsub, nsup >` Class Template Reference

```
#include <Matrices.hpp>
```

### Public Types

- typedef [fortranRectangularArray](#)< n, nsub, nsup > [MatrixReal](#)
- typedef [fortranRectangularComplexArray](#)< n, nsub, nsup > [MatrixComplex](#)

### Protected Member Functions

- [Matrices](#) ()  
*constructor.*
- void [decomr](#) (double fac1, const [MatrixReal](#) &Jac)  
*see full matrix case.*
- void [decomc](#) (double alpha, double beta, const [MatrixReal](#) &Jac)  
*see full matrix case.*
- void [solvereal](#) ([fortranVectorF](#)< n > &Z, const [MatrixReal](#) &Jac)  
*see full matrix case.*
- void [solvecomplex](#) ([fortranVectorF](#)< n > &Zr, [fortranVectorF](#)< n > &Zi, const [MatrixReal](#) &Jac)  
*see full matrix case.*
- int [lbegin](#) (int i) const  
*begining of line i.*
- int [lend](#) (int i) const  
*end of line i.*

### Protected Attributes

- bool [calhes](#)

## Private Attributes

- int [ipivr](#) [n]
- int [ipivc](#) [n]
- [MatrixReal](#) E1
- [MatrixComplex](#) E2R
- [fortranVectorF](#)< 2 \*n > [Z2N](#)

## Static Private Attributes

- static const int [ldab](#) =2\*nsub+nsup+1

### 7.12.1 Detailed Description

template<int n, int nsub, int nsup>class odes::Matrices< false, false, n, nsub, nsup >

banded matrices, with nsub subdiagonals (nsub>=0) and nsup (>=0) super diagonals. storage is adapted to lapack banded matrices routines.

### 7.12.2 Member Typedef Documentation

7.12.2.1 template<int n, int nsub, int nsup> typedef [fortranRectangularComplexArray](#)<n,nsub,nsup> [odes::Matrices](#)< false, false, n, nsub, nsup >::[MatrixComplex](#)

7.12.2.2 template<int n, int nsub, int nsup> typedef [fortranRectangularArray](#)<n,nsub,nsup> [odes::Matrices](#)< false, false, n, nsub, nsup >::[MatrixReal](#)

### 7.12.3 Constructor & Destructor Documentation

7.12.3.1 template<int n, int nsub, int nsup> [odes::Matrices](#)< false, false, n, nsub, nsup >::[Matrices](#) ( ) [inline], [protected]

constructor.

### 7.12.4 Member Function Documentation

7.12.4.1 template<int n, int nsub, int nsup> void [odes::Matrices](#)< false, false, n, nsub, nsup >::[decomc](#) ( double *alpha*, double *beta*, const [MatrixReal](#) & *Jac* ) [inline], [protected]

see full matrix case.

define the ivdep pragma for different compilers.

7.12.4.2 template<int n, int nsub, int nsup> void [odes::Matrices](#)< false, false, n, nsub, nsup >::[decomr](#) ( double *fac1*, const [MatrixReal](#) & *Jac* ) [inline], [protected]

see full matrix case.

7.12.4.3 template<int n, int nsub, int nsup> int [odes::Matrices](#)< false, false, n, nsub, nsup >::[lbegin](#) ( int *i* ) const [inline], [protected]

begining of line i.

7.12.4.4 `template<int n, int nsub, int nsup> int odes::Matrices< false, false, n, nsub, nsup >::lend ( int i ) const`  
`[inline], [protected]`

end of line i.

7.12.4.5 `template<int n, int nsub, int nsup> void odes::Matrices< false, false, n, nsub, nsup >::solvecomplex`  
`( fortranVectorF< n > & Zr, fortranVectorF< n > & Zi, const MatrixReal & Jac ) [inline],`  
`[protected]`

see full matrix case.

define the ivdep pragma for different compilers.

define the ivdep pragma for different compilers.

7.12.4.6 `template<int n, int nsub, int nsup> void odes::Matrices< false, false, n, nsub, nsup >::solvereal (`  
`fortranVectorF< n > & Z, const MatrixReal & Jac ) [inline], [protected]`

see full matrix case.

## 7.12.5 Member Data Documentation

7.12.5.1 `template<int n, int nsub, int nsup> bool odes::Matrices< false, false, n, nsub, nsup >::calhes [protected]`

7.12.5.2 `template<int n, int nsub, int nsup> MatrixReal odes::Matrices< false, false, n, nsub, nsup >::E1`  
`[private]`

7.12.5.3 `template<int n, int nsub, int nsup> MatrixComplex odes::Matrices< false, false, n, nsub, nsup >::E2R`  
`[private]`

7.12.5.4 `template<int n, int nsub, int nsup> int odes::Matrices< false, false, n, nsub, nsup >::ipivc[n] [private]`

7.12.5.5 `template<int n, int nsub, int nsup> int odes::Matrices< false, false, n, nsub, nsup >::ipivr[n] [private]`

7.12.5.6 `template<int n, int nsub, int nsup> const int odes::Matrices< false, false, n, nsub, nsup >::ldab =2*nsub+nsup+1`  
`[static], [private]`

7.12.5.7 `template<int n, int nsub, int nsup> fortranVectorF<2*n> odes::Matrices< false, false, n, nsub, nsup >::Z2N`  
`[private]`

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

- [common/include/Matrices.hpp](#)

## 7.13 odes::Matrices< false, true, n, nsub, nsup > Class Template Reference

Banded matrices, Hessenberg=true: error.

```
#include <Matrices.hpp>
```

### Public Types

- typedef [fortranArray< n > MatrixReal](#)
- typedef [fortranComplexArray< n > MatrixComplex](#)

## Protected Member Functions

- [Matrices](#) ()  
*constructor.*
- void [decomr](#) (double fac1, const [MatrixReal](#) &Jac)
- void [decomc](#) (double alpha, double beta, const [MatrixReal](#) &Jac)
- void [solvevreal](#) ([fortranVectorF](#)< n > &Z, const [MatrixReal](#) &Jac)
- void [solvecomplex](#) ([fortranVectorF](#)< n > &Zr, [fortranVectorF](#)< n > &Zi, const [fortranArray](#)< n > &Jac)
- int [lbegin](#) (int i) const
- int [lend](#) (int i) const

## Protected Attributes

- bool [calhes](#)

### 7.13.1 Detailed Description

template<int n, int nsub, int nsup>class odes::Matrices< false, true, n, nsub, nsup >

Banded matrices, Hessenberg=true: error.

### 7.13.2 Member Typedef Documentation

7.13.2.1 template<int n, int nsub, int nsup> typedef [fortranComplexArray](#)<n> odes::Matrices< false, true, n, nsub, nsup >::MatrixComplex

7.13.2.2 template<int n, int nsub, int nsup> typedef [fortranArray](#)<n> odes::Matrices< false, true, n, nsub, nsup >::MatrixReal

### 7.13.3 Constructor & Destructor Documentation

7.13.3.1 template<int n, int nsub, int nsup> odes::Matrices< false, true, n, nsub, nsup >::Matrices ( ) [inline], [protected]

constructor.

### 7.13.4 Member Function Documentation

7.13.4.1 template<int n, int nsub, int nsup> void odes::Matrices< false, true, n, nsub, nsup >::decomc ( double *alpha*, double *beta*, const [MatrixReal](#) & *Jac* ) [inline], [protected]

7.13.4.2 template<int n, int nsub, int nsup> void odes::Matrices< false, true, n, nsub, nsup >::decomr ( double *fac1*, const [MatrixReal](#) & *Jac* ) [inline], [protected]

7.13.4.3 template<int n, int nsub, int nsup> int odes::Matrices< false, true, n, nsub, nsup >::lbegin ( int *i* ) const [inline], [protected]

7.13.4.4 template<int n, int nsub, int nsup> int odes::Matrices< false, true, n, nsub, nsup >::lend ( int *i* ) const [inline], [protected]

7.13.4.5 template<int n, int nsub, int nsup> void odes::Matrices< false, true, n, nsub, nsup >::solvecomplex ( [fortranVectorF](#)< n > & *Zr*, [fortranVectorF](#)< n > & *Zi*, const [fortranArray](#)< n > & *Jac* ) [inline], [protected]

7.13.4.6 `template<int n, int nsub, int nsup> void odes::Matrices< false, true, n, nsub, nsup >::solvereal (fortranVectorF< n > & Z, const MatrixReal & Jac ) [inline], [protected]`

### 7.13.5 Member Data Documentation

7.13.5.1 `template<int n, int nsub, int nsup> bool odes::Matrices< false, true, n, nsub, nsup >::calhes [protected]`

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

- common/include/[Matrices.hpp](#)

## 7.14 odes::Matrices< true, true, n, nsub, nsup > Class Template Reference

Full matrices, Hessenberg=true.

```
#include <Matrices.hpp>
```

### Public Types

- typedef [fortranArray](#)< n > [MatrixReal](#)
- typedef [fortranComplexArray](#)< n > [MatrixComplex](#)

### Protected Member Functions

- [Matrices](#) ()  
*constructor.*
- [~Matrices](#) ()
- void [decomr](#) (double fac1, [MatrixReal](#) &Jac)
- void [solvereal](#) ([fortranVectorF](#)< n > &Z, const [MatrixReal](#) &Jac)
- void [decomc](#) (double alpha, double beta, const [MatrixReal](#) &Jac)
- void [solvecomplex](#) ([fortranVectorF](#)< n > &Zr, [fortranVectorF](#)< n > &Zi, const [fortranArray](#)< n > &Jac)
- int [lbegin](#) (int i) const
- int [lend](#) (int i) const

### Protected Attributes

- bool [calhes](#)

### Private Attributes

- double \* [work](#)
- int [lwork](#)
- int [ipr](#) [n]
- int [ipi](#) [n]
- double [tau](#) [n]
- [MatrixReal](#) E1
- [MatrixReal](#) E2R
- [MatrixReal](#) E2I
- [fortranVectorF](#)< n > Cr
- [fortranVectorF](#)< n > Ci
- [fortranVectorF](#)< n > Fr
- [fortranVectorF](#)< n > Fi

### 7.14.1 Detailed Description

template<int n, int nsub, int nsup>class odes::Matrices< true, true, n, nsub, nsup >

Full matrices, Hessenberg=true.

### 7.14.2 Member Typedef Documentation

7.14.2.1 template<int n, int nsub, int nsup> typedef fortranComplexArray<n> odes::Matrices< true, true, n, nsub, nsup >::MatrixComplex

7.14.2.2 template<int n, int nsub, int nsup> typedef fortranArray<n> odes::Matrices< true, true, n, nsub, nsup >::MatrixReal

### 7.14.3 Constructor & Destructor Documentation

7.14.3.1 template<int n, int nsub, int nsup> odes::Matrices< true, true, n, nsub, nsup >::Matrices ( ) [inline], [protected]

constructor.

7.14.3.2 template<int n, int nsub, int nsup> odes::Matrices< true, true, n, nsub, nsup >::~~Matrices ( ) [inline], [protected]

### 7.14.4 Member Function Documentation

7.14.4.1 template<int n, int nsub, int nsup> void odes::Matrices< true, true, n, nsub, nsup >::decom ( double *alpha*, double *beta*, const MatrixReal & *Jac* ) [inline], [protected]

define the ivdep pragma for different compilers.

define the ivdep pragma for different compilers.

7.14.4.2 template<int n, int nsub, int nsup> void odes::Matrices< true, true, n, nsub, nsup >::decomr ( double *fac1*, MatrixReal & *Jac* ) [inline], [protected]

7.14.4.3 template<int n, int nsub, int nsup> int odes::Matrices< true, true, n, nsub, nsup >::lbegin ( int *i* ) const [inline], [protected]

7.14.4.4 template<int n, int nsub, int nsup> int odes::Matrices< true, true, n, nsub, nsup >::lend ( int *i* ) const [inline], [protected]

7.14.4.5 template<int n, int nsub, int nsup> void odes::Matrices< true, true, n, nsub, nsup >::solvecomplex ( fortranVectorF<n> & *Zr*, fortranVectorF<n> & *Zi*, const fortranArray<n> & *Jac* ) [inline], [protected]

7.14.4.6 template<int n, int nsub, int nsup> void odes::Matrices< true, true, n, nsub, nsup >::solvereal ( fortranVectorF<n> & *Z*, const MatrixReal & *Jac* ) [inline], [protected]

### 7.14.5 Member Data Documentation

7.14.5.1 template<int n, int nsub, int nsup> bool odes::Matrices< true, true, n, nsub, nsup >::calhes [protected]

- 7.14.5.2 `template<int n, int nsub, int nsup> fortranVectorF<n> odes::Matrices< true, true, n, nsub, nsup >::Ci`  
[private]
- 7.14.5.3 `template<int n, int nsub, int nsup> fortranVectorF<n> odes::Matrices< true, true, n, nsub, nsup >::Cr`  
[private]
- 7.14.5.4 `template<int n, int nsub, int nsup> MatrixReal odes::Matrices< true, true, n, nsub, nsup >::E1` [private]
- 7.14.5.5 `template<int n, int nsub, int nsup> MatrixReal odes::Matrices< true, true, n, nsub, nsup >::E2I`  
[private]
- 7.14.5.6 `template<int n, int nsub, int nsup> MatrixReal odes::Matrices< true, true, n, nsub, nsup >::E2R`  
[private]
- 7.14.5.7 `template<int n, int nsub, int nsup> fortranVectorF<n> odes::Matrices< true, true, n, nsub, nsup >::Fi`  
[private]
- 7.14.5.8 `template<int n, int nsub, int nsup> fortranVectorF<n> odes::Matrices< true, true, n, nsub, nsup >::Fr`  
[private]
- 7.14.5.9 `template<int n, int nsub, int nsup> int odes::Matrices< true, true, n, nsub, nsup >::ipi[n]` [private]
- 7.14.5.10 `template<int n, int nsub, int nsup> int odes::Matrices< true, true, n, nsub, nsup >::ipr[n]` [private]
- 7.14.5.11 `template<int n, int nsub, int nsup> int odes::Matrices< true, true, n, nsub, nsup >::lwork` [private]
- 7.14.5.12 `template<int n, int nsub, int nsup> double odes::Matrices< true, true, n, nsub, nsup >::tau[n]` [private]
- 7.14.5.13 `template<int n, int nsub, int nsup> double* odes::Matrices< true, true, n, nsub, nsup >::work` [private]

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

- [common/include/Matrices.hpp](#)

## 7.15 odes::Matrixtype< n, nsub, nsup > Struct Template Reference

```
#include <MatrixType.hpp>
```

### Public Types

- typedef [Matrices](#)< [Full](#), false,  
n, nsub, nsup >::MatrixReal [Matrix](#)

### Static Public Attributes

- static const bool [Full](#) =(n-nsub)==1&&(n-nsup)==1

### 7.15.1 Member Typedef Documentation

- 7.15.1.1 `template<int n, int nsub, int nsup> typedef Matrices<Full,false,n,nsub,nsup>::MatrixReal odes::Matrixtype<n, nsub, nsup >::Matrix`

### 7.15.2 Member Data Documentation



7.15.2.1 `template<int n, int nsub, int nsup> const bool odes::Matrixtype< n, nsub, nsup >::Full  
=(n-nsub)==1&&(n-nsup)==1 [static]`

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

- common/include/[MatrixType.hpp](#)

## 7.16 odes::Radau5cc< Fonct > Class Template Reference

A C++ implementation of Radau5.

```
#include <Radau5cc.hpp>
```

### Public Member Functions

- void [setTestPolicy](#) (bool \_sameTestValue, double \_atol[], double \_rtol[])
- [MatrixReal](#) & [Jacobian](#) ()  
*return a reference to the Jacobian.*
- Fonct & [rhs](#) ()  
*return a reference to the right-hand side object function.*
- [Radau5cc](#) (bool \_sameTestValue, double \_atol[], double \_rtol[])
- [~Radau5cc](#) ()  
*destructor*
- void [performOnlyOneStep](#) (bool val)
- void [setRecomputeJacobianTreshold](#) (double \_thet)
- void [setMaxIterationsNewton](#) (int \_nit)
- void [setSafe](#) (double \_safe)
- void [setFacrFacI](#) (double \_facr, double \_facI)
- void [setGustafssonTest](#) (bool val)
- void [setNmax](#) (int \_nmax)
- void [setFnewt](#) (double \_fnewt)
- int [getNstep](#) () const  
*get number of steps performed.*
- int [getNaccpt](#) () const  
*get number of steps accepted.*
- int [getNrejt](#) () const  
*get number of steps rejected.*
- int [getNJac](#) () const  
*get number of jacobian computed.*
- int [getNdec](#) () const  
*get number of matrix decompositions.*
- double [getLastTimeStep](#) () const  
*get last time step used.*
- int [getNewt](#) () const  
*get number of Newton iterations performed.*
- int [getNfncCalled](#) () const
- double [getfirstAcceptedStep](#) () const
- void [operator\(\)](#) (double \_h, double &x, double \_xend, double y[])

## Protected Member Functions

- void `Jacobian` (double t, `fortranVector` y, const `fortranVector` Fy)
- void `doNewtonMatrices` ()  
*make Newton matrices (factorize them).*
- void `slvrad` ()  
*Linear system solution in Newton iterations.*
- std::pair< bool, double > `NewtonIterationTest` ()
- int `Newton` (double x, `fortranVector` &Y)
- void `doScal` (const `fortranVector` &Y)
- double `estrad` (`fortranVectorF`< n > &y0, `fortranVector` &y, double t, bool first, bool reject)
- void `seth` (double x, double newh)

## Protected Attributes

- Fonct `F`
- `fortranVectorF`< n > `Z1`
- `fortranVectorF`< n > `Z2`
- `fortranVectorF`< n > `Z3`
- `fortranVectorF`< n > `F1`
- `fortranVectorF`< n > `F2`
- `fortranVectorF`< n > `F3`
- `fortranVectorF`< n > `Z1T`
- `fortranVectorF`< n > `Z2T`
- `fortranVectorF`< n > `Z3T`

## Private Types

- enum `OutNewtonLoop` { `converged`, `willNotConverge`, `didNotConverge` }
- typedef `Matrices`< `MatrixFull`, `Hessenberg`, n, ninf, nsup > ::`MatrixReal` `MatrixReal`
- typedef `fortranComplexArray`< n > `MatrixComplex`

## Private Member Functions

- void `decomr` (double fac1, `MatrixReal` &Jac)
- void `decomc` (double alpha, double beta, const `MatrixReal` &Jac)
- void `solvereal` (`fortranVectorF`< n > &Z, const `MatrixReal` &Jac)
- void `solvecomplex` (`fortranVectorF`< n > &Zr, `fortranVectorF`< n > &Zi, const `fortranArray`< n > &Jac)
- int `lbegin` (int i) const  
*begining of line i.*
- int `lend` (int i) const  
*end of line i.*
- `ct_assert` (full||(!full &&!Hessenberg))

### Private Attributes

- double [SQ6](#)
- double [C1](#)
- double [C2](#)
- double [C1M1](#)
- double [C2M1](#)
- double [C1MC2](#)
- double [DD1](#)
- double [DD2](#)
- double [DD3](#)
- double [U1](#)
- double [ALPH](#)
- double [BETA](#)
- double [CNO](#)
- double [T11](#)
- double [T12](#)
- double [T13](#)
- double [T21](#)
- double [T22](#)
- double [T23](#)
- double [T31](#)
- double [T111](#)
- double [T112](#)
- double [T113](#)
- double [T121](#)
- double [T122](#)
- double [T123](#)
- double [T131](#)
- double [T132](#)
- double [T133](#)
- double [uround](#)
- double [faccon](#)
- double [fnewt](#)
- double [tolst](#)
- double [xph](#)
- double [firstAcceptedStep](#)
- double [thet](#)
- double [dynold](#)
- double [h](#)
- double [hhfac](#)
- double [theta](#)
- double [dyno](#)
- double [safe](#)
- double [cfac](#)
- double [facr](#)
- double [facI](#)
- double [thqold](#)
- double [hacc](#)
- double [erracc](#)
- double [hold](#)
- double [xold](#)
- double [xend](#)
- double [posneg](#)
- double [hmaxn](#)

- double [hmax](#)
- double [hopt](#)
- double [quot1](#)
- double [quot2](#)
- bool [sameTestValue](#)
- bool [startn](#)
- bool [first](#)
- bool [caljac](#)
- bool [reject](#)
- bool [gustafssonTest](#)
- bool [last](#)
- bool [onlyOneStep](#)
- [fortranVectorF](#)< n > [atol](#)
- [fortranVectorF](#)< n > [rtol](#)
- [fortranVectorF](#)< n > [CONT1](#)
- [fortranVectorF](#)< n > [scal](#)
- [fortranVectorF](#)< n > [save1](#)
- [fortranVectorF](#)< n > [save2](#)
- [fortranVectorF](#)< n > [save3](#)
- [MatrixReal Jac](#)
- int [naccpt](#)
- int [nit](#)
- int [newt](#)
- int [nmax](#)
- int [nreject](#)
- int [njac](#)
- int [nstep](#)
- int [ndec](#)
- int [nfonccalled](#)
- bool [calhes](#)

### Static Private Attributes

- static const int [n](#) =Fonct::n
- static const int [ninf](#) =Fonct::nsub
- static const int [nsup](#) =Fonct::nsup
- static const bool [MatrixFull](#) =(n-ninf)==1&&(n-nsup)==1
- static const bool [Hessenberg](#) =Fonct::Hessenberg
- static const bool [ComputeJacobianNumerically](#)

#### 7.16.1 Detailed Description

template<class Fonct>class odes::Radau5cc< Fonct >

A C++ implementation of Radau5.

Class [Radau5cc](#)

A C++ implementation of (most part of) Hairer & Wanner Radau5. program.

## 7.16.2 Member Typedef Documentation

7.16.2.1 `typedef fortranComplexArray<n> odes::Matrices< full, Hessengerg, n, nsub, nsup >::MatrixComplex` [inherited]

7.16.2.2 `template<class Fonct > typedef Matrices<MatrixFull,Hessenberg,n,ninf,nsup>::MatrixReal`  
`odes::Radau5cc< Fonct >::MatrixReal` [private]

## 7.16.3 Member Enumeration Documentation

7.16.3.1 `template<class Fonct > enum odes::Radau5cc::OutNewtonLoop` [private]

Enumerator

***converged***  
***willNotConverge***  
***didNotConverge***

## 7.16.4 Constructor & Destructor Documentation

7.16.4.1 `template<class Fonct > odes::Radau5cc< Fonct >::Radau5cc ( bool _sameTestValue, double _atol[], double _rtol[] )` [inline]

constructor

Parameters

<code>_sameTestValue</code>	if true error tolerance are scalar
<code>_atol</code>	absolute tolerance
<code>_rtol</code>	relative tolerance

Note

even if the tolerance are scalar, `_atol` and `_rtol` are arrays.

7.16.4.2 `template<class Fonct > odes::Radau5cc< Fonct >::~~Radau5cc ( )` [inline]

destructor

## 7.16.5 Member Function Documentation

7.16.5.1 `compat< full, Hessenberg >::ct_assert ( full|| !full &&!Hessenberg )` [inherited]

7.16.5.2 `void odes::Matrices< full, Hessengerg, n, nsub, nsup >::decomc ( double alpha, double beta, const MatrixReal & Jac )` [inline], [protected], [inherited]

build and factorize "complex" matrix

Parameters

<code>alpha</code>	: we add $\alpha \cdot I$ to the real part of the Jacobian.
<code>beta</code>	: we add $\alpha \cdot I$ to the imaginary part of the Jacobian.

<i>Jac</i>	the jacobian.
------------	---------------

define the ivdep pragma for different compilers.

define the ivdep pragma for different compilers.

**7.16.5.3** `void odes::Matrices< full, Hessengerg, n, nsub, nsup >::decomr ( double fac1, MatrixReal & Jac )`  
`[inline], [protected], [inherited]`

build and factorize "real" matrix

Parameters

<i>fac1</i>	: we add <i>fac1</i> *I to the Jacobian.
<i>Jac</i>	the jacobian.

**7.16.5.4** `template<class Fonct > void odes::Radau5cc< Fonct >::doNewtonMatrices ( )` `[inline],`  
`[protected]`

make Newton matrices (factorize them).

**7.16.5.5** `template<class Fonct > void odes::Radau5cc< Fonct >::doScal ( const fortranVector & Y )` `[inline],`  
`[protected]`

(re) compute scal

Parameters

<i>Y</i>	
----------	--

**7.16.5.6** `template<class Fonct > double odes::Radau5cc< Fonct >::estrat ( fortranVectorF< n > & y0,`  
`fortranVector & y, double t, bool first, bool reject )` `[inline], [protected]`

error estimation.

Parameters

<i>y0</i>	
<i>y</i>	
<i>t</i>	
<i>first</i>	
<i>reject</i>	

**7.16.5.7** `template<class Fonct > double odes::Radau5cc< Fonct >::getfirstAcceptedStep ( ) const` `[inline]`

**7.16.5.8** `template<class Fonct > double odes::Radau5cc< Fonct >::getLastTimeStep ( ) const` `[inline]`

get last time step used.

**7.16.5.9** `template<class Fonct > int odes::Radau5cc< Fonct >::getNaccpt ( ) const` `[inline]`

get number of steps accepted.

7.16.5.10 `template<class Fonct > int odes::Radau5cc< Fonct >::getNdec ( ) const [inline]`

get number of matrix decompositions.

7.16.5.11 `template<class Fonct > int odes::Radau5cc< Fonct >::getNewt ( ) const [inline]`

get number of Newton iterations performed.

7.16.5.12 `template<class Fonct > int odes::Radau5cc< Fonct >::getNfncCalled ( ) const [inline]`

get number of function call.

#### Note

Z! if jacobian is computed numerically, we have n call for it. else, an estimation of the cost must take account of the cost of the computation of the exact Jacobian.

7.16.5.13 `template<class Fonct > int odes::Radau5cc< Fonct >::getNJac ( ) const [inline]`

get number of jacobian computed.

7.16.5.14 `template<class Fonct > int odes::Radau5cc< Fonct >::getNrejt ( ) const [inline]`

get number of steps rejected.

7.16.5.15 `template<class Fonct > int odes::Radau5cc< Fonct >::getNstep ( ) const [inline]`

get number of steps performed.

7.16.5.16 `int odes::Matrices< full, Hessengerg, n, nsub, nsup >::lbegin ( int i ) const [inline], [protected], [inherited]`

begining of line i.

7.16.5.17 `int odes::Matrices< full, Hessengerg, n, nsub, nsup >::lend ( int i ) const [inline], [protected], [inherited]`

end of line i.

7.16.5.18 `template<class Fonct > void odes::Radau5cc< Fonct >::Jacobian ( double t, fortranVector y, const fortranVector Fy ) [inline], [protected]`

compute (numerically) jacobian of Fonct

#### Parameters

<i>t</i>	current time (IN)
<i>y</i>	current value of unknowns (IN)
<i>Fy</i>	computed values of F(t,y,...) (IN)

7.16.5.19 `template<class Fonct > MatrixReal& odes::Radau5cc< Fonct >::Jacobian ( ) [inline]`

return a reference to the Jacobian.

7.16.5.20 `template<class Fonct > int odes::Radau5cc< Fonct >::Newton ( double x, fortranVector & Y ) [inline], [protected]`

perform simplified Newton iterations.

Parameters

x	time
Y	current value

Note

we return the reason for which we break iterations.

7.16.5.21 `template<class Fonct > std::pair<bool,double> odes::Radau5cc< Fonct >::NewtonIterationTest ( ) [inline], [protected]`

Test simplified Newton iterations convergence.

Note

return a pair made ok: (true or false) depending if the convergence. a suggested new value for the time step h.

7.16.5.22 `template<class Fonct > void odes::Radau5cc< Fonct >::operator() ( double _h, double & x, double _xend, double y[] ) [inline]`

make one time step.

Parameters

_h	time step
x	actual time
_xend	we integrate from x to _xend
y[]	current unknowns values.

7.16.5.23 `template<class Fonct > void odes::Radau5cc< Fonct >::performOnlyOneStep ( bool val ) [inline]`

operator will perform only one step, and return,whatever the value of time x.

Parameters

val	set onlyOneStep to val
-----	------------------------

Note

default is false, ie operator() will integrate up to \_xend (see operator()).

7.16.5.24 `template<class Fonct > Fonct& odes::Radau5cc< Fonct >::rhs ( ) [inline]`

return a reference to the right-hand side object function.



7.16.5.25 `template<class Fonct > void odes::Radau5cc< Fonct >::setFacrFacI ( double _facr, double _facI )`  
`[inline]`

change parameters for step size reduction.

## Parameters

<i>_facr</i>	
<i>_facI</i>	

## Note

new step h is chose so that  $\_facI \leq h/hold \leq \_facr$

7.16.5.26 `template<class Fonct > void odes::Radau5cc< Fonct >::setFnewt ( double _fnewt ) [inline]`

change stopping criterion for Newton iterations.

## Parameters

<i>_fnewt</i>	new value.
---------------	------------

7.16.5.27 `template<class Fonct > void odes::Radau5cc< Fonct >::setGustafssonTest ( bool val ) [inline]`

perform Gustafsson test? (default yes).

## Parameters

<i>val</i>	boolean.
------------	----------

7.16.5.28 `template<class Fonct > void odes::Radau5cc< Fonct >::seth ( double x, double newh ) [inline],  
[protected]`

set value of h.

## Parameters

<i>x</i>	current value of time.
<i>newh</i>	newvalue of h.

## Note

can be used to keep an history of h modifications.

7.16.5.29 `template<class Fonct > void odes::Radau5cc< Fonct >::setMaxIterationsNewton ( int _nit ) [inline]`

7.16.5.30 `template<class Fonct > void odes::Radau5cc< Fonct >::setNmax ( int _nmax ) [inline]`

change the maximum number of allowed steps.

## Parameters

<i>_nmax</i>	new value.
--------------	------------

7.16.5.31 `template<class Fonct > void odes::Radau5cc< Fonct >::setRecomputeJacobianTreshold ( double _thet )  
[inline]`

7.16.5.32 `template<class Fonct > void odes::Radau5cc< Fonct >::setSafe ( double _safe ) [inline]`

change safety factor for step size reduction

## Parameters

<code>_safe</code>	new value
--------------------	-----------

7.16.5.33 `template<class Fonct > void odes::Radau5cc< Fonct >::setTestPolicy ( bool _sameTestValue, double _atol[], double _rtol[] ) [inline]`

set the error test policy

## Parameters

<code>_sameTestValue</code>	== true iff atol and rtol are scalar
<code>_atol</code>	
<code>_rtol</code>	

7.16.5.34 `template<class Fonct > void odes::Radau5cc< Fonct >::slvrad ( ) [inline],[protected]`

Linear system solution in Newton iterations.

7.16.5.35 `void odes::Matrices< full, Hessengerg, n, nsub, nsup >::solvecomplex ( fortranVectorF< n > & Zr, fortranVectorF< n > & Zi, const fortranArray< n > & Jac ) [inline],[protected],[inherited]`

solve "imaginary" part.

## Parameters

<code>Zr</code>	IN/OUT: RHS, real part (IN); result,real part (OUT).
<code>Zi</code>	IN/OUT: RHS, imag. part (IN); result,imag part (OUT).
<code>Jac</code>	the Jacobian matrix.

define the ivdep pragma for different compilers.

define the ivdep pragma for different compilers.

7.16.5.36 `void odes::Matrices< full, Hessengerg, n, nsub, nsup >::solvereal ( fortranVectorF< n > & Z, const MatrixReal & Jac ) [inline],[protected],[inherited]`

solve "real" part.

## Parameters

<code>Z</code>	IN/OUT: 2nd member (IN); result (OUT).
<code>Jac</code>	the Jacobian matrix.

## 7.16.6 Member Data Documentation

7.16.6.1 `template<class Fonct > double odes::Radau5cc< Fonct >::ALPH [private]`

7.16.6.2 `template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::atol [private]`

7.16.6.3 `template<class Fonct > double odes::Radau5cc< Fonct >::BETA [private]`

7.16.6.4 `template<class Fonct > double odes::Radau5cc< Fonct >::C1 [private]`

7.16.6.5 `template<class Fonct > double odes::Radau5cc< Fonct >::C1M1 [private]`

```

7.16.6.6  template<class Fonct > double odes::Radau5cc< Fonct >::C1MC2  [private]
7.16.6.7  template<class Fonct > double odes::Radau5cc< Fonct >::C2  [private]
7.16.6.8  template<class Fonct > double odes::Radau5cc< Fonct >::C2M1  [private]
7.16.6.9  bool odes::Matrices< full, Hessengerg, n, nsub, nsup >::calhes  [protected], [inherited]
7.16.6.10 template<class Fonct > bool odes::Radau5cc< Fonct >::caljac  [private]
7.16.6.11 template<class Fonct > double odes::Radau5cc< Fonct >::cfac  [private]
7.16.6.12 template<class Fonct > double odes::Radau5cc< Fonct >::CNO  [private]
7.16.6.13 template<class Fonct > const bool odes::Radau5cc< Fonct >::ComputeJacobianNumerically  [static],
        [private]

```

#### Initial value:

```

=
    Fonct::ComputeJacobianNumerically

```

```

7.16.6.14 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::CONT1  [private]
7.16.6.15 template<class Fonct > double odes::Radau5cc< Fonct >::DD1  [private]
7.16.6.16 template<class Fonct > double odes::Radau5cc< Fonct >::DD2  [private]
7.16.6.17 template<class Fonct > double odes::Radau5cc< Fonct >::DD3  [private]
7.16.6.18 template<class Fonct > double odes::Radau5cc< Fonct >::dyno  [private]
7.16.6.19 template<class Fonct > double odes::Radau5cc< Fonct >::dynold  [private]
7.16.6.20 template<class Fonct > double odes::Radau5cc< Fonct >::erracc  [private]
7.16.6.21 template<class Fonct > Fonct odes::Radau5cc< Fonct >::F  [protected]
7.16.6.22 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::F1  [protected]
7.16.6.23 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::F2  [protected]
7.16.6.24 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::F3  [protected]
7.16.6.25 template<class Fonct > double odes::Radau5cc< Fonct >::faccon  [private]
7.16.6.26 template<class Fonct > double odes::Radau5cc< Fonct >::facI  [private]
7.16.6.27 template<class Fonct > double odes::Radau5cc< Fonct >::facr  [private]
7.16.6.28 template<class Fonct > bool odes::Radau5cc< Fonct >::first  [private]
7.16.6.29 template<class Fonct > double odes::Radau5cc< Fonct >::firstAcceptedStep  [private]
7.16.6.30 template<class Fonct > double odes::Radau5cc< Fonct >::fnewt  [private]

```

- 7.16.6.31 `template<class Fonct > bool odes::Radau5cc< Fonct >::gustafssonTest` [private]
- 7.16.6.32 `template<class Fonct > double odes::Radau5cc< Fonct >::h` [private]
- 7.16.6.33 `template<class Fonct > double odes::Radau5cc< Fonct >::hacc` [private]
- 7.16.6.34 `template<class Fonct > const bool odes::Radau5cc< Fonct >::Hessenberg =Fonct::Hessenberg`  
[static],[private]
- 7.16.6.35 `template<class Fonct > double odes::Radau5cc< Fonct >::hhfac` [private]
- 7.16.6.36 `template<class Fonct > double odes::Radau5cc< Fonct >::hmax` [private]
- 7.16.6.37 `template<class Fonct > double odes::Radau5cc< Fonct >::hmaxn` [private]
- 7.16.6.38 `template<class Fonct > double odes::Radau5cc< Fonct >::hold` [private]
- 7.16.6.39 `template<class Fonct > double odes::Radau5cc< Fonct >::hopt` [private]
- 7.16.6.40 `template<class Fonct > MatrixReal odes::Radau5cc< Fonct >::Jac` [private]
- 7.16.6.41 `template<class Fonct > bool odes::Radau5cc< Fonct >::last` [private]
- 7.16.6.42 `template<class Fonct > const bool odes::Radau5cc< Fonct >::MatrixFull =(n-ninf)==1&&(n-nsup)==1`  
[static],[private]
- 7.16.6.43 `template<class Fonct > const int odes::Radau5cc< Fonct >::n =Fonct::n` [static],[private]
- 7.16.6.44 `template<class Fonct > int odes::Radau5cc< Fonct >::naccpt` [private]
- 7.16.6.45 `template<class Fonct > int odes::Radau5cc< Fonct >::ndec` [private]
- 7.16.6.46 `template<class Fonct > int odes::Radau5cc< Fonct >::newt` [private]
- 7.16.6.47 `template<class Fonct > int odes::Radau5cc< Fonct >::nfonccalled` [private]
- 7.16.6.48 `template<class Fonct > const int odes::Radau5cc< Fonct >::ninf =Fonct::nsub` [static],[private]
- 7.16.6.49 `template<class Fonct > int odes::Radau5cc< Fonct >::nit` [private]
- 7.16.6.50 `template<class Fonct > int odes::Radau5cc< Fonct >::njac` [private]
- 7.16.6.51 `template<class Fonct > int odes::Radau5cc< Fonct >::nmax` [private]
- 7.16.6.52 `template<class Fonct > int odes::Radau5cc< Fonct >::nreject` [private]
- 7.16.6.53 `template<class Fonct > int odes::Radau5cc< Fonct >::nstep` [private]
- 7.16.6.54 `template<class Fonct > const int odes::Radau5cc< Fonct >::nsup =Fonct::nsup` [static],  
[private]
- 7.16.6.55 `template<class Fonct > bool odes::Radau5cc< Fonct >::onlyOneStep` [private]
- 7.16.6.56 `template<class Fonct > double odes::Radau5cc< Fonct >::posneg` [private]
- 7.16.6.57 `template<class Fonct > double odes::Radau5cc< Fonct >::quot1` [private]

7.16.6.58 `template<class Fonct> double odes::Radau5cc<Fonct>::quot2` [private]

7.16.6.59 `template<class Fonct> bool odes::Radau5cc<Fonct>::reject` [private]

7.16.6.60 `template<class Fonct> fortranVectorF<n> odes::Radau5cc<Fonct>::rtol` [private]

7.16.6.61 `template<class Fonct> double odes::Radau5cc<Fonct>::safe` [private]

7.16.6.62 `template<class Fonct> bool odes::Radau5cc<Fonct>::sameTestValue` [private]

7.16.6.63 `template<class Fonct> fortranVectorF<n> odes::Radau5cc<Fonct>::save1` [private]

7.16.6.64 `template<class Fonct> fortranVectorF<n> odes::Radau5cc<Fonct>::save2` [private]

7.16.6.65 `template<class Fonct> fortranVectorF<n> odes::Radau5cc<Fonct>::save3` [private]

7.16.6.66 `template<class Fonct> fortranVectorF<n> odes::Radau5cc<Fonct>::scal` [private]

7.16.6.67 `template<class Fonct> double odes::Radau5cc<Fonct>::SQ6` [private]

7.16.6.68 `template<class Fonct> bool odes::Radau5cc<Fonct>::startn` [private]

7.16.6.69 `template<class Fonct> double odes::Radau5cc<Fonct>::T11` [private]

7.16.6.70 `template<class Fonct> double odes::Radau5cc<Fonct>::T12` [private]

7.16.6.71 `template<class Fonct> double odes::Radau5cc<Fonct>::T13` [private]

7.16.6.72 `template<class Fonct> double odes::Radau5cc<Fonct>::T21` [private]

7.16.6.73 `template<class Fonct> double odes::Radau5cc<Fonct>::T22` [private]

7.16.6.74 `template<class Fonct> double odes::Radau5cc<Fonct>::T23` [private]

7.16.6.75 `template<class Fonct> double odes::Radau5cc<Fonct>::T31` [private]

7.16.6.76 `template<class Fonct> double odes::Radau5cc<Fonct>::thet` [private]

7.16.6.77 `template<class Fonct> double odes::Radau5cc<Fonct>::theta` [private]

7.16.6.78 `template<class Fonct> double odes::Radau5cc<Fonct>::thqold` [private]

7.16.6.79 `template<class Fonct> double odes::Radau5cc<Fonct>::Tl11` [private]

7.16.6.80 `template<class Fonct> double odes::Radau5cc<Fonct>::Tl12` [private]

7.16.6.81 `template<class Fonct> double odes::Radau5cc<Fonct>::Tl13` [private]

7.16.6.82 `template<class Fonct> double odes::Radau5cc<Fonct>::Tl21` [private]

7.16.6.83 `template<class Fonct> double odes::Radau5cc<Fonct>::Tl22` [private]

7.16.6.84 `template<class Fonct> double odes::Radau5cc<Fonct>::Tl23` [private]

7.16.6.85 `template<class Fonct> double odes::Radau5cc<Fonct>::Tl31` [private]

- 7.16.6.86 `template<class Fonct > double odes::Radau5cc< Fonct >::Ti32` [private]
- 7.16.6.87 `template<class Fonct > double odes::Radau5cc< Fonct >::Ti33` [private]
- 7.16.6.88 `template<class Fonct > double odes::Radau5cc< Fonct >::tolst` [private]
- 7.16.6.89 `template<class Fonct > double odes::Radau5cc< Fonct >::U1` [private]
- 7.16.6.90 `template<class Fonct > double odes::Radau5cc< Fonct >::uround` [private]
- 7.16.6.91 `template<class Fonct > double odes::Radau5cc< Fonct >::xend` [private]
- 7.16.6.92 `template<class Fonct > double odes::Radau5cc< Fonct >::xold` [private]
- 7.16.6.93 `template<class Fonct > double odes::Radau5cc< Fonct >::xph` [private]
- 7.16.6.94 `template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::Z1` [protected]
- 7.16.6.95 `template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::Z1T` [protected]
- 7.16.6.96 `template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::Z2` [protected]
- 7.16.6.97 `template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::Z2T` [protected]
- 7.16.6.98 `template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::Z3` [protected]
- 7.16.6.99 `template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::Z3T` [protected]

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

- Radau5-NG/include/[Radau5cc.hpp](#)





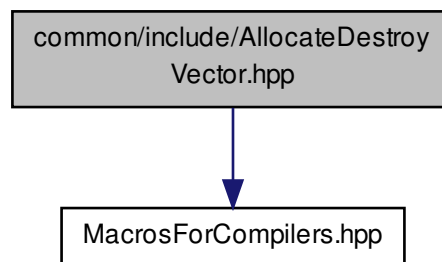
## Chapter 8

# File Documentation

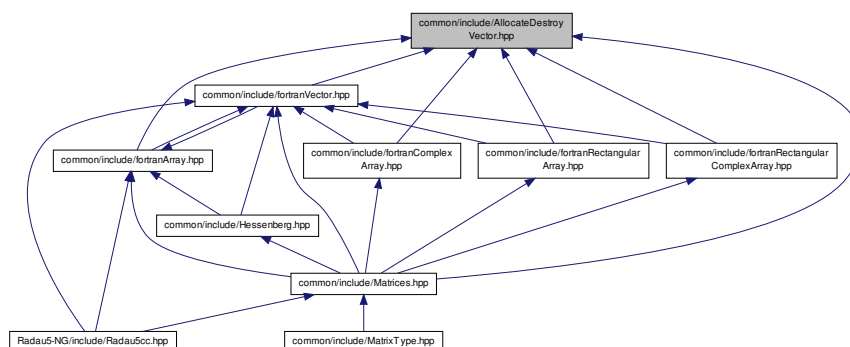
### 8.1 common/include/AllocateDestroyVector.hpp File Reference

```
#include "MacrosForCompilers.hpp"
```

Include dependency graph for AllocateDestroyVector.hpp:



This graph shows which files directly or indirectly include this file:



## Macros

- #define [ASSUME\\_ALIGNED](#)(lvalueptr)

## Functions

- double \* [allocDoubleArray](#) (int size)  
*Define different methods to allocate arrays.*
- void [destroyDoubleArray](#) (double \*x)

### 8.1.1 Macro Definition Documentation

#### 8.1.1.1 #define ASSUME\_ALIGNED( lvalueptr )

### 8.1.2 Function Documentation

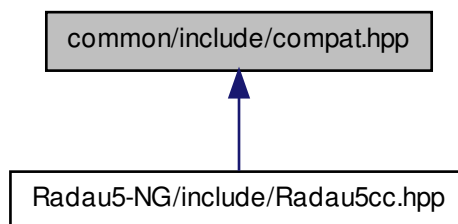
#### 8.1.2.1 double\* allocDoubleArray ( int size )

Define different methods to allocate arrays.

#### 8.1.2.2 void destroyDoubleArray ( double \* x )

## 8.2 common/include/compat.hpp File Reference

This graph shows which files directly or indirectly include this file:



## Classes

- struct [compat](#)< full, Hessenberg >

## Macros

- #define [ASSERT\\_CONCAT](#)\_(a, b) a##b
- #define [ASSERT\\_CONCAT](#)(a, b) [ASSERT\\_CONCAT](#)\_(a, b)
- #define [ct\\_assert](#)(e) enum { [ASSERT\\_CONCAT](#)(assert\_line\_, \_\_LINE\_\_) = 1/(!!(e)) }

### 8.2.1 Macro Definition Documentation

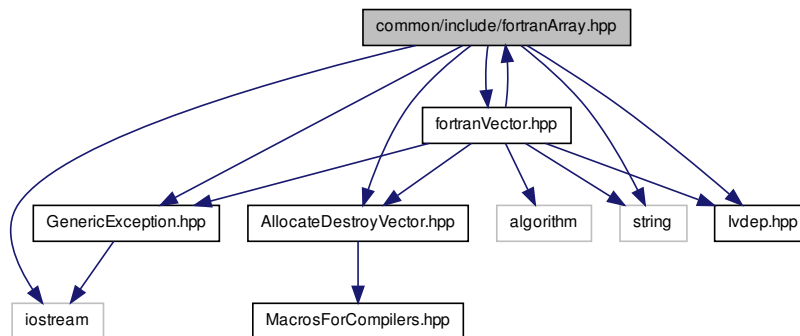
8.2.1.1 `#define ASSERT_CONCAT( a, b ) ASSERT_CONCAT_(a, b)`

8.2.1.2 `#define ASSERT_CONCAT_( a, b ) a##b`

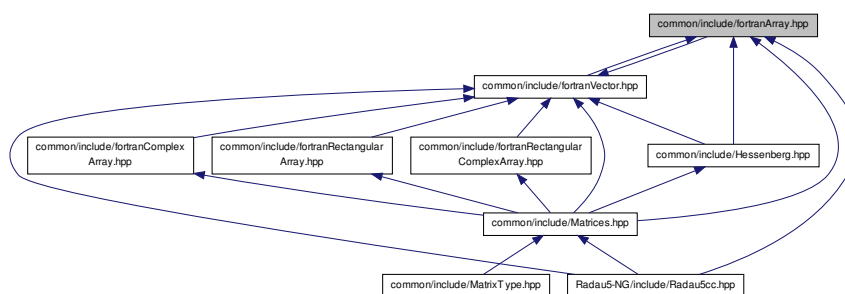
8.2.1.3 `#define ct_assert( e ) enum { ASSERT_CONCAT(assert_line_, __LINE__) = 1/(!!(e)) }`

## 8.3 common/include/fortranArray.hpp File Reference

```
#include "fortranVector.hpp"
#include "GenericException.hpp"
#include <string>
#include <iostream>
#include "AllocateDestroyVector.hpp"
#include "Ivdep.hpp"
Include dependency graph for fortranArray.hpp:
```



This graph shows which files directly or indirectly include this file:



### Classes

- class `odes::fortranArray< n >`  
Array of double.

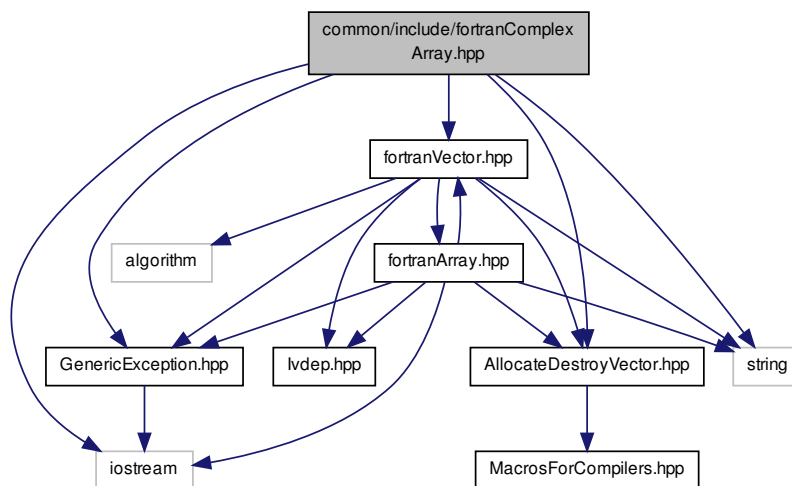
## Namespaces

- [odes](#)

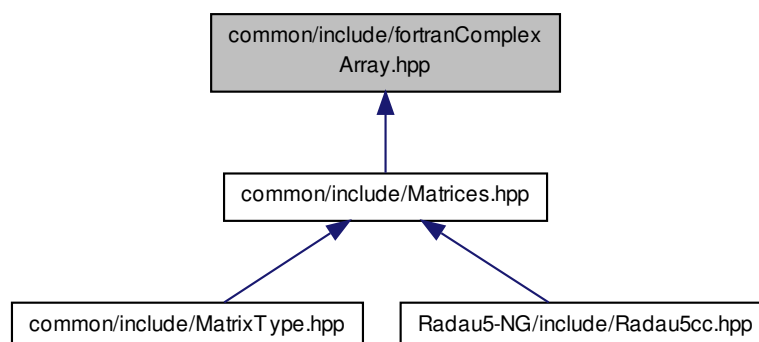
*Compute all matrix operations.*

## 8.4 common/include/fortranComplexArray.hpp File Reference

```
#include "fortranVector.hpp"
#include "GenericException.hpp"
#include <string>
#include <iostream>
#include "AllocateDestroyVector.hpp"
Include dependency graph for fortranComplexArray.hpp:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class `odes::fortranComplexArray< n >`

*Array of complex..*

## Namespaces

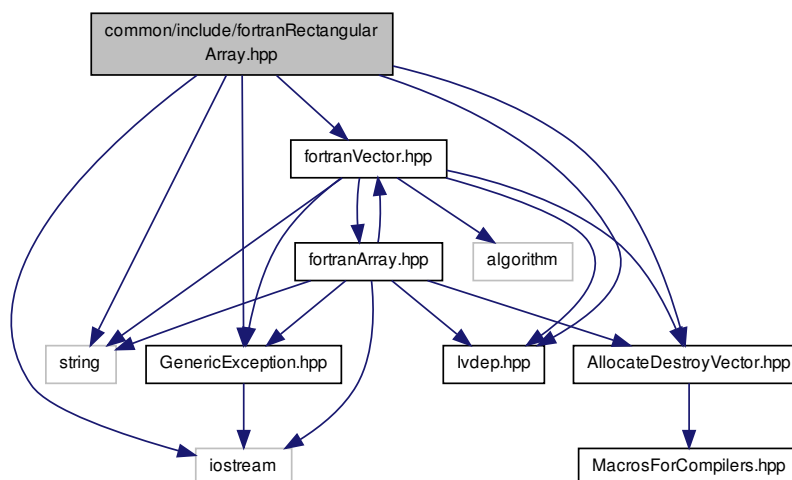
- `odes`

*Compute all matrix operations.*

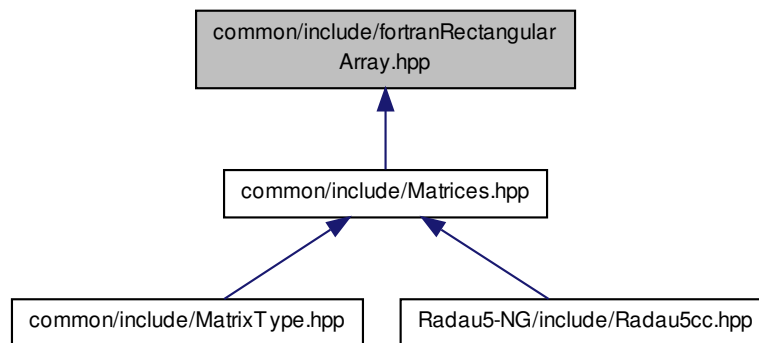
## 8.5 common/include/fortranRectangularArray.hpp File Reference

```
#include "fortranVector.hpp"
#include "GenericException.hpp"
#include "AllocateDestroyVector.hpp"
#include <string>
#include <iostream>
#include "Ivdep.hpp"
```

Include dependency graph for `fortranRectangularArray.hpp`:



This graph shows which files directly or indirectly include this file:



## Classes

- class `odes::fortranRectangularArray< n, kl, ku >`  
*Banded Array of double.*

## Namespaces

- `odes`  
*Compute all matrix operations.*

## Macros

- `#define MAX(x, y) ((x)>(y)?(x):(y))`
- `#define MIN(x, y) ((x)<(y)?(x):(y))`

### 8.5.1 Macro Definition Documentation

8.5.1.1 `#define MAX( x, y ) ((x)>(y)?(x):(y))`

8.5.1.2 `#define MIN( x, y ) ((x)<(y)?(x):(y))`

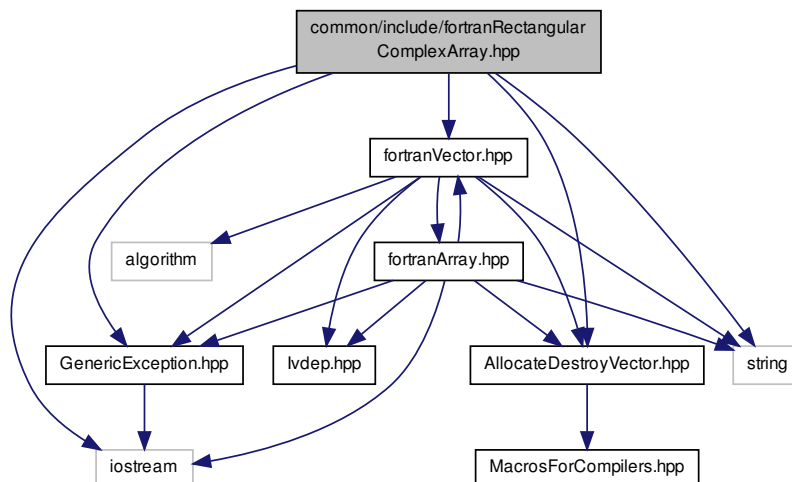
## 8.6 common/include/fortranRectangularComplexArray.hpp File Reference

```

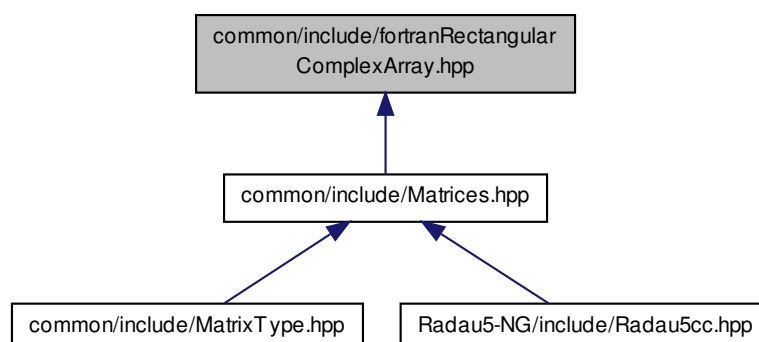
#include "fortranVector.hpp"
#include "GenericException.hpp"
#include "AllocateDestroyVector.hpp"
#include <string>
#include <iostream>

```

Include dependency graph for fortranRectangularComplexArray.hpp:



This graph shows which files directly or indirectly include this file:



## Classes

- class `odes::fortranRectangularComplexArray< n, kl, ku >`  
*Banded Array of complex..*

## Namespaces

- `odes`  
*Compute all matrix operations.*

## Macros

- `#define MAX(x, y) ((x)>(y)?(x):(y))`
- `#define MIN(x, y) ((x)<(y)?(x):(y))`

### 8.6.1 Macro Definition Documentation

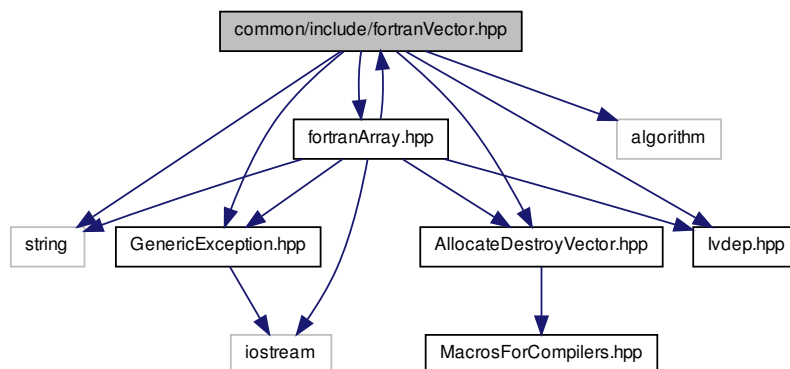
8.6.1.1 `#define MAX( x, y ) ((x)>(y)?(x):(y))`

8.6.1.2 `#define MIN( x, y ) ((x)<(y)?(x):(y))`

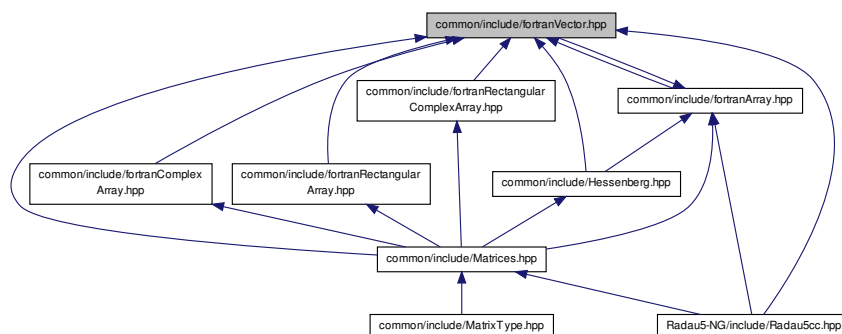
## 8.7 common/include/fortranVector.hpp File Reference

```
#include "GenericException.hpp"
#include "fortranArray.hpp"
#include "AllocateDestroyVector.hpp"
#include <algorithm>
#include <string>
#include "Ivdep.hpp"
```

Include dependency graph for fortranVector.hpp:



This graph shows which files directly or indirectly include this file:





## Classes

- class `odes::fortranVector`  
*Vector class (of doubles).*
- class `odes::fortranVectorF< n >`  
*Vector class (of doubles).*

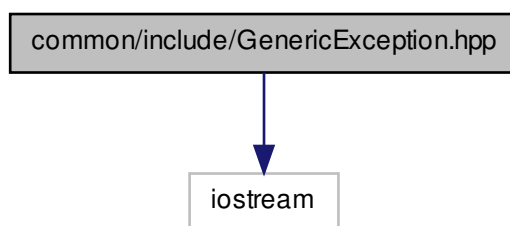
## Namespaces

- `odes`  
*Compute all matrix operations.*

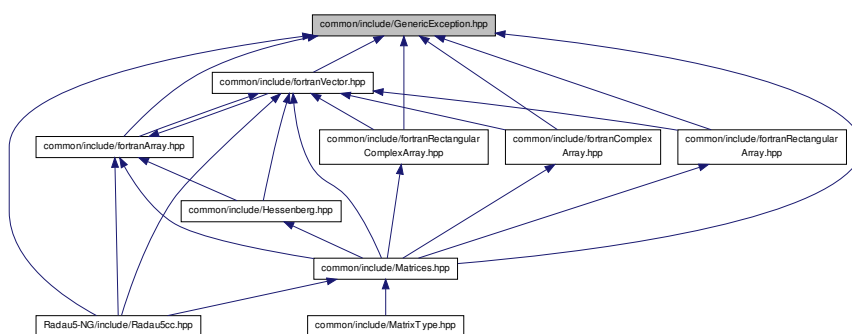
## 8.8 common/include/GenericException.hpp File Reference

```
#include <iostream>
```

Include dependency graph for GenericException.hpp:



This graph shows which files directly or indirectly include this file:

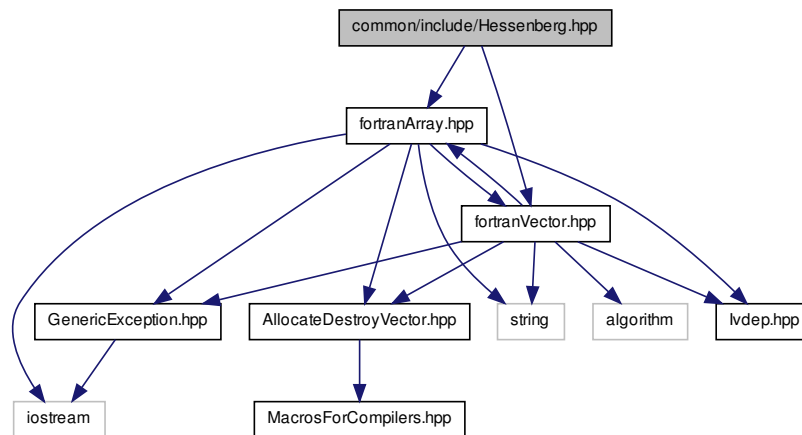


## Classes

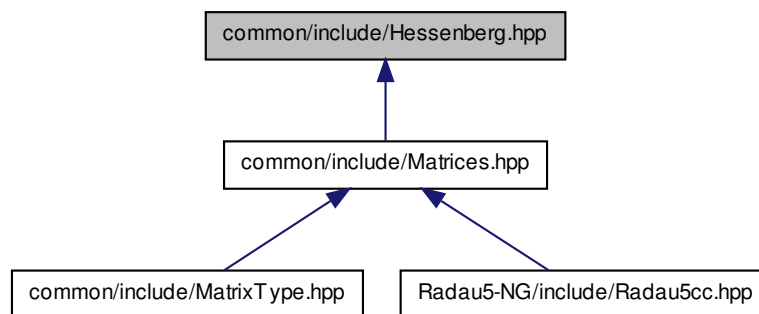
- class `GenericException`

## 8.9 common/include/Hessenberg.hpp File Reference

```
#include "fortranArray.hpp"
#include "fortranVector.hpp"
Include dependency graph for Hessenberg.hpp:
```



This graph shows which files directly or indirectly include this file:



### Macros

- `#define MIN(x, y) ((x)<(y)?(x):(y))`
- `#define ABS(a) (((a) >= (0.0)) ? (a) : (-a))`

### Functions

- `template<int n>`  
`int dech (fortranArray< n > &A, int ip[])`

*Triangularization by Gaussian elimination of a Hessenberk Matrix.*

- `template<int n>`  
void `solh` (const `fortranArray`< n > &A, const int ip[], `fortranVector` &B)
- `template<int n>`  
int `dechc` (`fortranArray`< n > &Ar, `fortranArray`< n > &Ai, int ip[])  
*Triangularization by Gaussian elimination of a complex.*
- `template<int n>`  
void `solhc` (const `fortranArray`< n > &Ar, const `fortranArray`< n > &Ai, const int ip[], `fortranVector` &Br, `fortranVector` &Bi)
- `template<int n>`  
void `householder` (const `fortranArray`< n > &A, const double tau[], const `fortranVector` &X, `fortranVector` &Y, bool direct)

### 8.9.1 Macro Definition Documentation

8.9.1.1 `#define ABS( a ) (((a) >= (0.0)) ? (a) : (-a))`

8.9.1.2 `#define MIN( x, y ) ((x)<(y)?(x):(y))`

### 8.9.2 Function Documentation

8.9.2.1 `template<int n> int dech ( fortranArray< n > & A, int ip[] )`

Triangularization by Gaussian elimination of a Hessenberk Matrix.

All what is necessary to compute with Hessenberg matrices (but not the reduction of a full matrix to Hessenberg form (see [Matrices.hpp](#)). Triangularization by Gaussian elimination of a Hessenberk Matrix with lower Bandwidth ==1.

Parameters

<i>A</i>	the matrix (IN/OUT).
<i>ip</i>	index of pivots.

Note

return 0 if everything went well, otherwise the rank when the matrix A was found singular.  
this is a C++ transcription of DECH (in H.&W. fortran code decsol).

8.9.2.2 `template<int n> int dechc ( fortranArray< n > & Ar, fortranArray< n > & Ai, int ip[] )`

Triangularization by Gaussian elimination of a complex.

Triangularization by Gaussian elimination of a Hessenberk Matrix with lower Bandwidth ==1. The matrix, here is complex, but treated as a pair of 2 real matrices. Hessenberg Matrix.

Note

pure transcription of H. & W. fortran code.

Parameters

<i>Ar</i>	the matrix,real part (IN/OUT).
<i>Ai</i>	the matrix,imaginary part (IN/OUT).

<i>ip</i>	index of pivots.
-----------	------------------

**Note**

return 0 if everything went well, otherwise the rank where the matrix A was found singular.  
this is a C++ transcription of DECHC (in H.&W. fortran code decsol).

**8.9.2.3** `template<int n> void householder ( const fortranArray< n > & A, const double tau[], const fortranVector & X, fortranVector & Y, bool direct )`

Given a Matrix A, in Hessenberg form, as computed by DGEHRD (lapack) with ilo=1 and ihi=n, and a vector X, compute: Y=(Product of Householder reflexions) X.

**Parameters**

<i>A</i>	matrix computed in DGEHRD (with ilo=1 and ihi=n).
<i>tau</i>	parameters computed in DGEHRD.
<i>X</i>	(IN).
<i>Y</i>	result.
<i>direct</i>	if true apply Householder reflexions from 1 to n-2 otherwise from n-2 down to 1.

**Note**

that we use dgehrd (in [Matrices.hpp](#)) to reduce the Jacobian matrix to Hessenberg form. We must adapt the computation to the output of this routine.

**8.9.2.4** `template<int n> void solh ( const fortranArray< n > & A, const int ip[], fortranVector & B )`

Solution of a linear system where the matrix A was computed in dech.

**Parameters**

<i>A</i>	the matrix computed in dech (IN)
<i>ip</i>	the pivots obtained in dech. (IN)
<i>B</i>	the RHS on entry, the solution at the end (IN/OUT).

**Note**

this is a transcription of the SOLH routine in H.& W. decsol.

**8.9.2.5** `template<int n> void solhc ( const fortranArray< n > & Ar, const fortranArray< n > & Ai, const int ip[], fortranVector & Br, fortranVector & Bi )`

Solution of a linear system where the matrix A was computed in dech.

**Parameters**

<i>Ar</i>	matrix computed in dechc (IN)
<i>Ai</i>	matrix computed in dechc (IN)
<i>ip</i>	the pivots obtained in dech. (IN)
<i>Br</i>	the RHS on entry (real part) the solution at the end (IN/OUT).

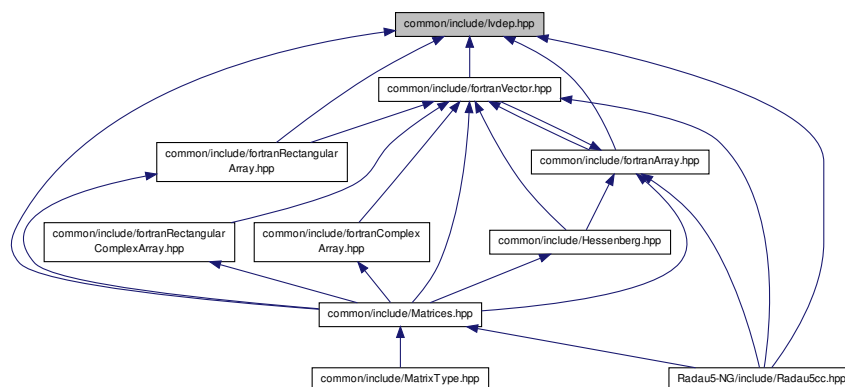
$Bi$	the RHS on entry (imag. part) the solution at the end (IN/OUT).
------	---

**Note**

this is a C++ transcription of SOLHC (in H.&W. fortran code decsol)

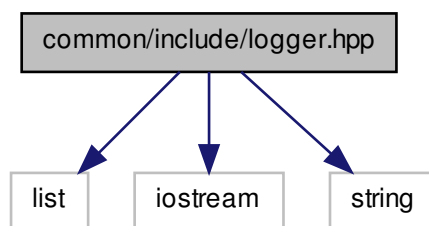
**8.10 common/include/lvdep.hpp File Reference**

This graph shows which files directly or indirectly include this file:

**8.11 common/include/logger.hpp File Reference**

```
#include <list>
#include <iostream>
#include <string>
```

Include dependency graph for logger.hpp:

**Classes**

- class `odes::logger`  
For logging events.
- struct `odes::logger::event`

## Namespaces

- [odes](#)

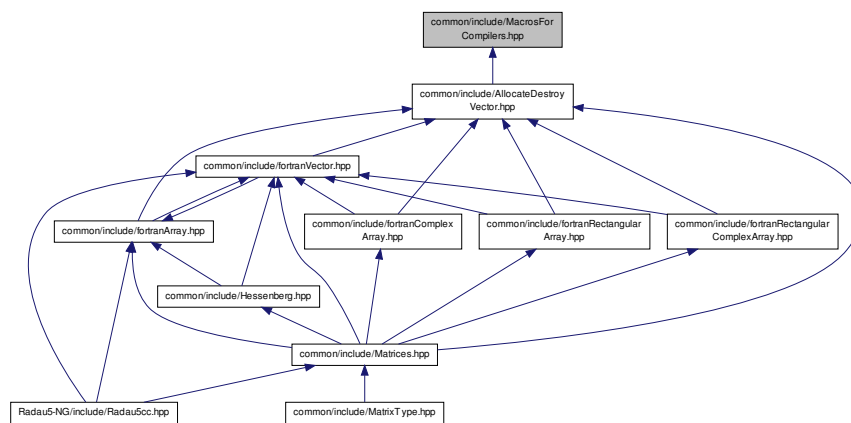
*Compute all matrix operations.*

## Enumerations

- enum [odes::eventType](#) {  
[odes::success](#) =0, [odes::rejectedStep](#), [odes::NewtonFailed](#), [odes::NewtonWillNotConverge](#),  
[odes::changedH](#), [odes::all](#) }

## 8.12 common/include/MacrosForCompilers.hpp File Reference

This graph shows which files directly or indirectly include this file:



## Macros

- `#define BLOCK\_FACTOR 8*sizeof(double)`

### 8.12.1 Macro Definition Documentation

#### 8.12.1.1 `#define BLOCK\_FACTOR 8*sizeof(double)`

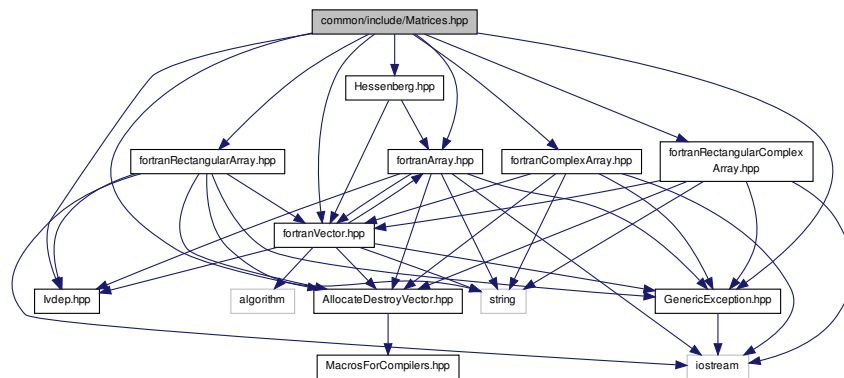
## 8.13 common/include/Matrices.hpp File Reference

```

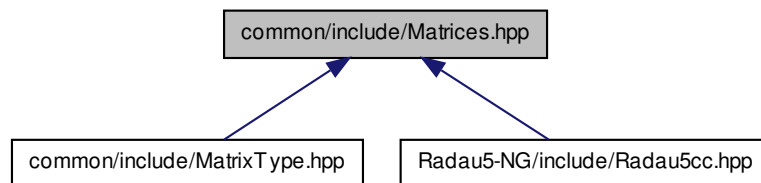
#include "fortranArray.hpp"
#include "fortranComplexArray.hpp"
#include "fortranRectangularArray.hpp"
#include "fortranRectangularComplexArray.hpp"
#include "fortranVector.hpp"
#include "AllocateDestroyVector.hpp"
#include "Hessenberg.hpp"
#include "GenericException.hpp"
#include "Ivdep.hpp"

```

Include dependency graph for Matrices.hpp:



This graph shows which files directly or indirectly include this file:



## Classes

- class `odes::Matrices< full, Hessengerg, n, nsub, nsup >`  
*full matrices.*
- class `odes::Matrices< false, false, n, nsub, nsup >`
- class `odes::Matrices< true, true, n, nsub, nsup >`  
*Full matrices, Hessenberg=true.*
- class `odes::Matrices< false, true, n, nsub, nsup >`  
*Banded matrices, Hessenberg=true: error.*

## Namespaces

- `odes`  
*Compute all matrix operations.*

## Macros

- `#define MAX(x, y) ((x)>(y)?(x):(y))`
- `#define MIN(x, y) ((x)<(y)?(x):(y))`

### 8.13.1 Macro Definition Documentation

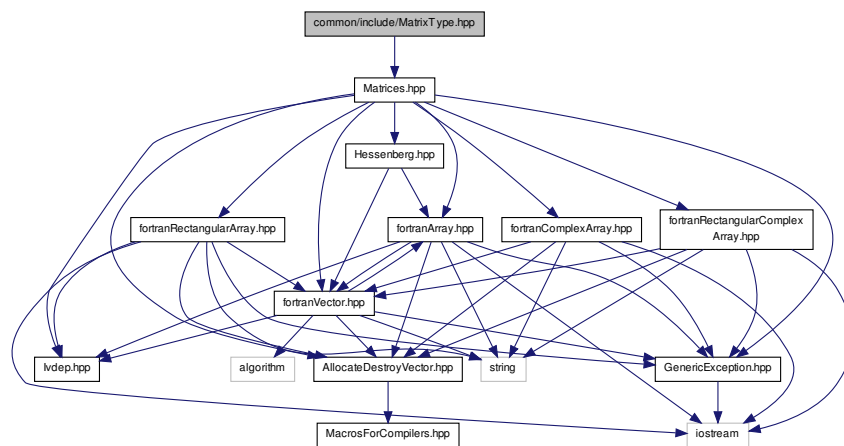
8.13.1.1 `#define MAX( x, y ) ((x)>(y)?(x):(y))`

8.13.1.2 `#define MIN( x, y ) ((x)<(y)?(x):(y))`

## 8.14 common/include/MatrixType.hpp File Reference

```
#include "Matrices.hpp"
```

Include dependency graph for MatrixType.hpp:



### Classes

- struct [odes::Matrixtype< n, nsub, nsup >](#)

### Namespaces

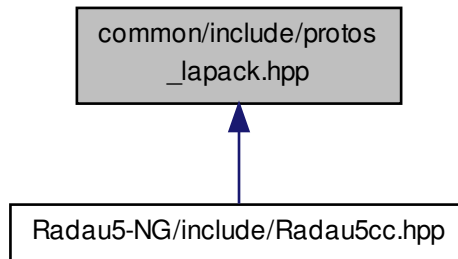
- [odes](#)

*Compute all matrix operations.*



## 8.15 common/include/protos\_lapack.hpp File Reference

This graph shows which files directly or indirectly include this file:



### Namespaces

- [odes](#)

*Compute all matrix operations.*

### Functions

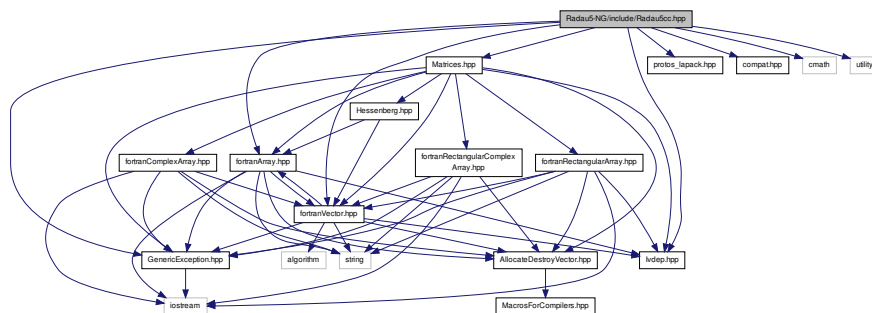
- void [odes::dgetrf\\_](#) (int \*n, int \*m, double \*a, int \*lda, int \*ipiv, int \*info)  
*prototypes for lapack routines.*
- void [odes::zgetrf\\_](#) (int \*n, int \*m, double \*a, int \*lda, int \*ipiv, int \*info)
- void [odes::dgetrs\\_](#) (const char \*s, int \*N, int \*NRHS, double \*A, int \*LDA, int \*IPIV, double \*B, int \*LDB, int \*INFO)
- void [odes::zgetrs\\_](#) (const char \*s, int \*N, int \*NRHS, double \*A, int \*LDA, int \*IPIV, double \*B, int \*LDB, int \*INFO)
- void [odes::dgbtrf\\_](#) (int \*n, int \*m, int \*k1, int \*k2, double \*a, int \*lda, int \*ipiv, int \*info)
- void [odes::zgbtrf\\_](#) (int \*n, int \*m, int \*k1, int \*k2, double \*a, int \*lda, int \*ipiv, int \*info)
- void [odes::dgbtrs\\_](#) (const char \*s, int \*N, int \*k1, int \*k2, int \*NRHS, double \*A, int \*LDA, int \*IPIV, double \*B, int \*LDB, int \*INFO)
- void [odes::zgbtrs\\_](#) (const char \*s, int \*N, int \*k1, int \*k2, int \*NRHS, double \*A, int \*LDA, int \*IPIV, double \*B, int \*LDB, int \*INFO)
- void [odes::dlarnv\\_](#) (int \*idist, int iseed[], int \*n, double \*x)
- void [odes::dgehrd\\_](#) (int \*n, int \*ilo, int \*ihi, double \*a, int \*lda, double tau[], double work[], int \*lwork, int \*info)
- void [odes::dorghr\\_](#) (int \*n, int \*ilo, int \*ihi, double \*a, int \*lda, double tau[], double work[], int \*lwork, int \*info)

## 8.16 Mainpage/mainpage.dox File Reference

## 8.17 Radau5-NG/include/Radau5cc.hpp File Reference

```
#include "GenericException.hpp"
#include "fortranArray.hpp"
#include "fortranVector.hpp"
#include "protos_lapack.hpp"
#include "Matrices.hpp"
#include "compat.hpp"
#include <cmath>
#include <utility>
#include "Ivdep.hpp"
```

Include dependency graph for Radau5cc.hpp:



### Classes

- class [odes::Radau5cc< Fonct >](#)  
A C++ implementation of Radau5.

### Namespaces

- [odes](#)  
Compute all matrix operations.

### Macros

- `#define` [MAX](#)(x, y) ((x)>(y)?(x):(y))
- `#define` [MIN](#)(x, y) ((x)<(y)?(x):(y))
- `#define` [ABS](#)(a) (((a) >= (0.0)) ? (a) : (-a))
- `#define` [SIGN](#)(a) (((a) >= (0.0)) ? (1.) : (-1.))

#### 8.17.1 Macro Definition Documentation

8.17.1.1 `#define` [ABS](#)( a ) (((a) >= (0.0)) ? (a) : (-a))

8.17.1.2 `#define` [MAX](#)( x, y ) ((x)>(y)?(x):(y))

8.17.1.3 `#define` [MIN](#)( x, y ) ((x)<(y)?(x):(y))

8.17.1.4 `#define` [SIGN](#)( a ) (((a) >= (0.0)) ? (1.) : (-1.))

# Index

- ~Matrices
  - odes::Matrices< true, true, n, nsub, nsup >, [48](#)
- ~Radau5cc
  - odes::Radau5cc, [54](#)
- ~fortranArray
  - odes::fortranArray, [17](#)
- ~fortranComplexArray
  - odes::fortranComplexArray, [20](#)
- ~fortranRectangularArray
  - odes::fortranRectangularArray, [22](#)
- ~fortranRectangularComplexArray
  - odes::fortranRectangularComplexArray, [24](#)
- ~fortranVector
  - odes::fortranVector, [27](#)
- ~fortranVectorF
  - odes::fortranVectorF, [30](#)
- ~logger
  - odes::logger, [38](#)
- a\_Y
  - odes::fortranVectorF, [30](#)
- a\_x\_plus\_y
  - odes::fortranVectorF, [30](#)
- ABS
  - Hessenberg.hpp, [75](#)
  - Radau5cc.hpp, [82](#)
- ALPH
  - odes::Radau5cc, [60](#)
- ASSERT\_CONCAT
  - compat.hpp, [67](#)
- ASSERT\_CONCAT\_
  - compat.hpp, [67](#)
- ASSUME\_ALIGNED
  - AllocateDestroyVector.hpp, [66](#)
- addDiag
  - odes::fortranArray, [17](#)
  - odes::fortranRectangularArray, [22](#)
- all
  - odes, [12](#)
- allocDoubleArray
  - AllocateDestroyVector.hpp, [66](#)
- AllocateDestroyVector.hpp
  - ASSUME\_ALIGNED, [66](#)
  - allocDoubleArray, [66](#)
  - destroyDoubleArray, [66](#)
- atol
  - odes::Radau5cc, [60](#)
- BETA
  - odes::Radau5cc, [60](#)
- BLOCK\_FACTOR
  - MacrosForCompilers.hpp, [78](#)
- C1
  - odes::Radau5cc, [60](#)
- C1M1
  - odes::Radau5cc, [60](#)
- C1MC2
  - odes::Radau5cc, [60](#)
- C2
  - odes::Radau5cc, [61](#)
- C2M1
  - odes::Radau5cc, [61](#)
- CNO
  - odes::Radau5cc, [61](#)
- CONT1
  - odes::Radau5cc, [61](#)
- calhes
  - odes::Matrices, [42](#)
  - odes::Matrices< false, false, n, nsub, nsup >, [45](#)
  - odes::Matrices< false, true, n, nsub, nsup >, [47](#)
  - odes::Matrices< true, true, n, nsub, nsup >, [48](#)
  - odes::Radau5cc, [61](#)
- caljac
  - odes::Radau5cc, [61](#)
- cc
  - odes::fortranRectangularArray, [23](#)
  - odes::fortranRectangularComplexArray, [25](#)
- cfac
  - odes::Radau5cc, [61](#)
- changedH
  - odes, [12](#)
- Ci
  - odes::Matrices< true, true, n, nsub, nsup >, [48](#)
- clear
  - odes::logger, [39](#)
- common/include/AllocateDestroyVector.hpp, [65](#)
- common/include/GenericException.hpp, [73](#)
- common/include/Hessenberg.hpp, [74](#)
- common/include/lvdep.hpp, [77](#)
- common/include/MacrosForCompilers.hpp, [78](#)
- common/include/Matrices.hpp, [78](#)
- common/include/MatrixType.hpp, [80](#)
- common/include/compat.hpp, [66](#)
- common/include/fortranArray.hpp, [67](#)
- common/include/fortranComplexArray.hpp, [68](#)
- common/include/fortranRectangularArray.hpp, [69](#)
- common/include/fortranRectangularComplexArray.hpp, [70](#)
- common/include/fortranVector.hpp, [72](#)

- common/include/logger.hpp, 77
- common/include/protos\_lapack.hpp, 81
- compat
  - ct\_assert, 15
- compat< full, Hessenberg >, 15
- compat.hpp
  - ASSERT\_CONCAT, 67
  - ASSERT\_CONCAT\_, 67
  - ct\_assert, 67
- ComputeJacobianNumerically
  - odes::Radau5cc, 61
- converged
  - odes::Radau5cc, 54
- Cr
  - odes::Matrices< true, true, n, nsub, nsup >, 49
- ct\_assert
  - compat, 15
  - compat.hpp, 67
  - odes::Radau5cc, 54
- d2
  - odes::fortranComplexArray, 21
- DD1
  - odes::Radau5cc, 61
- DD2
  - odes::Radau5cc, 61
- DD3
  - odes::Radau5cc, 61
- dech
  - Hessenberg.hpp, 75
- dechc
  - Hessenberg.hpp, 75
- decomc
  - odes::Matrices, 40
  - odes::Matrices< false, false, n, nsub, nsup >, 44
  - odes::Matrices< false, true, n, nsub, nsup >, 46
  - odes::Matrices< true, true, n, nsub, nsup >, 48
  - odes::Radau5cc, 54
- decomr
  - odes::Matrices, 40
  - odes::Matrices< false, false, n, nsub, nsup >, 44
  - odes::Matrices< false, true, n, nsub, nsup >, 46
  - odes::Matrices< true, true, n, nsub, nsup >, 48
  - odes::Radau5cc, 54
- deletable
  - odes::fortranVector, 28
  - odes::fortranVectorF, 35
- destroyDoubleArray
  - AllocateDestroyVector.hpp, 66
- dgbtrf\_
  - odes, 12
- dgbtrs\_
  - odes, 12
- dgehrd\_
  - odes, 12
- dgetrf\_
  - odes, 12
- dgetrs\_
  - odes, 12
- didNotConverge
  - odes::Radau5cc, 54
- dlarnv\_
  - odes, 13
- doNewtonMatrices
  - odes::Radau5cc, 56
- doScal
  - odes::Radau5cc, 56
- dorghr\_
  - odes, 13
- dyno
  - odes::Radau5cc, 61
- dynold
  - odes::Radau5cc, 61
- E1
  - odes::Matrices, 42
  - odes::Matrices< false, false, n, nsub, nsup >, 45
  - odes::Matrices< true, true, n, nsub, nsup >, 49
- E2I
  - odes::Matrices< true, true, n, nsub, nsup >, 49
- E2R
  - odes::Matrices, 42
  - odes::Matrices< false, false, n, nsub, nsup >, 45
  - odes::Matrices< true, true, n, nsub, nsup >, 49
- equal\_minus
  - odes::fortranArray, 17
  - odes::fortranRectangularArray, 22
- erracc
  - odes::Radau5cc, 61
- estrاد
  - odes::Radau5cc, 56
- event
  - odes::logger::event, 15
- eventType
  - odes, 12
- F
  - odes::Radau5cc, 61
- F1
  - odes::Radau5cc, 61
- F2
  - odes::Radau5cc, 61
- F3
  - odes::Radau5cc, 61
- faccon
  - odes::Radau5cc, 61
- facI
  - odes::Radau5cc, 61
- facr
  - odes::Radau5cc, 61
- Fi
  - odes::Matrices< true, true, n, nsub, nsup >, 49
- first
  - odes::Radau5cc, 61
- firstAcceptedStep
  - odes::Radau5cc, 61
- fnwt
  - odes::Radau5cc, 61

- fortranArray
  - odes::fortranArray, [17](#)
  - odes::fortranVector, [28](#)
- fortranComplexArray
  - odes::fortranComplexArray, [20](#)
- fortranRectangularArray
  - odes::fortranRectangularArray, [22](#)
- fortranRectangularArray.hpp
  - MAX, [70](#)
  - MIN, [70](#)
- fortranRectangularComplexArray
  - odes::fortranRectangularComplexArray, [24](#)
- fortranRectangularComplexArray.hpp
  - MAX, [72](#)
  - MIN, [72](#)
- fortranVector
  - odes::fortranVector, [27](#)
- fortranVectorF
  - odes::fortranVector, [28](#)
  - odes::fortranVectorF, [30](#)
- Fr
  - odes::Matrices< true, true, n, nsub, nsup >, [49](#)
- Full
  - odes::Matrixtype, [49](#)
- GenericException, [36](#)
  - GenericException, [37](#), [38](#)
  - GenericException, [37](#), [38](#)
- get\_size
  - odes::fortranVector, [27](#)
  - odes::fortranVectorF, [31](#)
- getLastTimeStep
  - odes::Radau5cc, [56](#)
- getNJac
  - odes::Radau5cc, [57](#)
- getNacpt
  - odes::Radau5cc, [56](#)
- getNdec
  - odes::Radau5cc, [56](#)
- getNewt
  - odes::Radau5cc, [56](#)
- getNfoncCalled
  - odes::Radau5cc, [56](#)
- getNrejt
  - odes::Radau5cc, [57](#)
- getNstep
  - odes::Radau5cc, [57](#)
- getfirstAcceptedStep
  - odes::Radau5cc, [56](#)
- gustafssonTest
  - odes::Radau5cc, [61](#)
- h
  - odes::Radau5cc, [62](#)
- hacc
  - odes::Radau5cc, [62](#)
- Hessenberg
  - odes::Radau5cc, [62](#)
- Hessenberg.hpp
  - ABS, [75](#)
  - dech, [75](#)
  - dechc, [75](#)
  - householder, [76](#)
  - MIN, [75](#)
  - solh, [76](#)
  - solhc, [76](#)
- hhfac
  - odes::Radau5cc, [62](#)
- hmax
  - odes::Radau5cc, [62](#)
- hmaxn
  - odes::Radau5cc, [62](#)
- hold
  - odes::Radau5cc, [62](#)
- hopt
  - odes::Radau5cc, [62](#)
- householder
  - Hessenberg.hpp, [76](#)
- lbegin
  - odes::Matrices, [42](#)
  - odes::Matrices< false, false, n, nsub, nsup >, [44](#)
  - odes::Matrices< false, true, n, nsub, nsup >, [46](#)
  - odes::Matrices< true, true, n, nsub, nsup >, [48](#)
  - odes::Radau5cc, [57](#)
- lend
  - odes::Matrices, [42](#)
  - odes::Matrices< false, false, n, nsub, nsup >, [44](#)
  - odes::Matrices< false, true, n, nsub, nsup >, [46](#)
  - odes::Matrices< true, true, n, nsub, nsup >, [48](#)
  - odes::Radau5cc, [57](#)
- Im
  - odes::fortranComplexArray, [20](#)
  - odes::fortranRectangularComplexArray, [25](#)
- ipi
  - odes::Matrices< true, true, n, nsub, nsup >, [49](#)
- ipivc
  - odes::Matrices, [42](#)
  - odes::Matrices< false, false, n, nsub, nsup >, [45](#)
- ipivr
  - odes::Matrices, [42](#)
  - odes::Matrices< false, false, n, nsub, nsup >, [45](#)
- ipr
  - odes::Matrices< true, true, n, nsub, nsup >, [49](#)
- Jac
  - odes::Radau5cc, [62](#)
- Jacobian
  - odes::Radau5cc, [57](#)
- klku
  - odes::fortranRectangularArray, [23](#)
  - odes::fortranRectangularComplexArray, [25](#)
- L
  - odes::logger, [39](#)
- l1
  - odes::fortranRectangularArray, [23](#)

- odes::fortranRectangularComplexArray, 25
- last
  - odes::Radau5cc, 62
- lc2
  - odes::fortranVectorF, 31
- lc3
  - odes::fortranVectorF, 31
- lc4
  - odes::fortranVectorF, 31
- lc5
  - odes::fortranVectorF, 32
- ldab
  - odes::fortranRectangularArray, 23
  - odes::fortranRectangularComplexArray, 25
  - odes::Matrices< false, false, n, nsub, nsup >, 45
- logger
  - odes::logger, 38
- lwork
  - odes::Matrices< true, true, n, nsub, nsup >, 49
- MAX
  - fortranRectangularArray.hpp, 70
  - fortranRectangularComplexArray.hpp, 72
  - Matrices.hpp, 80
  - Radau5cc.hpp, 82
- MIN
  - fortranRectangularArray.hpp, 70
  - fortranRectangularComplexArray.hpp, 72
  - Hessenberg.hpp, 75
  - Matrices.hpp, 80
  - Radau5cc.hpp, 82
- MacrosForCompilers.hpp
  - BLOCK\_FACTOR, 78
- Mainpage/mainpage.dox, 81
- Matrices
  - odes::Matrices, 40
  - odes::Matrices< false, false, n, nsub, nsup >, 44
  - odes::Matrices< false, true, n, nsub, nsup >, 46
  - odes::Matrices< true, true, n, nsub, nsup >, 48
- Matrices.hpp
  - MAX, 80
  - MIN, 80
- Matrix
  - odes::Matrixtype, 49
- MatrixComplex
  - odes::Matrices, 40
  - odes::Matrices< false, false, n, nsub, nsup >, 44
  - odes::Matrices< false, true, n, nsub, nsup >, 46
  - odes::Matrices< true, true, n, nsub, nsup >, 48
  - odes::Radau5cc, 54
- MatrixFull
  - odes::Radau5cc, 62
- MatrixReal
  - odes::Matrices, 40
  - odes::Matrices< false, false, n, nsub, nsup >, 44
  - odes::Matrices< false, true, n, nsub, nsup >, 46
  - odes::Matrices< true, true, n, nsub, nsup >, 48
  - odes::Radau5cc, 54
- n
  - odes::Radau5cc, 62
- n2
  - odes::fortranArray, 19
- naccpt
  - odes::Radau5cc, 62
- ndec
  - odes::Radau5cc, 62
- newt
  - odes::Radau5cc, 62
- Newton
  - odes::Radau5cc, 57
- NewtonFailed
  - odes, 12
- NewtonWillNotConverge
  - odes, 12
- NewtonIterationTest
  - odes::Radau5cc, 58
- nfonccalled
  - odes::Radau5cc, 62
- ninf
  - odes::Radau5cc, 62
- nit
  - odes::Radau5cc, 62
- njac
  - odes::Radau5cc, 62
- nmax
  - odes::Radau5cc, 62
- nreject
  - odes::Radau5cc, 62
- nstep
  - odes::Radau5cc, 62
- nsup
  - odes::Radau5cc, 62
- odes, 11
  - all, 12
  - changedH, 12
  - dgbtrf\_, 12
  - dgbtrs\_, 12
  - dgehrd\_, 12
  - dgetrf\_, 12
  - dgetrs\_, 12
  - dlarnv\_, 13
  - dorghr\_, 13
  - eventType, 12
  - NewtonFailed, 12
  - NewtonWillNotConverge, 12
  - rejectedStep, 12
  - success, 12
  - zgbtrf\_, 13
  - zgbtrs\_, 13
  - zgetrf\_, 13
  - zgetrs\_, 13
- odes::Radau5cc
  - converged, 54
  - didNotConverge, 54
  - willNotConverge, 54
- odes::Matrices

- calhes, [42](#)
- decomc, [40](#)
- decomr, [40](#)
- E1, [42](#)
- E2R, [42](#)
- lbegin, [42](#)
- lend, [42](#)
- ipivc, [42](#)
- ipivr, [42](#)
- Matrices, [40](#)
- MatrixComplex, [40](#)
- MatrixReal, [40](#)
- solvecomplex, [42](#)
- solvereal, [42](#)
- Z2N, [43](#)
- odes::Matrices< false, false, n, nsub, nsup >, [43](#)
  - calhes, [45](#)
  - decomc, [44](#)
  - decomr, [44](#)
  - E1, [45](#)
  - E2R, [45](#)
  - lbegin, [44](#)
  - lend, [44](#)
  - ipivc, [45](#)
  - ipivr, [45](#)
  - ldab, [45](#)
  - Matrices, [44](#)
  - MatrixComplex, [44](#)
  - MatrixReal, [44](#)
  - solvecomplex, [44](#)
  - solvereal, [45](#)
  - Z2N, [45](#)
- odes::Matrices< false, true, n, nsub, nsup >, [45](#)
  - calhes, [47](#)
  - decomc, [46](#)
  - decomr, [46](#)
  - lbegin, [46](#)
  - lend, [46](#)
  - Matrices, [46](#)
  - MatrixComplex, [46](#)
  - MatrixReal, [46](#)
  - solvecomplex, [46](#)
  - solvereal, [46](#)
- odes::Matrices< full, Hessengerg, n, nsub, nsup >, [39](#)
- odes::Matrices< true, true, n, nsub, nsup >, [47](#)
  - ~Matrices, [48](#)
  - calhes, [48](#)
  - Ci, [48](#)
  - Cr, [49](#)
  - decomc, [48](#)
  - decomr, [48](#)
  - E1, [49](#)
  - E2I, [49](#)
  - E2R, [49](#)
  - Fi, [49](#)
  - Fr, [49](#)
  - lbegin, [48](#)
  - lend, [48](#)
  - ipi, [49](#)
  - ipr, [49](#)
  - lwork, [49](#)
  - Matrices, [48](#)
  - MatrixComplex, [48](#)
  - MatrixReal, [48](#)
  - solvecomplex, [48](#)
  - solvereal, [48](#)
  - tau, [49](#)
  - work, [49](#)
- odes::Matrixtype
  - Full, [49](#)
  - Matrix, [49](#)
- odes::Matrixtype< n, nsub, nsup >, [49](#)
- odes::Radau5cc
  - ~Radau5cc, [54](#)
  - ALPH, [60](#)
  - atol, [60](#)
  - BETA, [60](#)
  - C1, [60](#)
  - C1M1, [60](#)
  - C1MC2, [60](#)
  - C2, [61](#)
  - C2M1, [61](#)
  - CNO, [61](#)
  - CONT1, [61](#)
  - calhes, [61](#)
  - caljac, [61](#)
  - cfac, [61](#)
  - ComputeJacobianNumerically, [61](#)
  - ct\_assert, [54](#)
  - DD1, [61](#)
  - DD2, [61](#)
  - DD3, [61](#)
  - decomc, [54](#)
  - decomr, [54](#)
  - doNewtonMatrices, [56](#)
  - doScal, [56](#)
  - dyno, [61](#)
  - dynold, [61](#)
  - erracc, [61](#)
  - estrاد, [56](#)
  - F, [61](#)
  - F1, [61](#)
  - F2, [61](#)
  - F3, [61](#)
  - faccon, [61](#)
  - facl, [61](#)
  - facr, [61](#)
  - first, [61](#)
  - firstAcceptedStep, [61](#)
  - fnewt, [61](#)
  - getLastTimeStep, [56](#)
  - getNJac, [57](#)
  - getNaccpt, [56](#)
  - getNdec, [56](#)
  - getNewt, [56](#)
  - getNfoncCalled, [56](#)

[getNreject](#), 57  
[getNstep](#), 57  
[getfirstAcceptedStep](#), 56  
[gustafssonTest](#), 61  
[h](#), 62  
[hacc](#), 62  
[Hessenberg](#), 62  
[hhfac](#), 62  
[hmax](#), 62  
[hmaxn](#), 62  
[hold](#), 62  
[hopt](#), 62  
[lbegin](#), 57  
[lend](#), 57  
[Jac](#), 62  
[Jacobian](#), 57  
[last](#), 62  
[MatrixComplex](#), 54  
[MatrixFull](#), 62  
[MatrixReal](#), 54  
[n](#), 62  
[naccpt](#), 62  
[ndec](#), 62  
[newt](#), 62  
[Newton](#), 57  
[NewtonIterationTest](#), 58  
[nfonccalled](#), 62  
[ninf](#), 62  
[nit](#), 62  
[njac](#), 62  
[nmax](#), 62  
[nreject](#), 62  
[nstep](#), 62  
[nsup](#), 62  
[onlyOneStep](#), 62  
[operator\(\)](#), 58  
[OutNewtonLoop](#), 54  
[performOnlyOneStep](#), 58  
[posneg](#), 62  
[quot1](#), 62  
[quot2](#), 62  
[Radau5cc](#), 54  
[reject](#), 63  
[rhs](#), 58  
[rtol](#), 63  
[SQ6](#), 63  
[safe](#), 63  
[sameTestValue](#), 63  
[save1](#), 63  
[save2](#), 63  
[save3](#), 63  
[scal](#), 63  
[setFacrFacI](#), 58  
[setFnewt](#), 59  
[setGustafssonTest](#), 59  
[setMaxIterationsNewton](#), 59  
[setNmax](#), 59  
[setRecomputeJacobianTreshold](#), 59  
[setSafe](#), 59  
[setTestPolicy](#), 60  
[seth](#), 59  
[slvrad](#), 60  
[solvecomplex](#), 60  
[solvereal](#), 60  
[startn](#), 63  
[T11](#), 63  
[T12](#), 63  
[T13](#), 63  
[T21](#), 63  
[T22](#), 63  
[T23](#), 63  
[T31](#), 63  
[TI11](#), 63  
[TI12](#), 63  
[TI13](#), 63  
[TI21](#), 63  
[TI22](#), 63  
[TI23](#), 63  
[TI31](#), 63  
[TI32](#), 63  
[TI33](#), 64  
[thet](#), 63  
[theta](#), 63  
[thqold](#), 63  
[tolst](#), 64  
[U1](#), 64  
[uround](#), 64  
[xend](#), 64  
[xold](#), 64  
[xph](#), 64  
[Z1](#), 64  
[Z1T](#), 64  
[Z2](#), 64  
[Z2T](#), 64  
[Z3](#), 64  
[Z3T](#), 64  
[odes::Radau5cc< Fonct >](#), 50  
[odes::fortranArray](#)  
     ~fortranArray, 17  
     addDiag, 17  
     equal\_minus, 17  
     fortranArray, 17  
     n2, 19  
     operator(), 17  
     operator&, 17  
     print, 19  
     x, 19  
[odes::fortranArray< n >](#), 16  
[odes::fortranComplexArray](#)  
     ~fortranComplexArray, 20  
     d2, 21  
     fortranComplexArray, 20  
     Im, 20  
     operator&, 20  
     print, 20  
     Re, 20



- set, 20
- size, 21
- x, 21
- odes::fortranComplexArray< n >, 19
- odes::fortranRectangularArray
  - ~fortranRectangularArray, 22
  - addDiag, 22
  - cc, 23
  - equal\_minus, 22
  - fortranRectangularArray, 22
  - klku, 23
  - l1, 23
  - ldab, 23
  - operator(), 22, 23
  - operator&, 22
  - print, 23
  - size, 23
  - x, 23
- odes::fortranRectangularArray< n, kl, ku >, 21
- odes::fortranRectangularComplexArray
  - ~fortranRectangularComplexArray, 24
  - cc, 25
  - fortranRectangularComplexArray, 24
  - Im, 25
  - klku, 25
  - l1, 25
  - ldab, 25
  - operator&, 25
  - print, 25
  - Re, 25
  - set, 25
  - size, 26
  - x, 26
- odes::fortranRectangularComplexArray< n, kl, ku >, 24
- odes::fortranVector, 26
  - ~fortranVector, 27
  - deletable, 28
  - fortranArray, 28
  - fortranVector, 27
  - fortranVectorF, 28
  - get\_size, 27
  - operator(), 27, 28
  - operator+=, 28
  - operator=, 28
  - operator&, 27
  - print, 28
  - setsize, 28
  - size, 29
  - x, 29
- odes::fortranVectorF
  - ~fortranVectorF, 30
  - a\_Y, 30
  - a\_x\_plus\_y, 30
  - deletable, 35
  - fortranVectorF, 30
  - get\_size, 31
  - lc2, 31
  - lc3, 31
  - lc4, 31
  - lc5, 32
  - operator(), 32
  - operator+=, 32, 33
  - operator=, 33
  - operator&, 32
  - print, 33
  - put, 33
  - setsize, 34
  - setsum2, 34
  - size, 36
  - sum, 34
  - switchadr, 34
  - x, 36
  - x\_lc2, 35
  - x\_lc3, 35
  - x\_lc4, 35
  - y, 36
- odes::fortranVectorF< n >, 29
- odes::logger, 38
  - ~logger, 38
  - clear, 39
  - L, 39
  - logger, 38
  - print, 39
  - put, 39
- odes::logger::event, 15
  - event, 15
  - step, 15
  - TheEvent, 16
  - time, 16
  - value, 16
- onlyOneStep
  - odes::Radau5cc, 62
- operator()
  - odes::fortranArray, 17
  - odes::fortranRectangularArray, 22, 23
  - odes::fortranVector, 27, 28
  - odes::fortranVectorF, 32
  - odes::Radau5cc, 58
- operator+=
  - odes::fortranVector, 28
  - odes::fortranVectorF, 32, 33
- operator=
  - odes::fortranVector, 28
  - odes::fortranVectorF, 33
- operator&
  - odes::fortranArray, 17
  - odes::fortranComplexArray, 20
  - odes::fortranRectangularArray, 22
  - odes::fortranRectangularComplexArray, 25
  - odes::fortranVector, 27
  - odes::fortranVectorF, 32
- OutNewtonLoop
  - odes::Radau5cc, 54
- performOnlyOneStep
  - odes::Radau5cc, 58
- posneg

- odes::Radau5cc, 62
- print
  - odes::fortranArray, 19
  - odes::fortranComplexArray, 20
  - odes::fortranRectangularArray, 23
  - odes::fortranRectangularComplexArray, 25
  - odes::fortranVector, 28
  - odes::fortranVectorF, 33
  - odes::logger, 39
- put
  - odes::fortranVectorF, 33
  - odes::logger, 39
- quot1
  - odes::Radau5cc, 62
- quot2
  - odes::Radau5cc, 62
- Radau5-NG/include/Radau5cc.hpp, 82
- Radau5cc
  - odes::Radau5cc, 54
- Radau5cc.hpp
  - ABS, 82
  - MAX, 82
  - MIN, 82
  - SIGN, 82
- Re
  - odes::fortranComplexArray, 20
  - odes::fortranRectangularComplexArray, 25
- reject
  - odes::Radau5cc, 63
- rejectedStep
  - odes, 12
- rhs
  - odes::Radau5cc, 58
- rtol
  - odes::Radau5cc, 63
- SIGN
  - Radau5cc.hpp, 82
- SQ6
  - odes::Radau5cc, 63
- safe
  - odes::Radau5cc, 63
- sameTestValue
  - odes::Radau5cc, 63
- save1
  - odes::Radau5cc, 63
- save2
  - odes::Radau5cc, 63
- save3
  - odes::Radau5cc, 63
- scal
  - odes::Radau5cc, 63
- set
  - odes::fortranComplexArray, 20
  - odes::fortranRectangularComplexArray, 25
- setFacrFacI
  - odes::Radau5cc, 58
- setFnewt
  - odes::Radau5cc, 59
- setGustafssonTest
  - odes::Radau5cc, 59
- setMaxIterationsNewton
  - odes::Radau5cc, 59
- setNmax
  - odes::Radau5cc, 59
- setRecomputeJacobianTreshold
  - odes::Radau5cc, 59
- setSafe
  - odes::Radau5cc, 59
- setTestPolicy
  - odes::Radau5cc, 60
- seth
  - odes::Radau5cc, 59
- setsize
  - odes::fortranVector, 28
  - odes::fortranVectorF, 34
- setsum2
  - odes::fortranVectorF, 34
- size
  - odes::fortranComplexArray, 21
  - odes::fortranRectangularArray, 23
  - odes::fortranRectangularComplexArray, 26
  - odes::fortranVector, 29
  - odes::fortranVectorF, 36
- slvrad
  - odes::Radau5cc, 60
- solh
  - Hessenberg.hpp, 76
- solhc
  - Hessenberg.hpp, 76
- solvecomplex
  - odes::Matrices, 42
  - odes::Matrices< false, false, n, nsub, nsup >, 44
  - odes::Matrices< false, true, n, nsub, nsup >, 46
  - odes::Matrices< true, true, n, nsub, nsup >, 48
  - odes::Radau5cc, 60
- solvereal
  - odes::Matrices, 42
  - odes::Matrices< false, false, n, nsub, nsup >, 45
  - odes::Matrices< false, true, n, nsub, nsup >, 46
  - odes::Matrices< true, true, n, nsub, nsup >, 48
  - odes::Radau5cc, 60
- startn
  - odes::Radau5cc, 63
- step
  - odes::logger::event, 15
- success
  - odes, 12
- sum
  - odes::fortranVectorF, 34
- switchadr
  - odes::fortranVectorF, 34
- T11
  - odes::Radau5cc, 63
- T12

- odes::Radau5cc, 63
- T13
  - odes::Radau5cc, 63
- T21
  - odes::Radau5cc, 63
- T22
  - odes::Radau5cc, 63
- T23
  - odes::Radau5cc, 63
- T31
  - odes::Radau5cc, 63
- TI11
  - odes::Radau5cc, 63
- TI12
  - odes::Radau5cc, 63
- TI13
  - odes::Radau5cc, 63
- TI21
  - odes::Radau5cc, 63
- TI22
  - odes::Radau5cc, 63
- TI23
  - odes::Radau5cc, 63
- TI31
  - odes::Radau5cc, 63
- TI32
  - odes::Radau5cc, 63
- TI33
  - odes::Radau5cc, 64
- tau
  - odes::Matrices< true, true, n, nsub, nsup >, 49
- TheEvent
  - odes::logger::event, 16
- thet
  - odes::Radau5cc, 63
- theta
  - odes::Radau5cc, 63
- thqold
  - odes::Radau5cc, 63
- time
  - odes::logger::event, 16
- tolst
  - odes::Radau5cc, 64
- U1
  - odes::Radau5cc, 64
- around
  - odes::Radau5cc, 64
- value
  - odes::logger::event, 16
- willNotConverge
  - odes::Radau5cc, 54
- work
  - odes::Matrices< true, true, n, nsub, nsup >, 49
- x
  - odes::fortranArray, 19
- odes::fortranComplexArray, 21
  - odes::fortranRectangularArray, 23
  - odes::fortranRectangularComplexArray, 26
  - odes::fortranVector, 29
  - odes::fortranVectorF, 36
- x\_lc2
  - odes::fortranVectorF, 35
- x\_lc3
  - odes::fortranVectorF, 35
- x\_lc4
  - odes::fortranVectorF, 35
- xend
  - odes::Radau5cc, 64
- xold
  - odes::Radau5cc, 64
- xph
  - odes::Radau5cc, 64
- y
  - odes::fortranVectorF, 36
- Z1
  - odes::Radau5cc, 64
- Z1T
  - odes::Radau5cc, 64
- Z2
  - odes::Radau5cc, 64
- Z2N
  - odes::Matrices, 43
  - odes::Matrices< false, false, n, nsub, nsup >, 45
- Z2T
  - odes::Radau5cc, 64
- Z3
  - odes::Radau5cc, 64
- Z3T
  - odes::Radau5cc, 64
- zgbtrf\_
  - odes, 13
- zgbtrs\_
  - odes, 13
- zgetrf\_
  - odes, 13
- zgetrs\_
  - odes, 13