

Quantum++

v1.0

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Quantum++</b>	<b>1</b>
<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>11</b>
5.1	File List . . . . .	11
<b>6</b>	<b>Namespace Documentation</b>	<b>13</b>
6.1	qpp Namespace Reference . . . . .	13
6.1.1	Detailed Description . . . . .	25
6.1.2	Typedef Documentation . . . . .	25
6.1.2.1	bigint . . . . .	25
6.1.2.2	bra . . . . .	26
6.1.2.3	cmat . . . . .	26
6.1.2.4	cplx . . . . .	26
6.1.2.5	dmat . . . . .	26
6.1.2.6	dyn_col_vect . . . . .	26
6.1.2.7	dyn_mat . . . . .	27
6.1.2.8	dyn_row_vect . . . . .	27

6.1.2.9	idx	27
6.1.2.10	ket	27
6.1.2.11	to_void	28
6.1.3	Function Documentation	28
6.1.3.1	absm()	28
6.1.3.2	abssq() [1/3]	28
6.1.3.3	abssq() [2/3]	29
6.1.3.4	abssq() [3/3]	29
6.1.3.5	adjoint()	29
6.1.3.6	anticomm()	30
6.1.3.7	apply() [1/5]	30
6.1.3.8	apply() [2/5]	31
6.1.3.9	apply() [3/5]	31
6.1.3.10	apply() [4/5]	32
6.1.3.11	apply() [5/5]	32
6.1.3.12	applyCTRL() [1/2]	33
6.1.3.13	applyCTRL() [2/2]	34
6.1.3.14	avg()	34
6.1.3.15	bloch2rho()	35
6.1.3.16	choi2kraus()	35
6.1.3.17	choi2super()	36
6.1.3.18	comm()	36
6.1.3.19	complement()	37
6.1.3.20	compperm()	37
6.1.3.21	concurrence()	37
6.1.3.22	conjugate()	39
6.1.3.23	contfrac2x()	39
6.1.3.24	cor()	40
6.1.3.25	cosm()	40
6.1.3.26	cov()	41

6.1.3.27	<code>cwise()</code>	41
6.1.3.28	<code>det()</code>	41
6.1.3.29	<code>dirsum()</code> [1/4]	42
6.1.3.30	<code>dirsum()</code> [2/4]	42
6.1.3.31	<code>dirsum()</code> [3/4]	43
6.1.3.32	<code>dirsum()</code> [4/4]	43
6.1.3.33	<code>dirsumpow()</code>	44
6.1.3.34	<code>disp()</code> [1/5]	44
6.1.3.35	<code>disp()</code> [2/5]	45
6.1.3.36	<code>disp()</code> [3/5]	45
6.1.3.37	<code>disp()</code> [4/5]	46
6.1.3.38	<code>disp()</code> [5/5]	46
6.1.3.39	<code>egcd()</code>	47
6.1.3.40	<code>eig()</code>	47
6.1.3.41	<code>entanglement()</code> [1/2]	47
6.1.3.42	<code>entanglement()</code> [2/2]	48
6.1.3.43	<code>entropy()</code> [1/2]	49
6.1.3.44	<code>entropy()</code> [2/2]	49
6.1.3.45	<code>evals()</code>	49
6.1.3.46	<code>evecs()</code>	50
6.1.3.47	<code>expm()</code>	50
6.1.3.48	<code>factors()</code>	51
6.1.3.49	<code>funm()</code>	51
6.1.3.50	<code>gcd()</code> [1/2]	51
6.1.3.51	<code>gcd()</code> [2/2]	52
6.1.3.52	<code>gconcurrence()</code>	52
6.1.3.53	<code>grams()</code> [1/3]	53
6.1.3.54	<code>grams()</code> [2/3]	53
6.1.3.55	<code>grams()</code> [3/3]	54
6.1.3.56	<code>heig()</code>	54

6.1.3.57	<code>hevals()</code>	54
6.1.3.58	<code>hevects()</code>	55
6.1.3.59	<code>inverse()</code>	55
6.1.3.60	<code>invperm()</code>	56
6.1.3.61	<code>ip()</code> [1/2]	56
6.1.3.62	<code>ip()</code> [2/2]	56
6.1.3.63	<code>isprime()</code>	57
6.1.3.64	<code>kraus2choi()</code>	57
6.1.3.65	<code>kraus2super()</code>	58
6.1.3.66	<code>kron()</code> [1/4]	58
6.1.3.67	<code>kron()</code> [2/4]	59
6.1.3.68	<code>kron()</code> [3/4]	59
6.1.3.69	<code>kron()</code> [4/4]	60
6.1.3.70	<code>kronpow()</code>	60
6.1.3.71	<code>lcm()</code> [1/2]	61
6.1.3.72	<code>lcm()</code> [2/2]	61
6.1.3.73	<code>load()</code>	62
6.1.3.74	<code>loadMATLAB()</code> [1/2]	62
6.1.3.75	<code>loadMATLAB()</code> [2/2]	63
6.1.3.76	<code>logdet()</code>	64
6.1.3.77	<code>logm()</code>	64
6.1.3.78	<code>lognegativity()</code> [1/2]	64
6.1.3.79	<code>lognegativity()</code> [2/2]	65
6.1.3.80	<code>marginalX()</code>	65
6.1.3.81	<code>marginalY()</code>	65
6.1.3.82	<code>measure()</code> [1/9]	66
6.1.3.83	<code>measure()</code> [2/9]	66
6.1.3.84	<code>measure()</code> [3/9]	67
6.1.3.85	<code>measure()</code> [4/9]	67
6.1.3.86	<code>measure()</code> [5/9]	68

6.1.3.87	<code>measure()</code> [6/9]	69
6.1.3.88	<code>measure()</code> [7/9]	69
6.1.3.89	<code>measure()</code> [8/9]	70
6.1.3.90	<code>measure()</code> [9/9]	71
6.1.3.91	<code>measure_seq()</code> [1/2]	71
6.1.3.92	<code>measure_seq()</code> [2/2]	72
6.1.3.93	<code>mket()</code> [1/2]	72
6.1.3.94	<code>mket()</code> [2/2]	73
6.1.3.95	<code>modinv()</code>	73
6.1.3.96	<code>modmul()</code>	74
6.1.3.97	<code>modpow()</code>	74
6.1.3.98	<code>mprj()</code> [1/2]	75
6.1.3.99	<code>mprj()</code> [2/2]	75
6.1.3.100	<code>multiidx2n()</code>	76
6.1.3.101	<code>n2multiidx()</code>	76
6.1.3.102	<code>negativity()</code> [1/2]	77
6.1.3.103	<code>negativity()</code> [2/2]	77
6.1.3.104	<code>norm()</code>	78
6.1.3.105	<code>omega()</code>	78
6.1.3.106	<code>operator""_i()</code>	78
6.1.3.107	<code>powm()</code>	79
6.1.3.108	<code>prj()</code>	79
6.1.3.109	<code>prod()</code> [1/3]	80
6.1.3.110	<code>prod()</code> [2/3]	80
6.1.3.111	<code>prod()</code> [3/3]	80
6.1.3.112	<code>ptrace()</code> [1/2]	81
6.1.3.113	<code>ptrace()</code> [2/2]	81
6.1.3.114	<code>ptrace1()</code> [1/2]	82
6.1.3.115	<code>ptrace1()</code> [2/2]	82
6.1.3.116	<code>ptrace2()</code> [1/2]	84

6.1.3.117 ptrace2() [2/2]	84
6.1.3.118 ptranspose() [1/2]	85
6.1.3.119 ptranspose() [2/2]	85
6.1.3.120 qmutualinfo() [1/2]	86
6.1.3.121 qmutualinfo() [2/2]	86
6.1.3.122 rand() [1/5]	87
6.1.3.123 rand() [2/5]	87
6.1.3.124 rand() [3/5]	88
6.1.3.125 rand() [4/5]	88
6.1.3.126 rand() [5/5]	89
6.1.3.127 randH()	89
6.1.3.128 randidx()	90
6.1.3.129 randket()	90
6.1.3.130 randkraus()	90
6.1.3.131 randn() [1/4]	91
6.1.3.132 randn() [2/4]	91
6.1.3.133 randn() [3/4]	92
6.1.3.134 randn() [4/4]	92
6.1.3.135 randperm()	93
6.1.3.136 randprime()	93
6.1.3.137 randprob()	94
6.1.3.138 randrho()	94
6.1.3.139 randU()	94
6.1.3.140 randV()	95
6.1.3.141 renyi() [1/2]	95
6.1.3.142 renyi() [2/2]	96
6.1.3.143 reshape()	96
6.1.3.144 rho2bloch()	97
6.1.3.145 rho2pure()	97
6.1.3.146 save()	98



6.1.3.147 saveMATLAB() [1/2] . . . . .	98
6.1.3.148 saveMATLAB() [2/2] . . . . .	99
6.1.3.149 schatten() . . . . .	99
6.1.3.150 schmidtA() [1/2] . . . . .	100
6.1.3.151 schmidtA() [2/2] . . . . .	100
6.1.3.152 schmidtB() [1/2] . . . . .	100
6.1.3.153 schmidtB() [2/2] . . . . .	101
6.1.3.154 schmidtcoeffs() [1/2] . . . . .	101
6.1.3.155 schmidtcoeffs() [2/2] . . . . .	102
6.1.3.156 schmidtprobs() [1/2] . . . . .	102
6.1.3.157 schmidtprobs() [2/2] . . . . .	103
6.1.3.158 sigma() . . . . .	103
6.1.3.159 sinm() . . . . .	104
6.1.3.160 spectralpowm() . . . . .	104
6.1.3.161 sqrtm() . . . . .	105
6.1.3.162 sum() [1/3] . . . . .	105
6.1.3.163 sum() [2/3] . . . . .	105
6.1.3.164 sum() [3/3] . . . . .	106
6.1.3.165 super2choi() . . . . .	106
6.1.3.166 svals() . . . . .	107
6.1.3.167 svd() . . . . .	107
6.1.3.168 svdU() . . . . .	107
6.1.3.169 svdV() . . . . .	108
6.1.3.170 syspermute() [1/2] . . . . .	108
6.1.3.171 syspermute() [2/2] . . . . .	109
6.1.3.172 trace() . . . . .	109
6.1.3.173 transpose() . . . . .	110
6.1.3.174 tsallis() [1/2] . . . . .	110
6.1.3.175 tsallis() [2/2] . . . . .	110
6.1.3.176 uniform() . . . . .	111

6.1.3.177	<code>var()</code>	111
6.1.3.178	<code>x2contfrac()</code>	112
6.1.4	Variable Documentation	112
6.1.4.1	<code>chop</code>	112
6.1.4.2	<code>ee</code>	112
6.1.4.3	<code>eps</code>	113
6.1.4.4	<code>infty</code>	113
6.1.4.5	<code>maxn</code>	113
6.1.4.6	<code>pi</code>	113
6.2	<code>qpp::exception</code> Namespace Reference	113
6.2.1	Detailed Description	115
6.3	<code>qpp::experimental</code> Namespace Reference	115
6.3.1	Detailed Description	115
6.4	<code>qpp::internal</code> Namespace Reference	115
6.4.1	Detailed Description	116
6.4.2	Function Documentation	116
6.4.2.1	<code>check_cvector()</code>	117
6.4.2.2	<code>check_dims()</code>	117
6.4.2.3	<code>check_dims_match_cvect()</code>	117
6.4.2.4	<code>check_dims_match_mat()</code>	117
6.4.2.5	<code>check_dims_match_rvect()</code>	117
6.4.2.6	<code>check_eq_dims()</code>	117
6.4.2.7	<code>check_matching_sizes()</code>	118
6.4.2.8	<code>check_nonzero_size()</code>	118
6.4.2.9	<code>check_perm()</code>	118
6.4.2.10	<code>check_qubit_cvector()</code>	118
6.4.2.11	<code>check_qubit_matrix()</code>	118
6.4.2.12	<code>check_qubit_rvector()</code>	118
6.4.2.13	<code>check_qubit_vector()</code>	119
6.4.2.14	<code>check_rvector()</code>	119

6.4.2.15	<code>check_square_mat()</code>	119
6.4.2.16	<code>check_subsys_match_dims()</code>	119
6.4.2.17	<code>check_vector()</code>	119
6.4.2.18	<code>dirsum2()</code>	119
6.4.2.19	<code>get_dim_subsys()</code>	120
6.4.2.20	<code>get_num_subsys()</code>	120
6.4.2.21	<code>kron2()</code>	120
6.4.2.22	<code>multiidx2n()</code>	120
6.4.2.23	<code>n2multiidx()</code>	120
6.4.2.24	<code>variadic_vector_emplace()</code> [1/2]	120
6.4.2.25	<code>variadic_vector_emplace()</code> [2/2]	121
6.5	<code>qpp::literals</code> Namespace Reference	121
6.5.1	Function Documentation	121
6.5.1.1	<code>operator"" _bra()</code>	121
6.5.1.2	<code>operator"" _i()</code>	122
6.5.1.3	<code>operator"" _ket()</code>	122
6.5.1.4	<code>operator"" _prj()</code>	122
<b>7</b>	<b>Class Documentation</b>	<b>125</b>
7.1	<code>qpp::Bit_circuit</code> Class Reference	125
7.1.1	Detailed Description	127
7.1.2	Constructor & Destructor Documentation	127
7.1.2.1	<code>Bit_circuit()</code>	127
7.1.3	Member Function Documentation	127
7.1.3.1	<code>CNOT()</code>	127
7.1.3.2	<code>FRED()</code>	127
7.1.3.3	<code>NOT()</code>	128
7.1.3.4	<code>reset()</code>	128
7.1.3.5	<code>SWAP()</code>	128
7.1.3.6	<code>TOF()</code>	129
7.1.3.7	<code>X()</code>	129

7.1.4	Member Data Documentation . . . . .	130
7.1.4.1	gate_count . . . . .	130
7.2	qpp::Codes Class Reference . . . . .	130
7.2.1	Detailed Description . . . . .	131
7.2.2	Member Enumeration Documentation . . . . .	131
7.2.2.1	Type . . . . .	131
7.2.3	Constructor & Destructor Documentation . . . . .	132
7.2.3.1	Codes() . . . . .	132
7.2.3.2	~Codes() . . . . .	132
7.2.4	Member Function Documentation . . . . .	132
7.2.4.1	codeword() . . . . .	132
7.2.5	Friends And Related Function Documentation . . . . .	132
7.2.5.1	internal::Singleton< const Codes > . . . . .	133
7.3	qpp::exception::CustomException Class Reference . . . . .	133
7.3.1	Detailed Description . . . . .	134
7.3.2	Constructor & Destructor Documentation . . . . .	134
7.3.2.1	CustomException() . . . . .	135
7.3.3	Member Function Documentation . . . . .	135
7.3.3.1	type_description() . . . . .	135
7.3.4	Member Data Documentation . . . . .	135
7.3.4.1	what_ . . . . .	135
7.4	qpp::exception::DimsInvalid Class Reference . . . . .	136
7.4.1	Detailed Description . . . . .	137
7.4.2	Member Function Documentation . . . . .	137
7.4.2.1	type_description() . . . . .	137
7.5	qpp::exception::DimsMismatchCvector Class Reference . . . . .	137
7.5.1	Detailed Description . . . . .	139
7.5.2	Member Function Documentation . . . . .	139
7.5.2.1	type_description() . . . . .	139
7.6	qpp::exception::DimsMismatchMatrix Class Reference . . . . .	139

7.6.1	Detailed Description	140
7.6.2	Member Function Documentation	140
7.6.2.1	type_description()	141
7.7	qpp::exception::DimsMismatchRvector Class Reference	141
7.7.1	Detailed Description	142
7.7.2	Member Function Documentation	142
7.7.2.1	type_description()	143
7.8	qpp::exception::DimsMismatchVector Class Reference	143
7.8.1	Detailed Description	144
7.8.2	Member Function Documentation	144
7.8.2.1	type_description()	145
7.9	qpp::exception::DimsNotEqual Class Reference	145
7.9.1	Detailed Description	146
7.9.2	Member Function Documentation	146
7.9.2.1	type_description()	146
7.10	qpp::internal::Display_Impl_ Struct Reference	147
7.10.1	Member Function Documentation	147
7.10.1.1	display_impl_()	147
7.11	qpp::Dynamic_bitset Class Reference	148
7.11.1	Detailed Description	150
7.11.2	Member Typedef Documentation	150
7.11.2.1	storage_type	150
7.11.2.2	value_type	150
7.11.3	Constructor & Destructor Documentation	150
7.11.3.1	Dynamic_bitset()	150
7.11.4	Member Function Documentation	151
7.11.4.1	all()	151
7.11.4.2	any()	151
7.11.4.3	count()	151
7.11.4.4	data()	152

7.11.4.5	<a href="#">display()</a>	152
7.11.4.6	<a href="#">flip()</a> [1/2]	153
7.11.4.7	<a href="#">flip()</a> [2/2]	153
7.11.4.8	<a href="#">get()</a>	153
7.11.4.9	<a href="#">index_()</a>	154
7.11.4.10	<a href="#">none()</a>	154
7.11.4.11	<a href="#">offset_()</a>	154
7.11.4.12	<a href="#">operator!=()</a>	155
7.11.4.13	<a href="#">operator-()</a>	155
7.11.4.14	<a href="#">operator==()</a>	155
7.11.4.15	<a href="#">rand()</a> [1/2]	157
7.11.4.16	<a href="#">rand()</a> [2/2]	157
7.11.4.17	<a href="#">reset()</a> [1/2]	158
7.11.4.18	<a href="#">reset()</a> [2/2]	158
7.11.4.19	<a href="#">set()</a> [1/2]	158
7.11.4.20	<a href="#">set()</a> [2/2]	159
7.11.4.21	<a href="#">size()</a>	159
7.11.4.22	<a href="#">storage_size()</a>	159
7.11.4.23	<a href="#">to_string()</a>	159
7.11.5	<a href="#">Member Data Documentation</a>	160
7.11.5.1	<a href="#">N_</a>	160
7.11.5.2	<a href="#">storage_size_</a>	160
7.11.5.3	<a href="#">v_</a>	160
7.12	<a href="#">qpp::exception::Exception Class Reference</a>	161
7.12.1	<a href="#">Detailed Description</a>	162
7.12.2	<a href="#">Constructor &amp; Destructor Documentation</a>	163
7.12.2.1	<a href="#">Exception()</a>	163
7.12.3	<a href="#">Member Function Documentation</a>	163
7.12.3.1	<a href="#">type_description()</a>	163
7.12.3.2	<a href="#">what()</a>	164

7.12.4	Member Data Documentation . . . . .	164
7.12.4.1	msg_ . . . . .	164
7.12.4.2	where_ . . . . .	164
7.13	qpp::Bit_circuit::Gate_count Struct Reference . . . . .	164
7.13.1	Member Data Documentation . . . . .	164
7.13.1.1	CNOT . . . . .	165
7.13.1.2	FRED . . . . .	165
7.13.1.3	NOT . . . . .	165
7.13.1.4	SWAP . . . . .	165
7.13.1.5	TOF . . . . .	165
7.13.1.6	X . . . . .	165
7.14	qpp::Gates Class Reference . . . . .	166
7.14.1	Detailed Description . . . . .	168
7.14.2	Constructor & Destructor Documentation . . . . .	168
7.14.2.1	Gates() . . . . .	168
7.14.2.2	~Gates() . . . . .	168
7.14.3	Member Function Documentation . . . . .	168
7.14.3.1	CTRL() . . . . .	168
7.14.3.2	expandout() [1/3] . . . . .	169
7.14.3.3	expandout() [2/3] . . . . .	169
7.14.3.4	expandout() [3/3] . . . . .	170
7.14.3.5	Fd() . . . . .	171
7.14.3.6	Id() . . . . .	171
7.14.3.7	Rn() . . . . .	172
7.14.3.8	Xd() . . . . .	172
7.14.3.9	Zd() . . . . .	172
7.14.4	Friends And Related Function Documentation . . . . .	173
7.14.4.1	internal::Singleton< const Gates > . . . . .	173
7.14.5	Member Data Documentation . . . . .	173
7.14.5.1	CNOT . . . . .	173

7.14.5.2	CNOTba	173
7.14.5.3	CZ	173
7.14.5.4	FRED	174
7.14.5.5	H	174
7.14.5.6	Id2	174
7.14.5.7	S	174
7.14.5.8	SWAP	174
7.14.5.9	T	174
7.14.5.10	TOF	175
7.14.5.11	X	175
7.14.5.12	Y	175
7.14.5.13	Z	175
7.15	qpp::IDisplay Class Reference	176
7.15.1	Detailed Description	177
7.15.2	Constructor & Destructor Documentation	177
7.15.2.1	IDisplay() [1/3]	177
7.15.2.2	IDisplay() [2/3]	177
7.15.2.3	IDisplay() [3/3]	177
7.15.2.4	~IDisplay()	177
7.15.3	Member Function Documentation	178
7.15.3.1	display()	178
7.15.3.2	operator=() [1/2]	178
7.15.3.3	operator=() [2/2]	178
7.15.4	Friends And Related Function Documentation	178
7.15.4.1	operator<<	178
7.16	qpp::Init Class Reference	179
7.16.1	Detailed Description	180
7.16.2	Constructor & Destructor Documentation	180
7.16.2.1	Init()	180
7.16.2.2	~Init()	180



7.16.3 Friends And Related Function Documentation . . . . .	180
7.16.3.1 internal::Singleton< const Init > . . . . .	180
7.17 qpp::internal::IOManipEigen Class Reference . . . . .	181
7.17.1 Constructor & Destructor Documentation . . . . .	182
7.17.1.1 IOManipEigen() [1/2] . . . . .	182
7.17.1.2 IOManipEigen() [2/2] . . . . .	182
7.17.2 Member Function Documentation . . . . .	182
7.17.2.1 display() . . . . .	182
7.17.3 Member Data Documentation . . . . .	182
7.17.3.1 A_ . . . . .	183
7.17.3.2 chop_ . . . . .	183
7.18 qpp::internal::IOManipPointer< PointerType > Class Template Reference . . . . .	183
7.18.1 Constructor & Destructor Documentation . . . . .	184
7.18.1.1 IOManipPointer() [1/2] . . . . .	185
7.18.1.2 IOManipPointer() [2/2] . . . . .	185
7.18.2 Member Function Documentation . . . . .	185
7.18.2.1 display() . . . . .	185
7.18.2.2 operator=() . . . . .	185
7.18.3 Member Data Documentation . . . . .	185
7.18.3.1 end_ . . . . .	186
7.18.3.2 N_ . . . . .	186
7.18.3.3 p_ . . . . .	186
7.18.3.4 separator_ . . . . .	186
7.18.3.5 start_ . . . . .	186
7.19 qpp::internal::IOManipRange< InputIterator > Class Template Reference . . . . .	187
7.19.1 Constructor & Destructor Documentation . . . . .	188
7.19.1.1 IOManipRange() [1/2] . . . . .	188
7.19.1.2 IOManipRange() [2/2] . . . . .	188
7.19.2 Member Function Documentation . . . . .	188
7.19.2.1 display() . . . . .	188

7.19.2.2	<code>operator=()</code>	189
7.19.3	Member Data Documentation	189
7.19.3.1	<code>end_</code>	189
7.19.3.2	<code>first_</code>	189
7.19.3.3	<code>last_</code>	189
7.19.3.4	<code>separator_</code>	189
7.19.3.5	<code>start_</code>	189
7.20	<code>qpp::is_complex&lt; T &gt;</code> Struct Template Reference	190
7.20.1	Detailed Description	190
7.21	<code>qpp::is_complex&lt; std::complex&lt; T &gt; &gt;</code> Struct Template Reference	191
7.21.1	Detailed Description	191
7.22	<code>qpp::is_iterable&lt; T, typename &gt;</code> Struct Template Reference	192
7.22.1	Detailed Description	192
7.23	<code>qpp::is_iterable&lt; T, to_void&lt; decltype(std::declval&lt; T &gt;().begin()), decltype(std::declval&lt; T &gt;().end()), typename T::value_type &gt; &gt;</code> Struct Template Reference	193
7.23.1	Detailed Description	194
7.24	<code>qpp::is_matrix_expression&lt; Derived &gt;</code> Struct Template Reference	194
7.24.1	Detailed Description	195
7.25	<code>qpp::make_void&lt; Ts &gt;</code> Struct Template Reference	195
7.25.1	Detailed Description	195
7.25.2	Member Typedef Documentation	195
7.25.2.1	<code>type</code>	195
7.26	<code>qpp::exception::MatrixMismatchSubsys</code> Class Reference	196
7.26.1	Detailed Description	197
7.26.2	Member Function Documentation	197
7.26.2.1	<code>type_description()</code>	197
7.27	<code>qpp::exception::MatrixNotCvector</code> Class Reference	197
7.27.1	Detailed Description	199
7.27.2	Member Function Documentation	199
7.27.2.1	<code>type_description()</code>	199
7.28	<code>qpp::exception::MatrixNotRvector</code> Class Reference	199

7.28.1 Detailed Description . . . . .	200
7.28.2 Member Function Documentation . . . . .	200
7.28.2.1 type_description() . . . . .	201
7.29 qpp::exception::MatrixNotSquare Class Reference . . . . .	201
7.29.1 Detailed Description . . . . .	202
7.29.2 Member Function Documentation . . . . .	202
7.29.2.1 type_description() . . . . .	203
7.30 qpp::exception::MatrixNotSquareNorCvector Class Reference . . . . .	203
7.30.1 Detailed Description . . . . .	204
7.30.2 Member Function Documentation . . . . .	204
7.30.2.1 type_description() . . . . .	205
7.31 qpp::exception::MatrixNotSquareNorRvector Class Reference . . . . .	205
7.31.1 Detailed Description . . . . .	206
7.31.2 Member Function Documentation . . . . .	206
7.31.2.1 type_description() . . . . .	207
7.32 qpp::exception::MatrixNotSquareNorVector Class Reference . . . . .	207
7.32.1 Detailed Description . . . . .	208
7.32.2 Member Function Documentation . . . . .	208
7.32.2.1 type_description() . . . . .	209
7.33 qpp::exception::MatrixNotVector Class Reference . . . . .	209
7.33.1 Detailed Description . . . . .	210
7.33.2 Member Function Documentation . . . . .	210
7.33.2.1 type_description() . . . . .	211
7.34 qpp::exception::NoCodeword Class Reference . . . . .	211
7.34.1 Detailed Description . . . . .	212
7.34.2 Member Function Documentation . . . . .	212
7.34.2.1 type_description() . . . . .	212
7.35 qpp::exception::NotBipartite Class Reference . . . . .	213
7.35.1 Detailed Description . . . . .	214
7.35.2 Member Function Documentation . . . . .	214

7.35.2.1	<code>type_description()</code>	214
7.36	<code>qpp::exception::NotQubitCvector</code> Class Reference	215
7.36.1	Detailed Description	216
7.36.2	Member Function Documentation	216
7.36.2.1	<code>type_description()</code>	217
7.37	<code>qpp::exception::NotQubitMatrix</code> Class Reference	217
7.37.1	Detailed Description	218
7.37.2	Member Function Documentation	218
7.37.2.1	<code>type_description()</code>	219
7.38	<code>qpp::exception::NotQubitRvector</code> Class Reference	219
7.38.1	Detailed Description	220
7.38.2	Member Function Documentation	220
7.38.2.1	<code>type_description()</code>	221
7.39	<code>qpp::exception::NotQubitSubsys</code> Class Reference	221
7.39.1	Detailed Description	222
7.39.2	Member Function Documentation	222
7.39.2.1	<code>type_description()</code>	223
7.40	<code>qpp::exception::NotQubitVector</code> Class Reference	223
7.40.1	Detailed Description	224
7.40.2	Member Function Documentation	224
7.40.2.1	<code>type_description()</code>	225
7.41	<code>qpp::exception::OutOfRange</code> Class Reference	225
7.41.1	Detailed Description	226
7.41.2	Member Function Documentation	226
7.41.2.1	<code>type_description()</code>	226
7.42	<code>qpp::exception::PermInvalid</code> Class Reference	227
7.42.1	Detailed Description	228
7.42.2	Member Function Documentation	228
7.42.2.1	<code>type_description()</code>	228
7.43	<code>qpp::exception::PermMismatchDims</code> Class Reference	229

7.43.1 Detailed Description . . . . .	230
7.43.2 Member Function Documentation . . . . .	230
7.43.2.1 type_description() . . . . .	231
7.44 qpp::RandomDevices Class Reference . . . . .	231
7.44.1 Detailed Description . . . . .	232
7.44.2 Constructor & Destructor Documentation . . . . .	232
7.44.2.1 RandomDevices() . . . . .	233
7.44.2.2 ~RandomDevices() . . . . .	233
7.44.3 Member Function Documentation . . . . .	233
7.44.3.1 get_prng() . . . . .	233
7.44.3.2 load() . . . . .	233
7.44.3.3 save() . . . . .	234
7.44.4 Friends And Related Function Documentation . . . . .	234
7.44.4.1 internal::Singleton< RandomDevices > . . . . .	234
7.44.5 Member Data Documentation . . . . .	234
7.44.5.1 prng_ . . . . .	234
7.44.5.2 rd_ . . . . .	234
7.45 qpp::internal::Singleton< T > Class Template Reference . . . . .	235
7.45.1 Detailed Description . . . . .	235
7.45.2 Constructor & Destructor Documentation . . . . .	236
7.45.2.1 Singleton() [1/2] . . . . .	236
7.45.2.2 Singleton() [2/2] . . . . .	236
7.45.2.3 ~Singleton() . . . . .	236
7.45.3 Member Function Documentation . . . . .	236
7.45.3.1 get_instance() . . . . .	236
7.45.3.2 get_thread_local_instance() . . . . .	236
7.45.3.3 operator=() . . . . .	237
7.46 qpp::exception::SizeMismatch Class Reference . . . . .	237
7.46.1 Detailed Description . . . . .	238
7.46.2 Member Function Documentation . . . . .	238

7.46.2.1	<code>type_description()</code>	238
7.47	<code>qpp::States</code> Class Reference	239
7.47.1	Detailed Description	241
7.47.2	Constructor & Destructor Documentation	241
7.47.2.1	<code>States()</code>	241
7.47.2.2	<code>~States()</code>	241
7.47.3	Member Function Documentation	242
7.47.3.1	<code>jn()</code>	242
7.47.3.2	<code>mes()</code>	242
7.47.3.3	<code>minus()</code>	242
7.47.3.4	<code>one()</code>	243
7.47.3.5	<code>plus()</code>	243
7.47.3.6	<code>zero()</code>	244
7.47.4	Friends And Related Function Documentation	244
7.47.4.1	<code>internal::Singleton&lt; const States &gt;</code>	244
7.47.5	Member Data Documentation	244
7.47.5.1	<code>b00</code>	244
7.47.5.2	<code>b01</code>	244
7.47.5.3	<code>b10</code>	245
7.47.5.4	<code>b11</code>	245
7.47.5.5	<code>GHZ</code>	245
7.47.5.6	<code>pb00</code>	245
7.47.5.7	<code>pb01</code>	245
7.47.5.8	<code>pb10</code>	245
7.47.5.9	<code>pb11</code>	246
7.47.5.10	<code>pGHZ</code>	246
7.47.5.11	<code>pW</code>	246
7.47.5.12	<code>px0</code>	246
7.47.5.13	<code>px1</code>	246
7.47.5.14	<code>py0</code>	246

7.47.5.15 py1 . . . . .	247
7.47.5.16 pz0 . . . . .	247
7.47.5.17 pz1 . . . . .	247
7.47.5.18 W . . . . .	247
7.47.5.19 x0 . . . . .	247
7.47.5.20 x1 . . . . .	247
7.47.5.21 y0 . . . . .	248
7.47.5.22 y1 . . . . .	248
7.47.5.23 z0 . . . . .	248
7.47.5.24 z1 . . . . .	248
7.48 qpp::exception::SubsysMismatchDims Class Reference . . . . .	249
7.48.1 Detailed Description . . . . .	250
7.48.2 Member Function Documentation . . . . .	250
7.48.2.1 type_description() . . . . .	250
7.49 qpp::Timer< T, CLOCK_T > Class Template Reference . . . . .	250
7.49.1 Detailed Description . . . . .	252
7.49.2 Constructor & Destructor Documentation . . . . .	252
7.49.2.1 Timer() [1/3] . . . . .	252
7.49.2.2 Timer() [2/3] . . . . .	252
7.49.2.3 Timer() [3/3] . . . . .	252
7.49.2.4 ~Timer() . . . . .	253
7.49.3 Member Function Documentation . . . . .	253
7.49.3.1 display() . . . . .	253
7.49.3.2 get_duration() . . . . .	253
7.49.3.3 operator=() [1/2] . . . . .	254
7.49.3.4 operator=() [2/2] . . . . .	254
7.49.3.5 tic() . . . . .	254
7.49.3.6 tics() . . . . .	254
7.49.3.7 toc() . . . . .	255
7.49.4 Member Data Documentation . . . . .	255

7.49.4.1	<code>end_</code>	255
7.49.4.2	<code>start_</code>	255
7.50	<code>qpp::exception::TypeMismatch</code> Class Reference	256
7.50.1	Detailed Description	257
7.50.2	Member Function Documentation	257
7.50.2.1	<code>type_description()</code>	257
7.51	<code>qpp::exception::UndefinedType</code> Class Reference	257
7.51.1	Detailed Description	259
7.51.2	Member Function Documentation	259
7.51.2.1	<code>type_description()</code>	259
7.52	<code>qpp::exception::Unknown</code> Class Reference	259
7.52.1	Detailed Description	260
7.52.2	Member Function Documentation	260
7.52.2.1	<code>type_description()</code>	260
7.53	<code>qpp::exception::ZeroSize</code> Class Reference	261
7.53.1	Detailed Description	262
7.53.2	Member Function Documentation	262
7.53.2.1	<code>type_description()</code>	262
<b>8</b>	<b>File Documentation</b>	<b>263</b>
8.1	<code>classes/codes.h</code> File Reference	263
8.1.1	Detailed Description	263
8.2	<code>classes/exception.h</code> File Reference	264
8.2.1	Detailed Description	265
8.3	<code>classes/gates.h</code> File Reference	266
8.3.1	Detailed Description	266
8.4	<code>classes/ideisplay.h</code> File Reference	266
8.4.1	Detailed Description	267
8.5	<code>classes/init.h</code> File Reference	267
8.5.1	Detailed Description	267
8.6	<code>classes/random_devices.h</code> File Reference	268



8.6.1	Detailed Description	268
8.7	classes/reversible.h File Reference	268
8.7.1	Detailed Description	269
8.8	classes/states.h File Reference	269
8.8.1	Detailed Description	270
8.9	classes/timer.h File Reference	270
8.9.1	Detailed Description	270
8.10	constants.h File Reference	271
8.10.1	Detailed Description	272
8.11	entanglement.h File Reference	272
8.11.1	Detailed Description	273
8.12	entropies.h File Reference	273
8.12.1	Detailed Description	274
8.13	experimental/experimental.h File Reference	275
8.13.1	Detailed Description	275
8.14	functions.h File Reference	275
8.14.1	Detailed Description	279
8.15	input_output.h File Reference	280
8.15.1	Detailed Description	281
8.16	instruments.h File Reference	281
8.16.1	Detailed Description	282
8.17	internal/classes/iomanip.h File Reference	282
8.17.1	Detailed Description	283
8.18	internal/classes/singleton.h File Reference	283
8.18.1	Detailed Description	284
8.19	internal/util.h File Reference	284
8.19.1	Detailed Description	285
8.20	MATLAB/matlab.h File Reference	286
8.20.1	Detailed Description	286
8.21	number_theory.h File Reference	286

8.21.1 Detailed Description . . . . .	288
8.22 operations.h File Reference . . . . .	288
8.22.1 Detailed Description . . . . .	290
8.23 qpp.h File Reference . . . . .	290
8.23.1 Detailed Description . . . . .	291
8.23.2 Macro Definition Documentation . . . . .	291
8.23.2.1 QPP_UNUSED_ . . . . .	291
8.24 random.h File Reference . . . . .	291
8.24.1 Detailed Description . . . . .	293
8.25 statistics.h File Reference . . . . .	293
8.25.1 Detailed Description . . . . .	294
8.26 traits.h File Reference . . . . .	294
8.26.1 Detailed Description . . . . .	295
8.27 types.h File Reference . . . . .	295
8.27.1 Detailed Description . . . . .	296
8.28 /Users/vlad/qpp/README.md File Reference . . . . .	296
<b>Index</b>	<b>297</b>

# Chapter 1

## Quantum++

Version 1.0 - 3 July 2018

**Build status:**

**Chat (questions/issues)**

### About

Quantum++ is a modern C++11 general purpose quantum computing library, composed solely of template header files. Quantum++ is written in standard C++11 and has very low external dependencies, using only the [Eigen 3](#) linear algebra header-only template library and, if available, the [OpenMP](#) multi-processing library.

Quantum++ is not restricted to qubit systems or specific quantum information processing tasks, being capable of simulating arbitrary quantum processes. The main design factors taken in consideration were the ease of use, high portability, and high performance. The library's simulation capabilities are only restricted by the amount of available physical memory. On a typical machine (Intel i5 8Gb RAM) Quantum++ can successfully simulate the evolution of 25 qubits in a pure state or of 12 qubits in a mixed state reasonably fast.

To report any bugs or ask for additional features/enhancements, please [submit an issue](#) with an appropriate label.

If you are interesting in contributing to this project, feel free to contact me. Alternatively, create a custom branch, add your contribution, then finally create a pull request. If I accept the pull request, I will merge your custom branch with the latest development branch. The latter will eventually be merged into a future release version. To contribute, you need to have a solid knowledge of C++ (preferably C++11), including templates and the standard library, a basic knowledge of quantum computing and linear algebra, and working experience with [Eigen 3](#).

For additional [Eigen 3](#) documentation see <http://eigen.tuxfamily.org/dox/>. For a simple [Eigen 3](#) quick ASCII reference see <http://eigen.tuxfamily.org/dox/AsciiQuickReference.txt>.

Copyright (c) 2013 - 2018 Vlad Gheorghiu, vgheorgh AT gmail DOT com.

### License

[Quantum++](#) is distributed under the MIT license. Please see the [LICENSE](#) file for more details.

### Installation instructions and further documentation

Please see the installation guide <https://github.com/vsoftco/qpp/blob/master/INSTALL.md> "INSTALL.md" and the comprehensive [Wiki](#) for further documentation and detailed examples.

The official API documentation is available in PDF and HTML formats in the [doc](#) folder.



## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">qpp</a>	Quantum++ main namespace . . . . .	13
<a href="#">qpp::exception</a>	Quantum++ exception hierarchy namespace . . . . .	113
<a href="#">qpp::experimental</a>	Experimental/test functions/classes, do not use or modify . . . . .	115
<a href="#">qpp::internal</a>	Internal utility functions, do not use them directly or modify them . . . . .	115
<a href="#">qpp::literals</a>	. . . . .	121



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

qpp::internal::Display_Impl_ . . . . .	147
qpp::internal::IOManipEigen . . . . .	181
std::exception	
qpp::exception::Exception . . . . .	161
qpp::exception::CustomException . . . . .	133
qpp::exception::DimsInvalid . . . . .	136
qpp::exception::DimsMismatchCvector . . . . .	137
qpp::exception::DimsMismatchMatrix . . . . .	139
qpp::exception::DimsMismatchRvector . . . . .	141
qpp::exception::DimsMismatchVector . . . . .	143
qpp::exception::DimsNotEqual . . . . .	145
qpp::exception::MatrixMismatchSubsys . . . . .	196
qpp::exception::MatrixNotCvector . . . . .	197
qpp::exception::MatrixNotRvector . . . . .	199
qpp::exception::MatrixNotSquare . . . . .	201
qpp::exception::MatrixNotSquareNorCvector . . . . .	203
qpp::exception::MatrixNotSquareNorRvector . . . . .	205
qpp::exception::MatrixNotSquareNorVector . . . . .	207
qpp::exception::MatrixNotVector . . . . .	209
qpp::exception::NoCodeword . . . . .	211
qpp::exception::NotBipartite . . . . .	213
qpp::exception::NotQubitCvector . . . . .	215
qpp::exception::NotQubitMatrix . . . . .	217
qpp::exception::NotQubitRvector . . . . .	219
qpp::exception::NotQubitSubsys . . . . .	221
qpp::exception::NotQubitVector . . . . .	223
qpp::exception::OutOfRange . . . . .	225
qpp::exception::PermInvalid . . . . .	227
qpp::exception::PermMismatchDims . . . . .	229
qpp::exception::SizeMismatch . . . . .	237
qpp::exception::SubsysMismatchDims . . . . .	249
qpp::exception::TypeMismatch . . . . .	256
qpp::exception::UndefinedType . . . . .	257
qpp::exception::Unknown . . . . .	259
qpp::exception::ZeroSize . . . . .	261

false_type	
qpp::is_complex< T > . . . . .	190
qpp::is_iterable< T, typename > . . . . .	192
qpp::Bit_circuit::Gate_count . . . . .	164
qpp::IDisplay . . . . .	176
qpp::Dynamic_bitset . . . . .	148
qpp::Bit_circuit . . . . .	125
qpp::internal::IOManipEigen . . . . .	181
qpp::internal::IOManipPointer< PointerType > . . . . .	183
qpp::internal::IOManipRange< InputIterator > . . . . .	187
qpp::Timer< T, CLOCK_T > . . . . .	250
is_base_of	
qpp::is_matrix_expression< Derived > . . . . .	194
qpp::make_void< Ts > . . . . .	195
qpp::internal::Singleton< T > . . . . .	235
qpp::internal::Singleton< const Codes > . . . . .	235
qpp::Codes . . . . .	130
qpp::internal::Singleton< const Gates > . . . . .	235
qpp::Gates . . . . .	166
qpp::internal::Singleton< const Init > . . . . .	235
qpp::Init . . . . .	179
qpp::internal::Singleton< const States > . . . . .	235
qpp::States . . . . .	239
qpp::internal::Singleton< RandomDevices > . . . . .	235
qpp::RandomDevices . . . . .	231
true_type	
qpp::is_complex< std::complex< T > > . . . . .	191
qpp::is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), typename T::value_type > > . . . . .	193



## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">qpp::Bit_circuit</a>	Classical reversible circuit simulator . . . . .	125
<a href="#">qpp::Codes</a>	Const Singleton class that defines quantum error correcting codes . . . . .	130
<a href="#">qpp::exception::CustomException</a>	Custom exception . . . . .	133
<a href="#">qpp::exception::DimsInvalid</a>	Invalid dimension(s) exception . . . . .	136
<a href="#">qpp::exception::DimsMismatchCvector</a>	Dimension(s) mismatch column vector size exception . . . . .	137
<a href="#">qpp::exception::DimsMismatchMatrix</a>	Dimension(s) mismatch matrix size exception . . . . .	139
<a href="#">qpp::exception::DimsMismatchRvector</a>	Dimension(s) mismatch row vector size exception . . . . .	141
<a href="#">qpp::exception::DimsMismatchVector</a>	Dimension(s) mismatch vector size exception . . . . .	143
<a href="#">qpp::exception::DimsNotEqual</a>	Dimensions not equal exception . . . . .	145
<a href="#">qpp::internal::Display_Impl_</a>		147
<a href="#">qpp::Dynamic_bitset</a>	Dynamic bitset class, allows the specification of the number of bits at runtime (unlike <code>std::bitset&lt;N&gt;</code> ) . . . . .	148
<a href="#">qpp::exception::Exception</a>	Base class for generating Quantum++ custom exceptions . . . . .	161
<a href="#">qpp::Bit_circuit::Gate_count</a>		164
<a href="#">qpp::Gates</a>	Const Singleton class that implements most commonly used gates . . . . .	166
<a href="#">qpp::IDisplay</a>	Abstract class (interface) that mandates the definition of virtual <code>std::ostream&amp; display(std::ostream&amp; os) const</code> . . . . .	176
<a href="#">qpp::Init</a>	Const Singleton class that performs additional initializations/cleanups . . . . .	179
<a href="#">qpp::internal::IOManipEigen</a>		181
<a href="#">qpp::internal::IOManipPointer&lt; PointerType &gt;</a>		183
<a href="#">qpp::internal::IOManipRange&lt; InputIterator &gt;</a>		187

<a href="#">qpp::is_complex&lt; T &gt;</a>	190
Checks whether the type is a complex type . . . . .	
<a href="#">qpp::is_complex&lt; std::complex&lt; T &gt; &gt;</a>	191
Checks whether the type is a complex number type, specialization for complex types . . . . .	
<a href="#">qpp::is_iterable&lt; T, typename &gt;</a>	192
Checks whether <i>T</i> is compatible with an STL-like iterable container . . . . .	
<a href="#">qpp::is_iterable&lt; T, to_void&lt; decltype(std::declval&lt; T &gt;().begin()), decltype(std::declval&lt; T &gt;().end()), typename T::value_type &gt; &gt;</a>	193
Checks whether <i>T</i> is compatible with an STL-like iterable container, specialization for STL-like iterable containers . . . . .	
<a href="#">qpp::is_matrix_expression&lt; Derived &gt;</a>	194
Checks whether the type is an Eigen matrix expression . . . . .	
<a href="#">qpp::make_void&lt; Ts &gt;</a>	195
Helper for <a href="#">qpp::to_void&lt;&gt;</a> alias template . . . . .	
<a href="#">qpp::exception::MatrixMismatchSubsys</a>	196
Matrix mismatch subsystems exception . . . . .	
<a href="#">qpp::exception::MatrixNotCvector</a>	197
Matrix is not a column vector exception . . . . .	
<a href="#">qpp::exception::MatrixNotRvector</a>	199
Matrix is not a row vector exception . . . . .	
<a href="#">qpp::exception::MatrixNotSquare</a>	201
Matrix is not square exception . . . . .	
<a href="#">qpp::exception::MatrixNotSquareNorCvector</a>	203
Matrix is not square nor column vector exception . . . . .	
<a href="#">qpp::exception::MatrixNotSquareNorRvector</a>	205
Matrix is not square nor row vector exception . . . . .	
<a href="#">qpp::exception::MatrixNotSquareNorVector</a>	207
Matrix is not square nor vector exception . . . . .	
<a href="#">qpp::exception::MatrixNotVector</a>	209
Matrix is not a vector exception . . . . .	
<a href="#">qpp::exception::NoCodeword</a>	211
Codeword does not exist exception . . . . .	
<a href="#">qpp::exception::NotBipartite</a>	213
Not bi-partite exception . . . . .	
<a href="#">qpp::exception::NotQubitCvector</a>	215
Column vector is not 2 x 1 exception . . . . .	
<a href="#">qpp::exception::NotQubitMatrix</a>	217
Matrix is not 2 x 2 exception . . . . .	
<a href="#">qpp::exception::NotQubitRvector</a>	219
Row vector is not 1 x 2 exception . . . . .	
<a href="#">qpp::exception::NotQubitSubsys</a>	221
Subsystems are not qubits exception . . . . .	
<a href="#">qpp::exception::NotQubitVector</a>	223
Vector is not 2 x 1 nor 1 x 2 exception . . . . .	
<a href="#">qpp::exception::OutOfRange</a>	225
Parameter out of range exception . . . . .	
<a href="#">qpp::exception::PermInvalid</a>	227
Invalid permutation exception . . . . .	
<a href="#">qpp::exception::PermMismatchDims</a>	229
Permutation mismatch dimensions exception . . . . .	
<a href="#">qpp::RandomDevices</a>	231
Singleton class that manages the source of randomness in the library . . . . .	
<a href="#">qpp::internal::Singleton&lt; T &gt;</a>	235
<a href="#">Singleton</a> policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern) . . . . .	
<a href="#">qpp::exception::SizeMismatch</a>	237
Size mismatch exception . . . . .	

<a href="#">qpp::States</a>	
Const Singleton class that implements most commonly used states . . . . .	239
<a href="#">qpp::exception::SubsysMismatchDims</a>	
Subsystems mismatch dimensions exception . . . . .	249
<a href="#">qpp::Timer&lt; T, CLOCK_T &gt;</a>	
Chronometer . . . . .	250
<a href="#">qpp::exception::TypeMismatch</a>	
Type mismatch exception . . . . .	256
<a href="#">qpp::exception::UndefinedType</a>	
Not defined for this type exception . . . . .	257
<a href="#">qpp::exception::Unknown</a>	
Unknown exception . . . . .	259
<a href="#">qpp::exception::ZeroSize</a>	
Object has zero size exception . . . . .	261



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">constants.h</a>	
Constants . . . . .	271
<a href="#">entanglement.h</a>	
Entanglement functions . . . . .	272
<a href="#">entropies.h</a>	
Entropy functions . . . . .	273
<a href="#">functions.h</a>	
Generic quantum computing functions . . . . .	275
<a href="#">input_output.h</a>	
Input/output functions . . . . .	280
<a href="#">instruments.h</a>	
Measurement functions . . . . .	281
<a href="#">number_theory.h</a>	
Number theory functions . . . . .	286
<a href="#">operations.h</a>	
Quantum operation functions . . . . .	288
<a href="#">qpp.h</a>	
Quantum++ main header file, includes all other necessary headers . . . . .	290
<a href="#">random.h</a>	
Randomness-related functions . . . . .	291
<a href="#">statistics.h</a>	
Statistics functions . . . . .	293
<a href="#">traits.h</a>	
Type traits . . . . .	294
<a href="#">types.h</a>	
Type aliases . . . . .	295
classes/ <a href="#">codes.h</a>	
Quantum error correcting codes . . . . .	263
classes/ <a href="#">exception.h</a>	
Exceptions . . . . .	264
classes/ <a href="#">gates.h</a>	
Quantum gates . . . . .	266
classes/ <a href="#">display.h</a>	
Display interface via the non-virtual interface (NVI) . . . . .	266
classes/ <a href="#">init.h</a>	
Initialization . . . . .	267

classes/ <a href="#">random_devices.h</a>	
Random devices . . . . .	268
classes/ <a href="#">reversible.h</a>	
Support for classical reversible circuits . . . . .	268
classes/ <a href="#">states.h</a>	
Quantum states . . . . .	269
classes/ <a href="#">timer.h</a>	
Timing . . . . .	270
experimental/ <a href="#">experimental.h</a>	
Experimental/test functions/classes . . . . .	275
internal/ <a href="#">util.h</a>	
Internal utility functions . . . . .	284
internal/classes/ <a href="#">iomanip.h</a>	
Input/output manipulators . . . . .	282
internal/classes/ <a href="#">singleton.h</a>	
Singleton pattern via CRTP . . . . .	283
MATLAB/ <a href="#">matlab.h</a>	
Input/output interfacing with MATLAB . . . . .	286

## Chapter 6

# Namespace Documentation

### 6.1 qpp Namespace Reference

Quantum++ main namespace.

#### Namespaces

- [exception](#)  
*Quantum++ exception hierarchy namespace.*
- [experimental](#)  
*Experimental/test functions/classes, do not use or modify.*
- [internal](#)  
*Internal utility functions, do not use them directly or modify them.*
- [literals](#)

#### Classes

- class [Bit\\_circuit](#)  
*Classical reversible circuit simulator.*
- class [Codes](#)  
*const Singleton class that defines quantum error correcting codes*
- class [Dynamic\\_bitset](#)  
*Dynamic bitset class, allows the specification of the number of bits at runtime (unlike `std::bitset<N>`)*
- class [Gates](#)  
*const Singleton class that implements most commonly used gates*
- class [IDisplay](#)  
*Abstract class (interface) that mandates the definition of virtual `std::ostream& display(std::ostream& os) const`.*
- class [Init](#)  
*const Singleton class that performs additional initializations/cleanups*
- struct [is\\_complex](#)  
*Checks whether the type is a complex type.*
- struct [is\\_complex< std::complex< T > >](#)  
*Checks whether the type is a complex number type, specialization for complex types.*
- struct [is\\_iterable](#)

*Checks whether T is compatible with an STL-like iterable container.*

- struct `is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), typename T::value_type > >`

*Checks whether T is compatible with an STL-like iterable container, specialization for STL-like iterable containers.*

- struct `is_matrix_expression`

*Checks whether the type is an Eigen matrix expression.*

- struct `make_void`

*Helper for `qpp::to_void<>` alias template.*

- class `RandomDevices`

*Singleton class that manages the source of randomness in the library.*

- class `States`

*const Singleton class that implements most commonly used states*

- class `Timer`

*Chronometer.*

## Typedefs

- template<typename... Ts>

using `to_void` = typename `make_void< Ts... >::type`

*Alias template that implements the proposal for void\_t.*

- using `idx` = `std::size_t`

*Non-negative integer index.*

- using `bigint` = `long long int`

*Big integer.*

- using `cplx` = `std::complex< double >`

*Complex number in double precision.*

- using `ket` = `Eigen::VectorXcd`

*Complex (double precision) dynamic Eigen column vector.*

- using `bra` = `Eigen::RowVectorXcd`

*Complex (double precision) dynamic Eigen row vector.*

- using `cmat` = `Eigen::MatrixXcd`

*Complex (double precision) dynamic Eigen matrix.*

- using `dmat` = `Eigen::MatrixXd`

*Real (double precision) dynamic Eigen matrix.*

- template<typename Scalar >

using `dyn_mat` = `Eigen::Matrix< Scalar, Eigen::Dynamic, Eigen::Dynamic >`

*Dynamic Eigen matrix over the field specified by Scalar.*

- template<typename Scalar >

using `dyn_col_vect` = `Eigen::Matrix< Scalar, Eigen::Dynamic, 1 >`

*Dynamic Eigen column vector over the field specified by Scalar.*

- template<typename Scalar >

using `dyn_row_vect` = `Eigen::Matrix< Scalar, 1, Eigen::Dynamic >`

*Dynamic Eigen row vector over the field specified by Scalar.*



## Functions

- constexpr `cplx operator"" _i` (long double x) noexcept  
*User-defined literal for complex  $i = \sqrt{-1}$  (real overload)*
- `cplx omega` (idx D)  
*D-th root of unity.*
- template<typename Derived >  
`dyn_col_vect< double > schmidtcoeffs` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)  
*Schmidt coefficients of the bi-partite pure state A.*
- template<typename Derived >  
`dyn_col_vect< double > schmidtcoeffs` (const Eigen::MatrixBase< Derived > &A, idx d=2)  
*Schmidt coefficients of the bi-partite pure state A.*
- template<typename Derived >  
`cmat schmidtA` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)  
*Schmidt basis on Alice side.*
- template<typename Derived >  
`cmat schmidtA` (const Eigen::MatrixBase< Derived > &A, idx d=2)  
*Schmidt basis on Alice side.*
- template<typename Derived >  
`cmat schmidtB` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)  
*Schmidt basis on Bob side.*
- template<typename Derived >  
`cmat schmidtB` (const Eigen::MatrixBase< Derived > &A, idx d=2)  
*Schmidt basis on Bob side.*
- template<typename Derived >  
`std::vector< double > schmidtprobs` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)  
*Schmidt probabilities of the bi-partite pure state A.*
- template<typename Derived >  
`std::vector< double > schmidtprobs` (const Eigen::MatrixBase< Derived > &A, idx d=2)  
*Schmidt probabilities of the bi-partite pure state A.*
- template<typename Derived >  
`double entanglement` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)  
*Entanglement of the bi-partite pure state A.*
- template<typename Derived >  
`double entanglement` (const Eigen::MatrixBase< Derived > &A, idx d=2)  
*Entanglement of the bi-partite pure state A.*
- template<typename Derived >  
`double gconcurrence` (const Eigen::MatrixBase< Derived > &A)  
*G-concurrence of the bi-partite pure state A.*
- template<typename Derived >  
`double negativity` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)  
*Negativity of the bi-partite mixed state A.*
- template<typename Derived >  
`double negativity` (const Eigen::MatrixBase< Derived > &A, idx d=2)  
*Negativity of the bi-partite mixed state A.*
- template<typename Derived >  
`double lognegativity` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)  
*Logarithmic negativity of the bi-partite mixed state A.*
- template<typename Derived >  
`double lognegativity` (const Eigen::MatrixBase< Derived > &A, idx d=2)  
*Logarithmic negativity of the bi-partite mixed state A.*

- `template<typename Derived >`  
`double concurrence (const Eigen::MatrixBase< Derived > &A)`  
*Wootters concurrence of the bi-partite qubit mixed state A.*
- `template<typename Derived >`  
`double entropy (const Eigen::MatrixBase< Derived > &A)`  
*von-Neumann entropy of the density matrix A*
- `double entropy (const std::vector< double > &prob)`  
*Shannon entropy of the probability distribution prob.*
- `template<typename Derived >`  
`double renyi (const Eigen::MatrixBase< Derived > &A, double alpha)`  
*Renyi-  $\alpha$  entropy of the density matrix A, for  $\alpha \geq 0$ .*
- `double renyi (const std::vector< double > &prob, double alpha)`  
*Renyi-  $\alpha$  entropy of the probability distribution prob, for  $\alpha \geq 0$ .*
- `template<typename Derived >`  
`double tsallis (const Eigen::MatrixBase< Derived > &A, double q)`  
*Tsallis-  $q$  entropy of the density matrix A, for  $q \geq 0$ .*
- `double tsallis (const std::vector< double > &prob, double q)`  
*Tsallis-  $q$  entropy of the probability distribution prob, for  $q \geq 0$ .*
- `template<typename Derived >`  
`double qmutualinfo (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsA, const std::vector< idx > &subsB, const std::vector< idx > &dims)`  
*Quantum mutual information between 2 subsystems of a composite system.*
- `template<typename Derived >`  
`double qmutualinfo (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsA, const std::vector< idx > &subsB, idx d=2)`  
*Quantum mutual information between 2 subsystems of a composite system.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > transpose (const Eigen::MatrixBase< Derived > &A)`  
*Transpose.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > conjugate (const Eigen::MatrixBase< Derived > &A)`  
*Complex conjugate.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > adjoint (const Eigen::MatrixBase< Derived > &A)`  
*Adjoint.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > inverse (const Eigen::MatrixBase< Derived > &A)`  
*Inverse.*
- `template<typename Derived >`  
`Derived::Scalar trace (const Eigen::MatrixBase< Derived > &A)`  
*Trace.*
- `template<typename Derived >`  
`Derived::Scalar det (const Eigen::MatrixBase< Derived > &A)`  
*Determinant.*
- `template<typename Derived >`  
`Derived::Scalar logdet (const Eigen::MatrixBase< Derived > &A)`  
*Logarithm of the determinant.*
- `template<typename Derived >`  
`Derived::Scalar sum (const Eigen::MatrixBase< Derived > &A)`  
*Element-wise sum of A.*
- `template<typename Derived >`  
`Derived::Scalar prod (const Eigen::MatrixBase< Derived > &A)`  
*Element-wise product of A.*

- `template<typename Derived >`  
`double norm (const Eigen::MatrixBase< Derived > &A)`  
*Frobenius norm.*
- `template<typename Derived >`  
`std::pair< dyn_col_vect< cplx >, cmat > eig (const Eigen::MatrixBase< Derived > &A)`  
*Full eigen decomposition.*
- `template<typename Derived >`  
`dyn_col_vect< cplx > evals (const Eigen::MatrixBase< Derived > &A)`  
*Eigenvalues.*
- `template<typename Derived >`  
`cmat evecs (const Eigen::MatrixBase< Derived > &A)`  
*Eigenvectors.*
- `template<typename Derived >`  
`std::pair< dyn_col_vect< double >, cmat > heig (const Eigen::MatrixBase< Derived > &A)`  
*Full eigen decomposition of Hermitian expression.*
- `template<typename Derived >`  
`dyn_col_vect< double > hevals (const Eigen::MatrixBase< Derived > &A)`  
*Hermitian eigenvalues.*
- `template<typename Derived >`  
`cmat hevecs (const Eigen::MatrixBase< Derived > &A)`  
*Hermitian eigenvectors.*
- `template<typename Derived >`  
`std::tuple< cmat, dyn_col_vect< double >, cmat > svd (const Eigen::MatrixBase< Derived > &A)`  
*Full singular value decomposition.*
- `template<typename Derived >`  
`dyn_col_vect< double > svals (const Eigen::MatrixBase< Derived > &A)`  
*Singular values.*
- `template<typename Derived >`  
`cmat svdU (const Eigen::MatrixBase< Derived > &A)`  
*Left singular vectors.*
- `template<typename Derived >`  
`cmat svdV (const Eigen::MatrixBase< Derived > &A)`  
*Right singular vectors.*
- `template<typename Derived >`  
`cmat funm (const Eigen::MatrixBase< Derived > &A, cplx(*f)(const cplx &))`  
*Functional calculus  $f(A)$*
- `template<typename Derived >`  
`cmat sqrtm (const Eigen::MatrixBase< Derived > &A)`  
*Matrix square root.*
- `template<typename Derived >`  
`cmat absm (const Eigen::MatrixBase< Derived > &A)`  
*Matrix absolute value.*
- `template<typename Derived >`  
`cmat expm (const Eigen::MatrixBase< Derived > &A)`  
*Matrix exponential.*
- `template<typename Derived >`  
`cmat logm (const Eigen::MatrixBase< Derived > &A)`  
*Matrix logarithm.*
- `template<typename Derived >`  
`cmat sinm (const Eigen::MatrixBase< Derived > &A)`  
*Matrix sin.*
- `template<typename Derived >`  
`cmat cosm (const Eigen::MatrixBase< Derived > &A)`

*Matrix cos.*

- `template<typename Derived >`  
`cmat spectralpowm` (const Eigen::MatrixBase< Derived > &A, const `cplx` z)

*Matrix power.*

- `template<typename Derived >`  
`dyn_mat`< typename Derived::Scalar > `powm` (const Eigen::MatrixBase< Derived > &A, `idx` n)

*Fast matrix power based on the SQUARE-AND-MULTIPLY algorithm.*

- `template<typename Derived >`  
double `schatten` (const Eigen::MatrixBase< Derived > &A, double p)

*Schatten matrix norm.*

- `template<typename OutputScalar , typename Derived >`  
`dyn_mat`< OutputScalar > `cwise` (const Eigen::MatrixBase< Derived > &A, OutputScalar(\*f)(const type-name Derived::Scalar &))

*Functor.*

- `template<typename T >`  
`dyn_mat`< typename T::Scalar > `kron` (const T &head)

*Kronecker product.*

- `template<typename T , typename... Args>`  
`dyn_mat`< typename T::Scalar > `kron` (const T &head, const Args &... tail)

*Kronecker product.*

- `template<typename Derived >`  
`dyn_mat`< typename Derived::Scalar > `kron` (const std::vector< Derived > &As)

*Kronecker product.*

- `template<typename Derived >`  
`dyn_mat`< typename Derived::Scalar > `kron` (const std::initializer\_list< Derived > &As)

*Kronecker product.*

- `template<typename Derived >`  
`dyn_mat`< typename Derived::Scalar > `kronpow` (const Eigen::MatrixBase< Derived > &A, `idx` n)

*Kronecker power.*

- `template<typename T >`  
`dyn_mat`< typename T::Scalar > `dirsum` (const T &head)

*Direct sum.*

- `template<typename T , typename... Args>`  
`dyn_mat`< typename T::Scalar > `dirsum` (const T &head, const Args &... tail)

*Direct sum.*

- `template<typename Derived >`  
`dyn_mat`< typename Derived::Scalar > `dirsum` (const std::vector< Derived > &As)

*Direct sum.*

- `template<typename Derived >`  
`dyn_mat`< typename Derived::Scalar > `dirsum` (const std::initializer\_list< Derived > &As)

*Direct sum.*

- `template<typename Derived >`  
`dyn_mat`< typename Derived::Scalar > `dirsumpow` (const Eigen::MatrixBase< Derived > &A, `idx` n)

*Direct sum power.*

- `template<typename Derived >`  
`dyn_mat`< typename Derived::Scalar > `reshape` (const Eigen::MatrixBase< Derived > &A, `idx` rows, `idx` cols)

*Reshape.*

- `template<typename Derived1 , typename Derived2 >`  
`dyn_mat`< typename Derived1::Scalar > `comm` (const Eigen::MatrixBase< Derived1 > &A, const Eigen::↔ MatrixBase< Derived2 > &B)

*Commutator.*

- `template<typename Derived1 , typename Derived2 >`  
`dyn_mat< typename Derived1::Scalar > anticomm` (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)  
*Anti-commutator.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > prj` (const Eigen::MatrixBase< Derived > &A)  
*Projector.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > grams` (const std::vector< Derived > &As)  
*Gram-Schmidt orthogonalization.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > grams` (const std::initializer\_list< Derived > &As)  
*Gram-Schmidt orthogonalization.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > grams` (const Eigen::MatrixBase< Derived > &A)  
*Gram-Schmidt orthogonalization.*
- `std::vector< idx > n2multiidx` (idx n, const std::vector< idx > &dims)  
*Non-negative integer index to multi-index.*
- `idx multiidx2n` (const std::vector< idx > &midx, const std::vector< idx > &dims)  
*Multi-index to non-negative integer index.*
- `ket mket` (const std::vector< idx > &mask, const std::vector< idx > &dims)  
*Multi-partite qudit ket.*
- `ket mket` (const std::vector< idx > &mask, idx d=2)  
*Multi-partite qudit ket.*
- `cmat mprj` (const std::vector< idx > &mask, const std::vector< idx > &dims)  
*Projector onto multi-partite qudit ket.*
- `cmat mprj` (const std::vector< idx > &mask, idx d=2)  
*Projector onto multi-partite qudit ket.*
- `template<typename InputIterator >`  
`std::vector< double > abssq` (InputIterator first, InputIterator last)  
*Computes the absolute values squared of an STL-like range of complex numbers.*
- `template<typename Container >`  
`std::vector< double > abssq` (const Container &c, typename std::enable\_if< is\_iterable< Container >::value >::type \*==nullptr)  
*Computes the absolute values squared of an STL-like container.*
- `template<typename Derived >`  
`std::vector< double > abssq` (const Eigen::MatrixBase< Derived > &A)  
*Computes the absolute values squared of an Eigen expression.*
- `template<typename InputIterator >`  
`std::iterator_traits< InputIterator >::value_type sum` (InputIterator first, InputIterator last)  
*Element-wise sum of an STL-like range.*
- `template<typename Container >`  
`Container::value_type sum` (const Container &c, typename std::enable\_if< is\_iterable< Container >::value >::type \*==nullptr)  
*Element-wise sum of the elements of an STL-like container.*
- `template<typename InputIterator >`  
`std::iterator_traits< InputIterator >::value_type prod` (InputIterator first, InputIterator last)  
*Element-wise product of an STL-like range.*
- `template<typename Container >`  
`Container::value_type prod` (const Container &c, typename std::enable\_if< is\_iterable< Container >::value >::type \*==nullptr)  
*Element-wise product of the elements of an STL-like container.*

- `template<typename Derived >`  
`dyn_col_vect< typename Derived::Scalar > rho2pure` (const Eigen::MatrixBase< Derived > &A)  
*Finds the pure state representation of a matrix proportional to a projector onto a pure state.*
- `template<typename T >`  
`std::vector< T > complement` (std::vector< T > subsys, `idx` N)  
*Constructs the complement of a subsystem vector.*
- `template<typename Derived >`  
`std::vector< double > rho2bloch` (const Eigen::MatrixBase< Derived > &A)  
*Computes the 3-dimensional real Bloch vector corresponding to the qubit density matrix A.*
- `cmat bloch2rho` (const std::vector< double > &r)  
*Computes the density matrix corresponding to the 3-dimensional real Bloch vector r.*
- `template<typename Derived >`  
`internal::IOManipEigen disp` (const Eigen::MatrixBase< Derived > &A, double `chop`=qpp::chop)  
*Eigen expression ostream manipulator.*
- `internal::IOManipEigen disp` (cplx z, double `chop`=qpp::chop)  
*Complex number ostream manipulator.*
- `template<typename InputIterator >`  
`internal::IOManipRange< InputIterator > disp` (InputIterator first, InputIterator last, const std::string &separator, const std::string &start="[", const std::string &end="]")  
*Range ostream manipulator.*
- `template<typename Container >`  
`internal::IOManipRange< typename Container::const_iterator > disp` (const Container &c, const std::string &separator, const std::string &start="[", const std::string &end="]", typename std::enable\_if< `is_iterable`< Container >::value >::type \*==nullptr)  
*Standard container ostream manipulator. The container must support std::begin(), std::end() and forward iteration.*
- `template<typename PointerType >`  
`internal::IOManipPointer< PointerType > disp` (const PointerType \*p, `idx` N, const std::string &separator, const std::string &start="[", const std::string &end="]")  
*C-style pointer ostream manipulator.*
- `template<typename Derived >`  
`void save` (const Eigen::MatrixBase< Derived > &A, const std::string &fname)  
*Saves Eigen expression to a binary file (internal format) in double precision.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > load` (const std::string &fname)  
*Loads Eigen matrix from a binary file (internal format) in double precision.*
- `template<typename Derived >`  
`dyn_col_vect< typename Derived::Scalar > ip` (const Eigen::MatrixBase< Derived > &phi, const Eigen::MatrixBase< Derived > &psi, const std::vector< `idx` > &subsys, const std::vector< `idx` > &dims)  
*Generalized inner product.*
- `template<typename Derived >`  
`dyn_col_vect< typename Derived::Scalar > ip` (const Eigen::MatrixBase< Derived > &phi, const Eigen::MatrixBase< Derived > &psi, const std::vector< `idx` > &subsys, `idx` d=2)  
*Generalized inner product.*
- `template<typename Derived >`  
`std::tuple< idx, std::vector< double >, std::vector< cmat > > measure` (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks)  
*Measures the state A using the set of Kraus operators Ks.*
- `template<typename Derived >`  
`std::tuple< idx, std::vector< double >, std::vector< cmat > > measure` (const Eigen::MatrixBase< Derived > &A, const std::initializer\_list< cmat > &Ks)  
*Measures the state A using the set of Kraus operators Ks.*
- `template<typename Derived >`  
`std::tuple< idx, std::vector< double >, std::vector< cmat > > measure` (const Eigen::MatrixBase< Derived > &A, const cmat &U)

*Measures the state A in the orthonormal basis specified by the unitary matrix U.*

- `template<typename Derived >`  
`std::tuple< idx, std::vector< double >, std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks, const std::vector< idx > &subsys, const std::vector< idx > &dims)`

*Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*

- `template<typename Derived >`  
`std::tuple< idx, std::vector< double >, std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const std::initializer_list< cmat > &Ks, const std::vector< idx > &subsys, const std::vector< idx > &dims)`

*Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*

- `template<typename Derived >`  
`std::tuple< idx, std::vector< double >, std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks, const std::vector< idx > &subsys, idx d=2)`

*Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*

- `template<typename Derived >`  
`std::tuple< idx, std::vector< double >, std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const std::initializer_list< cmat > &Ks, const std::vector< idx > &subsys, idx d=2)`

*Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*

- `template<typename Derived >`  
`std::tuple< idx, std::vector< double >, std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const cmat &V, const std::vector< idx > &subsys, const std::vector< idx > &dims)`

*Measures the part subsys of the multi-partite state vector or density matrix A in the orthonormal basis or rank-1 POVM specified by the matrix V.*

- `template<typename Derived >`  
`std::tuple< idx, std::vector< double >, std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const cmat &V, const std::vector< idx > &subsys, idx d=2)`

*Measures the part subsys of the multi-partite state vector or density matrix A in the orthonormal basis or rank-1 POVM specified by the matrix V.*

- `template<typename Derived >`  
`std::tuple< std::vector< idx >, double, cmat > measure_seq (const Eigen::MatrixBase< Derived > &A, std::vector< idx > subsys, std::vector< idx > dims)`

*Sequentially measures the part subsys of the multi-partite state vector or density matrix A in the computational basis.*

- `template<typename Derived >`  
`std::tuple< std::vector< idx >, double, cmat > measure_seq (const Eigen::MatrixBase< Derived > &A, std::vector< idx > subsys, idx d=2)`

*Sequentially measures the part subsys of the multi-partite state vector or density matrix A in the computational basis.*

- `template<typename Derived >`  
`std::enable_if< std::is_same< typename Derived::Scalar, cplx >::value, dyn_mat< cplx > >::type loadMATLAB (const std::string &mat_file, const std::string &var_name)`

*Loads a complex Eigen dynamic matrix from a MATLAB .mat file,.*

- `template<typename Derived >`  
`std::enable_if< !std::is_same< typename Derived::Scalar, cplx >::value, dyn_mat< typename Derived::Scalar > >::type loadMATLAB (const std::string &mat_file, const std::string &var_name)`

*Loads a non-complex Eigen dynamic matrix from a MATLAB .mat file,.*

- `template<typename Derived >`  
`std::enable_if< std::is_same< typename Derived::Scalar, cplx >::value >::type saveMATLAB (const Eigen::MatrixBase< Derived > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)`

*Saves a complex Eigen dynamic matrix to a MATLAB .mat file,.*

- `template<typename Derived >`  
`std::enable_if< !std::is_same< typename Derived::Scalar, cplx >::value >::type saveMATLAB (const Eigen::MatrixBase< Derived > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)`

*Saves a non-complex Eigen dynamic matrix to a MATLAB .mat file,.*

- `std::vector< int > x2confrac (double x, idx N, idx cut=1e5)`



- Simple continued fraction expansion.*

  - double `contrac2x` (const std::vector< int > &cf, `idx` N=`idx`(-1))
- Real representation of a simple continued fraction.*

  - `bigint gcd` (`bigint` a, `bigint` b)
- Greatest common divisor of two integers.*

  - `bigint gcd` (const std::vector< `bigint` > &as)
- Greatest common divisor of a list of integers.*

  - `bigint lcm` (`bigint` a, `bigint` b)
- Least common multiple of two integers.*

  - `bigint lcm` (const std::vector< `bigint` > &as)
- Least common multiple of a list of integers.*

  - std::vector< `idx` > `invperm` (const std::vector< `idx` > &perm)
- Inverse permutation.*

  - std::vector< `idx` > `compperm` (const std::vector< `idx` > &perm, const std::vector< `idx` > &sigma)
- Compose permutations.*

  - std::vector< `bigint` > `factors` (`bigint` a)
- Prime factor decomposition.*

  - `bigint modmul` (`bigint` a, `bigint` b, `bigint` p)
- Modular multiplication without overflow.*

  - `bigint modpow` (`bigint` a, `bigint` n, `bigint` p)
- Fast integer power modulo p based on the SQUARE-AND-MULTIPLY algorithm.*

  - std::tuple< `bigint`, `bigint`, `bigint` > `egcd` (`bigint` a, `bigint` b)
- Extended greatest common divisor of two integers.*

  - `bigint modinv` (`bigint` a, `bigint` p)
- Modular inverse of a mod p.*

  - bool `isprime` (`bigint` p, `idx` k=80)
- Primality test based on the Miller-Rabin's algorithm.*

  - `bigint randprime` (`bigint` a, `bigint` b, `idx` N=1000)
- Generates a random big prime uniformly distributed in the interval [a, b].*

  - template<typename Derived1 , typename Derived2 >  
`dyn_mat`< typename Derived1::Scalar > `applyCTRL` (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< `idx` > &ctrl, const std::vector< `idx` > &subsys, const std::vector< `idx` > &dims)
- Applies the controlled-gate A to the part subsys of the multi-partite state vector or density matrix state.*

  - template<typename Derived1 , typename Derived2 >  
`dyn_mat`< typename Derived1::Scalar > `applyCTRL` (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< `idx` > &ctrl, const std::vector< `idx` > &subsys, `idx` d=2)
- Applies the controlled-gate A to the part subsys of the multi-partite state vector or density matrix state.*

  - template<typename Derived1 , typename Derived2 >  
`dyn_mat`< typename Derived1::Scalar > `apply` (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< `idx` > &subsys, const std::vector< `idx` > &dims)
- Applies the gate A to the part subsys of the multi-partite state vector or density matrix state.*

  - template<typename Derived1 , typename Derived2 >  
`dyn_mat`< typename Derived1::Scalar > `apply` (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< `idx` > &subsys, `idx` d=2)
- Applies the gate A to the part subsys of the multi-partite state vector or density matrix state.*

  - template<typename Derived >  
`cmat apply` (const Eigen::MatrixBase< Derived > &A, const std::vector< `cmat` > &Ks)
- Applies the channel specified by the set of Kraus operators Ks to the density matrix A.*

  - template<typename Derived >  
`cmat apply` (const Eigen::MatrixBase< Derived > &A, const std::vector< `cmat` > &Ks, const std::vector< `idx` > &subsys, const std::vector< `idx` > &dims)



*Applies the channel specified by the set of Kraus operators Ks to the part subsys of the multi-partite density matrix A.*

- `template<typename Derived >`  
`cmat apply` (const Eigen::MatrixBase< Derived > &A, const std::vector< `cmat` > &Ks, const std::vector< `idx` > &subsys, `idx` d=2)

*Applies the channel specified by the set of Kraus operators Ks to the part subsys of the multi-partite density matrix A.*

- `cmat kraus2super` (const std::vector< `cmat` > &Ks)

*Superoperator matrix.*

- `cmat kraus2choi` (const std::vector< `cmat` > &Ks)

*Choi matrix.*

- `std::vector< cmat > choi2kraus` (const `cmat` &A)

*Orthogonal Kraus operators from Choi matrix.*

- `cmat choi2super` (const `cmat` &A)

*Converts Choi matrix to superoperator matrix.*

- `cmat super2choi` (const `cmat` &A)

*Converts superoperator matrix to Choi matrix.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > ptrace1` (const Eigen::MatrixBase< Derived > &A, const std::vector< `idx` > &dims)

*Partial trace.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > ptrace1` (const Eigen::MatrixBase< Derived > &A, `idx` d=2)

*Partial trace.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > ptrace2` (const Eigen::MatrixBase< Derived > &A, const std::vector< `idx` > &dims)

*Partial trace.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > ptrace2` (const Eigen::MatrixBase< Derived > &A, `idx` d=2)

*Partial trace.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > ptrace` (const Eigen::MatrixBase< Derived > &A, const std::vector< `idx` > &subsys, const std::vector< `idx` > &dims)

*Partial trace.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > ptrace` (const Eigen::MatrixBase< Derived > &A, const std::vector< `idx` > &subsys, `idx` d=2)

*Partial trace.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > ptranspose` (const Eigen::MatrixBase< Derived > &A, const std::vector< `idx` > &subsys, const std::vector< `idx` > &dims)

*Partial transpose.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > ptranspose` (const Eigen::MatrixBase< Derived > &A, const std::vector< `idx` > &subsys, `idx` d=2)

*Partial transpose.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > syspermute` (const Eigen::MatrixBase< Derived > &A, const std::vector< `idx` > &perm, const std::vector< `idx` > &dims)

*Subsystem permutation.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > syspermute` (const Eigen::MatrixBase< Derived > &A, const std::vector< `idx` > &perm, `idx` d=2)

*Subsystem permutation.*

- `double rand (double a, double b)`  
*Generates a random real number uniformly distributed in the interval [a, b]*
- `bigint rand (bigint a, bigint b)`  
*Generates a random big integer uniformly distributed in the interval [a, b].*
- `idx randidx (idx a=std::numeric_limits< idx >::min(), idx b=std::numeric_limits< idx >::max())`  
*Generates a random index (idx) uniformly distributed in the interval [a, b].*
- `template<typename Derived >`  
`Derived rand (idx rows, idx cols, double a=0, double b=1)`  
*Generates a random matrix with entries uniformly distributed in the interval [a, b]*
- `template<>`  
`dmat rand (idx rows, idx cols, double a, double b)`  
*Generates a random real matrix with entries uniformly distributed in the interval [a, b], specialization for double matrices ([qpp::dmat](#))*
- `template<>`  
`cmat rand (idx rows, idx cols, double a, double b)`  
*Generates a random complex matrix with entries (both real and imaginary) uniformly distributed in the interval [a, b], specialization for complex matrices ([qpp::cmat](#))*
- `template<typename Derived >`  
`Derived randn (idx rows, idx cols, double mean=0, double sigma=1)`  
*Generates a random matrix with entries normally distributed in  $N(\text{mean}, \text{sigma})$*
- `template<>`  
`dmat randn (idx rows, idx cols, double mean, double sigma)`  
*Generates a random real matrix with entries normally distributed in  $N(\text{mean}, \text{sigma})$ , specialization for double matrices ([qpp::dmat](#))*
- `template<>`  
`cmat randn (idx rows, idx cols, double mean, double sigma)`  
*Generates a random complex matrix with entries (both real and imaginary) normally distributed in  $N(\text{mean}, \text{sigma})$ , specialization for complex matrices ([qpp::cmat](#))*
- `double randn (double mean=0, double sigma=1)`  
*Generates a random real number (double) normally distributed in  $N(\text{mean}, \text{sigma})$*
- `cmat randU (idx D=2)`  
*Generates a random unitary matrix.*
- `cmat randV (idx Din, idx Dout)`  
*Generates a random isometry matrix.*
- `std::vector< cmat > randkraus (idx N, idx D=2)`  
*Generates a set of random Kraus operators.*
- `cmat randH (idx D=2)`  
*Generates a random Hermitian matrix.*
- `ket randket (idx D=2)`  
*Generates a random normalized ket (pure state vector)*
- `cmat randrho (idx D=2)`  
*Generates a random density matrix.*
- `std::vector< idx > randperm (idx N)`  
*Generates a random uniformly distributed permutation.*
- `std::vector< double > randprob (idx N)`  
*Generates a random probability vector uniformly distributed over the probability simplex.*
- `std::vector< double > uniform (idx N)`  
*Uniform probability distribution vector.*
- `std::vector< double > marginalX (const dmat &probXY)`  
*Marginal distribution.*
- `std::vector< double > marginalY (const dmat &probXY)`  
*Marginal distribution.*

- `template<typename Container >`  
`double avg (const std::vector< double > &prob, const Container &X, typename std::enable_if< is_iterable< Container >::value >::type !=nullptr)`  
*Average.*
- `template<typename Container >`  
`double cov (const dmat &probXY, const Container &X, const Container &Y, typename std::enable_if< is_iterable< Container >::value >::type !=nullptr)`  
*Covariance.*
- `template<typename Container >`  
`double var (const std::vector< double > &prob, const Container &X, typename std::enable_if< is_iterable< Container >::value >::type !=nullptr)`  
*Variance.*
- `template<typename Container >`  
`double sigma (const std::vector< double > &prob, const Container &X, typename std::enable_if< is_iterable< Container >::value >::type !=nullptr)`  
*Standard deviation.*
- `template<typename Container >`  
`double cor (const dmat &probXY, const Container &X, const Container &Y, typename std::enable_if< is_iterable< Container >::value >::type !=nullptr)`  
*Correlation.*

## Variables

- `constexpr double chop = 1e-10`  
*Used in `qpp::disp()` for setting to zero numbers that have their absolute value smaller than `qpp::chop`.*
- `constexpr double eps = 1e-12`  
*Used to decide whether a number or expression in double precision is zero or not.*
- `constexpr idx maxn = 64`  
*Maximum number of allowed qubits/qudits (subsystems)*
- `constexpr double pi = 3.141592653589793238462643383279502884`  
 $\pi$
- `constexpr double ee = 2.718281828459045235360287471352662497`  
*Base of natural logarithm, e.*
- `constexpr double infy = std::numeric_limits<double>::max()`  
*Used to denote infinity in double precision.*

### 6.1.1 Detailed Description

Quantum++ main namespace.

### 6.1.2 Typedef Documentation

#### 6.1.2.1 bigint

```
using qpp::bigint = typedef long long int
```

Big integer.

#### 6.1.2.2 bra

```
using qpp::bra = typedef Eigen::RowVectorXcd
```

Complex (double precision) dynamic Eigen row vector.

#### 6.1.2.3 cmat

```
using qpp::cmat = typedef Eigen::MatrixXcd
```

Complex (double precision) dynamic Eigen matrix.

#### 6.1.2.4 cplx

```
using qpp::cplx = typedef std::complex<double>
```

Complex number in double precision.

#### 6.1.2.5 dmat

```
using qpp::dmat = typedef Eigen::MatrixXd
```

Real (double precision) dynamic Eigen matrix.

#### 6.1.2.6 dyn\_col\_vect

```
template<typename Scalar >  
using qpp::dyn_col_vect = typedef Eigen::Matrix<Scalar, Eigen::Dynamic, 1>
```

Dynamic Eigen column vector over the field specified by *Scalar*.

Example:

```
// type of colvect is Eigen::Matrix<float, Eigen::Dynamic, 1>  
dyn_col_vect<float> colvect(2);
```

### 6.1.2.7 dyn\_mat

```
template<typename Scalar >  
using qpp::dyn_mat = typedef Eigen::Matrix<Scalar, Eigen::Dynamic, Eigen::Dynamic>
```

Dynamic Eigen matrix over the field specified by *Scalar*.

Example:

```
// type of mat is Eigen::Matrix<float, Eigen::Dynamic, Eigen::Dynamic>  
dyn_mat<float> mat(2, 3);
```

### 6.1.2.8 dyn\_row\_vect

```
template<typename Scalar >  
using qpp::dyn_row_vect = typedef Eigen::Matrix<Scalar, 1, Eigen::Dynamic>
```

Dynamic Eigen row vector over the field specified by *Scalar*.

Example:

```
// type of rowvect is Eigen::Matrix<float, 1, Eigen::Dynamic>  
dyn_row_vect<float> rowvect(3);
```

### 6.1.2.9 idx

```
using qpp::idx = typedef std::size_t
```

Non-negative integer index.

### 6.1.2.10 ket

```
using qpp::ket = typedef Eigen::VectorXcd
```

Complex (double precision) dynamic Eigen column vector.

### 6.1.2.11 to\_void

```
template<typename... Ts>
using qpp::to_void = typedef typename make_void<Ts...>::type
```

Alias template that implements the proposal for void\_t.

#### See also

<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2014/n3911>

## 6.1.3 Function Documentation

### 6.1.3.1 absm()

```
template<typename Derived >
cmat qpp::absm (
    const Eigen::MatrixBase< Derived > & A )
```

Matrix absolute value.

#### Parameters

<i>A</i>	Eigen expression
----------	------------------

#### Returns

Matrix absolute value of *A*

### 6.1.3.2 abssq() [1/3]

```
template<typename InputIterator >
std::vector<double> qpp::abssq (
    InputIterator first,
    InputIterator last )
```

Computes the absolute values squared of an STL-like range of complex numbers.

#### Parameters

<i>first</i>	Iterator to the first element of the range
<i>last</i>	Iterator to the last element of the range

**Returns**

Real vector consisting of the range absolute values squared

**6.1.3.3 abssq()** [2/3]

```
template<typename Container >
std::vector<double> qpp::abssq (
    const Container & c,
    typename std::enable_if< is\_iterable< Container >::value >::type * = nullptr )
```

Computes the absolute values squared of an STL-like container.

**Parameters**

<i>c</i>	STL-like container
----------	--------------------

**Returns**

Real vector consisting of the container's absolute values squared

**6.1.3.4 abssq()** [3/3]

```
template<typename Derived >
std::vector<double> qpp::abssq (
    const Eigen::MatrixBase< Derived > & A )
```

Computes the absolute values squared of an Eigen expression.

**Parameters**

<i>A</i>	Eigen expression
----------	------------------

**Returns**

Real vector consisting of the absolute values squared

**6.1.3.5 adjoint()**

```
template<typename Derived >
dyn\_mat<typename Derived::Scalar> qpp::adjoint (
    const Eigen::MatrixBase< Derived > & A )
```

Adjoint.

## Parameters

$A$	Eigen expression
-----	------------------

## Returns

Adjoint (Hermitian conjugate) of  $A$ , as a dynamic matrix over the same scalar field as  $A$

6.1.3.6 `anticomm()`

```
template<typename Derived1 , typename Derived2 >
dyn_mat<typename Derived1::Scalar> qpp::anticomm (
    const Eigen::MatrixBase< Derived1 > & A,
    const Eigen::MatrixBase< Derived2 > & B )
```

Anti-commutator.

## See also

[qpp::comm\(\)](#)

Anti-commutator  $\{A, B\} = AB + BA$ . Both  $A$  and  $B$  must be Eigen expressions over the same scalar field.

## Parameters

$A$	Eigen expression
$B$	Eigen expression

## Returns

Anti-commutator  $AB + BA$ , as a dynamic matrix over the same scalar field as  $A$

6.1.3.7 `apply()` [1/5]

```
template<typename Derived1 , typename Derived2 >
dyn_mat<typename Derived1::Scalar> qpp::apply (
    const Eigen::MatrixBase< Derived1 > & state,
    const Eigen::MatrixBase< Derived2 > & A,
    const std::vector< idx > & subsys,
    const std::vector< idx > & dims )
```

Applies the gate  $A$  to the part *subsys* of the multi-partite state vector or density matrix *state*.

## Note

The dimension of the gate  $A$  must match the dimension of *subsys*



## Parameters

<i>state</i>	Eigen expression
<i>A</i>	Eigen expression
<i>subsys</i>	Subsystem indexes where the gate <i>A</i> is applied
<i>dims</i>	Dimensions of the multi-partite system

## Returns

Gate *A* applied to the part *subsys* of *state*

6.1.3.8 `apply()` [2/5]

```
template<typename Derived1 , typename Derived2 >
dyn_mat<typename Derived1::Scalar> qpp::apply (
    const Eigen::MatrixBase< Derived1 > & state,
    const Eigen::MatrixBase< Derived2 > & A,
    const std::vector< idx > & subsys,
    idx d = 2 )
```

Applies the gate *A* to the part *subsys* of the multi-partite state vector or density matrix *state*.

## Note

The dimension of the gate *A* must match the dimension of *subsys*

## Parameters

<i>state</i>	Eigen expression
<i>A</i>	Eigen expression
<i>subsys</i>	Subsystem indexes where the gate <i>A</i> is applied
<i>d</i>	Subsystem dimensions

## Returns

Gate *A* applied to the part *subsys* of *state*

6.1.3.9 `apply()` [3/5]

```
template<typename Derived >
cmat qpp::apply (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< cmat > & Ks )
```

Applies the channel specified by the set of Kraus operators *Ks* to the density matrix *A*.

**Parameters**

<i>A</i>	Eigen expression
<i>Ks</i>	Set of Kraus operators

**Returns**

Output density matrix after the action of the channel

**6.1.3.10 apply()** [4/5]

```
template<typename Derived >
cmat qpp::apply (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< cmat > & Ks,
    const std::vector< idx > & subsys,
    const std::vector< idx > & dims )
```

Applies the channel specified by the set of Kraus operators *Ks* to the part *subsys* of the multi-partite density matrix *A*.

**Parameters**

<i>A</i>	Eigen expression
<i>Ks</i>	Set of Kraus operators
<i>subsys</i>	Subsystem indexes where the Kraus operators <i>Ks</i> are applied
<i>dims</i>	Dimensions of the multi-partite system

**Returns**

Output density matrix after the action of the channel

**6.1.3.11 apply()** [5/5]

```
template<typename Derived >
cmat qpp::apply (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< cmat > & Ks,
    const std::vector< idx > & subsys,
    idx d = 2 )
```

Applies the channel specified by the set of Kraus operators *Ks* to the part *subsys* of the multi-partite density matrix *A*.

## Parameters

<i>A</i>	Eigen expression
<i>Ks</i>	Set of Kraus operators
<i>subsys</i>	Subsystem indexes where the Kraus operators <i>Ks</i> are applied
<i>d</i>	Subsystem dimensions

## Returns

Output density matrix after the action of the channel

6.1.3.12 `applyCTRL()` [1/2]

```
template<typename Derived1 , typename Derived2 >
dyn_mat<typename Derived1::Scalar> qpp::applyCTRL (
    const Eigen::MatrixBase< Derived1 > & state,
    const Eigen::MatrixBase< Derived2 > & A,
    const std::vector< idx > & ctrl,
    const std::vector< idx > & subsys,
    const std::vector< idx > & dims )
```

Applies the controlled-gate *A* to the part *subsys* of the multi-partite state vector or density matrix *state*.

## See also

[qpp::Gates::CTRL\(\)](#)

## Note

The dimension of the gate *A* must match the dimension of *subsys*. Also, all control subsystems in *ctrl* must have the same dimension.

## Parameters

<i>state</i>	Eigen expression
<i>A</i>	Eigen expression
<i>ctrl</i>	Control subsystem indexes
<i>subsys</i>	Subsystem indexes where the gate <i>A</i> is applied
<i>dims</i>	Dimensions of the multi-partite system

## Returns

CTRL-A gate applied to the part *subsys* of *state*

### 6.1.3.13 `applyCTRL()` [2/2]

```
template<typename Derived1 , typename Derived2 >
dyn_mat<typename Derived1::Scalar> qpp::applyCTRL (
    const Eigen::MatrixBase< Derived1 > & state,
    const Eigen::MatrixBase< Derived2 > & A,
    const std::vector< idx > & ctrl,
    const std::vector< idx > & subsys,
    idx d = 2 )
```

Applies the controlled-gate *A* to the part *subsys* of the multi-partite state vector or density matrix *state*.

See also

[qpp::Gates::CTRL\(\)](#)

Note

The dimension of the gate *A* must match the dimension of *subsys*

Parameters

<i>state</i>	Eigen expression
<i>A</i>	Eigen expression
<i>ctrl</i>	Control subsystem indexes
<i>subsys</i>	Subsystem indexes where the gate <i>A</i> is applied
<i>d</i>	Subsystem dimensions

Returns

CTRL-A gate applied to the part *subsys* of *state*

### 6.1.3.14 `avg()`

```
template<typename Container >
double qpp::avg (
    const std::vector< double > & prob,
    const Container & X,
    typename std::enable_if< is_iterable< Container >::value >::type * = nullptr )
```

Average.

Parameters

<i>prob</i>	Real probability vector representing the probability distribution of <i>X</i>
<i>X</i>	Real random variable values represented by an STL-like container

## Returns

Average of  $X$

## 6.1.3.15 bloch2rho()

```
cmat qpp::bloch2rho (
    const std::vector< double > & r ) [inline]
```

Computes the density matrix corresponding to the 3-dimensional real Bloch vector  $r$ .

## See also

[qpp::rho2bloch\(\)](#)

## Parameters

$r$	3-dimensional real vector
-----	---------------------------

## Returns

Qubit density matrix

## 6.1.3.16 choi2kraus()

```
std::vector<cmat> qpp::choi2kraus (
    const cmat & A ) [inline]
```

Orthogonal Kraus operators from Choi matrix.

## See also

[qpp::kraus2choi\(\)](#)

Extracts a set of orthogonal (under Hilbert-Schmidt operator norm) Kraus operators from the Choi matrix  $A$

## Note

The Kraus operators satisfy  $Tr(K_i^\dagger K_j) = \delta_{ij}$  for all  $i \neq j$

## Parameters

$A$	Choi matrix
-----	-------------

**Returns**

Set of orthogonal Kraus operators

**6.1.3.17 choi2super()**

```
cmat qpp::choi2super (
    const cmat & A ) [inline]
```

Converts Choi matrix to superoperator matrix.

**See also**

[qpp::super2choi\(\)](#)

**Parameters**

<i>A</i>	Choi matrix
----------	-------------

**Returns**

Superoperator matrix

**6.1.3.18 comm()**

```
template<typename Derived1 , typename Derived2 >
dyn_mat<typename Derived1::Scalar> qpp::comm (
    const Eigen::MatrixBase< Derived1 > & A,
    const Eigen::MatrixBase< Derived2 > & B )
```

Commutator.

**See also**

[qpp::anticomm\(\)](#)

Commutator  $[A, B] = AB - BA$ . Both *A* and *B* must be Eigen expressions over the same scalar field.

**Parameters**

<i>A</i>	Eigen expression
<i>B</i>	Eigen expression

## Returns

Commutator  $AB - BA$ , as a dynamic matrix over the same scalar field as  $A$

## 6.1.3.19 complement()

```
template<typename T >
std::vector<T> qpp::complement (
    std::vector< T > subsys,
    idx N )
```

Constructs the complement of a subsystem vector.

## Parameters

<i>subsys</i>	Subsystem vector
<i>N</i>	Total number of systems

## Returns

Complement of *subsys* with respect to the set  $\{0, 1, \dots, N - 1\}$

## 6.1.3.20 compperm()

```
std::vector<idx> qpp::comperm (
    const std::vector< idx > & perm,
    const std::vector< idx > & sigma ) [inline]
```

Compose permutations.

## Parameters

<i>perm</i>	Permutation
<i>sigma</i>	Permutation

## Returns

Composition of the permutations  $perm \circ sigma = perm(sigma)$

## 6.1.3.21 concurrence()

```
template<typename Derived >
double qpp::concurrence (
    const Eigen::MatrixBase< Derived > & A )
```

Wootters concurrence of the bi-partite qubit mixed state  $A$ .



## Parameters

<i>A</i>	Eigen expression
----------	------------------

## Returns

Wootters concurrence

## 6.1.3.22 conjugate()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::conjugate (
    const Eigen::MatrixBase< Derived > & A )
```

Complex conjugate.

## Parameters

<i>A</i>	Eigen expression
----------	------------------

## Returns

Complex conjugate of *A*, as a dynamic matrix over the same scalar field as *A*

## 6.1.3.23 contfrac2x()

```
double qpp::contfrac2x (
    const std::vector< int > & cf,
    idx N = idx(-1) ) [inline]
```

Real representation of a simple continued fraction.

## See also

[qpp::x2contfrac\(\)](#)

## Note

If *N* is greater than the size of *cf* (by default it is), then all terms in *cf* are considered.

## Parameters

<i>cf</i>	Integer vector containing the simple continued fraction expansion
<i>N</i>	Number of terms considered in the continued fraction expansion.

**Returns**

Real representation of the simple continued fraction

**6.1.3.24 cor()**

```
template<typename Container >
double qpp::cor (
    const dmat & probXY,
    const Container & X,
    const Container & Y,
    typename std::enable_if< is_iterable< Container >::value >::type * = nullptr )
```

Correlation.

**Parameters**

<i>probXY</i>	Real matrix representing the joint probability distribution of <i>X</i> and <i>Y</i> in lexicographical order ( <i>X</i> labels the rows, <i>Y</i> labels the columns)
<i>X</i>	Real random variable values represented by an STL-like container
<i>Y</i>	Real random variable values represented by an STL-like container

**Returns**

Correlation of *X* and *Y*

**6.1.3.25 cosm()**

```
template<typename Derived >
cmat qpp::cosm (
    const Eigen::MatrixBase< Derived > & A )
```

Matrix cos.

**Parameters**

<i>A</i>	Eigen expression
----------	------------------

**Returns**

Matrix cosine of *A*

## 6.1.3.26 cov()

```
template<typename Container >
double qpp::cov (
    const dmat & probXY,
    const Container & X,
    const Container & Y,
    typename std::enable_if< is_iterable< Container >::value >::type * = nullptr )
```

Covariance.

## Parameters

<i>probXY</i>	Real matrix representing the joint probability distribution of <i>X</i> and <i>Y</i> in lexicographical order ( <i>X</i> labels the rows, <i>Y</i> labels the columns)
<i>X</i>	Real random variable values represented by an STL-like container
<i>Y</i>	Real random variable values represented by an STL-like container

## Returns

Covariance of *X* and *Y*

## 6.1.3.27 cwise()

```
template<typename OutputScalar , typename Derived >
dyn_mat<OutputScalar> qpp::cwise (
    const Eigen::MatrixBase< Derived > & A,
    OutputScalar(*) (const typename Derived::Scalar &) f )
```

Functor.

## Parameters

<i>A</i>	Eigen expression
<i>f</i>	Pointer-to-function from scalars of <i>A</i> to <i>OutputScalar</i>

## Returns

Component-wise  $f(A)$ , as a dynamic matrix over the *OutputScalar* scalar field

## 6.1.3.28 det()

```
template<typename Derived >
Derived::Scalar qpp::det (
    const Eigen::MatrixBase< Derived > & A )
```

Determinant.

**Parameters**

<i>A</i>	Eigen expression
----------	------------------

**Returns**

Determinant of *A*, as a scalar over the same scalar field as *A*. Returns  $\pm\infty$  when the determinant overflows/underflows.

**6.1.3.29 `dirsum()`** [1/4]

```
template<typename T >
dyn_mat<typename T::Scalar> qpp::dirsum (
    const T & head )
```

Direct sum.

**See also**

[qpp::dirsumpow\(\)](#)

Used to stop the recursion for the variadic template version of [qpp::dirsum\(\)](#)

**Parameters**

<i>head</i>	Eigen expression
-------------	------------------

**Returns**

Its argument *head*

**6.1.3.30 `dirsum()`** [2/4]

```
template<typename T , typename... Args>
dyn_mat<typename T::Scalar> qpp::dirsum (
    const T & head,
    const Args &... tail )
```

Direct sum.

**See also**

[qpp::dirsumpow\(\)](#)

## Parameters

<i>head</i>	Eigen expression
<i>tail</i>	Variadic Eigen expression (zero or more parameters)

## Returns

Direct sum of all input parameters, evaluated from left to right, as a dynamic matrix over the same scalar field as its arguments

6.1.3.31 `dirsum()` [3/4]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::dirsum (
    const std::vector< Derived > & As )
```

Direct sum.

## See also

[qpp::dirsumpow\(\)](#)

## Parameters

<i>As</i>	std::vector of Eigen expressions
-----------	----------------------------------

## Returns

Direct sum of all elements in *As*, evaluated from left to right, as a dynamic matrix over the same scalar field as its arguments

6.1.3.32 `dirsum()` [4/4]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::dirsum (
    const std::initializer_list< Derived > & As )
```

Direct sum.

## See also

[qpp::dirsumpow\(\)](#)

## Parameters

<i>As</i>	std::initializer_list of Eigen expressions, such as { <i>A1</i> , <i>A2</i> , ... , <i>Ak</i> }
-----------	---

## Returns

Direct sum of all elements in *As*, evaluated from left to right, as a dynamic matrix over the same scalar field as its arguments

## 6.1.3.33 dirsumpow()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::dirsumpow (
    const Eigen::MatrixBase< Derived > & A,
    idx n )
```

Direct sum power.

## See also

[qpp::dirsum\(\)](#)

## Parameters

<i>A</i>	Eigen expression
<i>n</i>	Non-negative integer

## Returns

Direct sum of *A* with itself *n* times  $A^{\oplus n}$ , as a dynamic matrix over the same scalar field as *A*

## 6.1.3.34 disp() [1/5]

```
template<typename Derived >
internal::IOManipEigen qpp::disp (
    const Eigen::MatrixBase< Derived > & A,
    double chop = qpp::chop )
```

Eigen expression ostream manipulator.

## Parameters

<i>A</i>	Eigen expression
<i>chop</i>	Set to zero the elements smaller in absolute value than <i>chop</i>

## Returns

Instance of [qpp::internal::IOManipEigen](#)

6.1.3.35 `disp()` [2/5]

```
internal::IOManipEigen qpp::disp (
    cplx z,
    double chop = qpp::chop ) [inline]
```

Complex number ostream manipulator.

## Parameters

<i>z</i>	Complex number (or any other type implicitly cast-able to <code>std::complex&lt;double&gt;</code> )
<i>chop</i>	Set to zero the elements smaller in absolute value than <i>chop</i>

## Returns

Instance of [qpp::internal::IOManipEigen](#)

6.1.3.36 `disp()` [3/5]

```
template<typename InputIterator >
internal::IOManipRange<InputIterator> qpp::disp (
    InputIterator first,
    InputIterator last,
    const std::string & separator,
    const std::string & start = "[",
    const std::string & end = "]" )
```

Range ostream manipulator.

## Parameters

<i>first</i>	Iterator to the first element of the range
<i>last</i>	Iterator to the last element of the range
<i>separator</i>	Separator
<i>start</i>	Left marking
<i>end</i>	Right marking

## Returns

Instance of [qpp::internal::IOManipRange](#)

6.1.3.37 `disp()` [4/5]

```
template<typename Container >
internal::IOManipRange<typename Container::const_iterator> qpp::disp (
    const Container & c,
    const std::string & separator,
    const std::string & start = "[",
    const std::string & end = "]",
    typename std::enable_if< is_iterable< Container >::value >::type * = nullptr )
```

Standard container ostream manipulator. The container must support `std::begin()`, `std::end()` and forward iteration.

## Parameters

<i>c</i>	Container
<i>separator</i>	Separator
<i>start</i>	Left marking
<i>end</i>	Right marking

## Returns

Instance of `qpp::internal::IOManipRange`

6.1.3.38 `disp()` [5/5]

```
template<typename PointerType >
internal::IOManipPointer<PointerType> qpp::disp (
    const PointerType * p,
    idx N,
    const std::string & separator,
    const std::string & start = "[",
    const std::string & end = "]" )
```

C-style pointer ostream manipulator.

## Parameters

<i>p</i>	Pointer to the first element
<i>N</i>	Number of elements to be displayed
<i>separator</i>	Separator
<i>start</i>	Left marking
<i>end</i>	Right marking

## Returns

Instance of `qpp::internal::IOManipPointer`



## 6.1.3.39 egcd()

```
std::tuple<bigint, bigint, bigint> qpp::egcd (
    bigint a,
    bigint b ) [inline]
```

Extended greatest common divisor of two integers.

See also

[qpp::gcd\(\)](#)

## Parameters

<i>a</i>	Integer
<i>b</i>	Integer

## Returns

Tuple of: 1. Integer  $m$ , 2. Integer  $n$ , and 3. Non-negative integer  $\gcd(a, b)$  such that  $ma + nb = \gcd(a, b)$

## 6.1.3.40 eig()

```
template<typename Derived >
std::pair<dyn_col_vect<cplx>, cmat> qpp::eig (
    const Eigen::MatrixBase< Derived > & A )
```

Full eigen decomposition.

See also

[qpp::heig\(\)](#)

## Parameters

<i>A</i>	Eigen expression
----------	------------------

## Returns

Pair of: 1. Eigenvalues of  $A$ , as a complex dynamic column vector, and 2. Eigenvectors of  $A$ , as columns of a complex dynamic matrix

## 6.1.3.41 entanglement() [1/2]

```
template<typename Derived >
double qpp::entanglement (
```

```
const Eigen::MatrixBase< Derived > & A,
const std::vector< idx > & dims )
```

Entanglement of the bi-partite pure state  $A$ .

Defined as the von-Neumann entropy of the reduced density matrix of one of the subsystems

See also

[qpp::entropy\(\)](#)

#### Parameters

$A$	Eigen expression
$dims$	Dimensions of the bi-partite system

#### Returns

Entanglement, with the logarithm in base 2

#### 6.1.3.42 entanglement() [2/2]

```
template<typename Derived >
double qpp::entanglement (
    const Eigen::MatrixBase< Derived > & A,
    idx d = 2 )
```

Entanglement of the bi-partite pure state  $A$ .

Defined as the von-Neumann entropy of the reduced density matrix of one of the subsystems

See also

[qpp::entropy\(\)](#)

#### Parameters

$A$	Eigen expression
$d$	Subsystem dimensions

#### Returns

Entanglement, with the logarithm in base 2

**6.1.3.43 entropy()** [1/2]

```
template<typename Derived >
double qpp::entropy (
    const Eigen::MatrixBase< Derived > & A )
```

von-Neumann entropy of the density matrix *A*

**Parameters**

<i>A</i>	Eigen expression
----------	------------------

**Returns**

von-Neumann entropy, with the logarithm in base 2

**6.1.3.44 entropy()** [2/2]

```
double qpp::entropy (
    const std::vector< double > & prob ) [inline]
```

Shannon entropy of the probability distribution *prob*.

**Parameters**

<i>prob</i>	Real probability vector
-------------	-------------------------

**Returns**

Shannon entropy, with the logarithm in base 2

**6.1.3.45 evals()**

```
template<typename Derived >
dyn_col_vect<cplx> qpp::evals (
    const Eigen::MatrixBase< Derived > & A )
```

Eigenvalues.

**See also**

[qpp::hevals\(\)](#)

**Parameters**

$A$	Eigen expression
-----	------------------

**Returns**

Eigenvalues of  $A$ , as a complex dynamic column vector

**6.1.3.46 `evecs()`**

```
template<typename Derived >
cmat qpp::evecs (
    const Eigen::MatrixBase< Derived > & A )
```

Eigenvectors.

**See also**

[`qpp::hevecs\(\)`](#)

**Parameters**

$A$	Eigen expression
-----	------------------

**Returns**

Eigenvectors of  $A$ , as columns of a complex dynamic matrix

**6.1.3.47 `expm()`**

```
template<typename Derived >
cmat qpp::expm (
    const Eigen::MatrixBase< Derived > & A )
```

Matrix exponential.

**Parameters**

$A$	Eigen expression
-----	------------------

**Returns**

Matrix exponential of  $A$

## 6.1.3.48 factors()

```
std::vector<bigint> qpp::factors (
    bigint a ) [inline]
```

Prime factor decomposition.

## Note

Runs in  $\mathcal{O}(\sqrt{n})$  time complexity

## Parameters

<i>a</i>	Integer different from 0, 1 or -1
----------	-----------------------------------

## Returns

Integer vector containing the factors

## 6.1.3.49 funm()

```
template<typename Derived >
cmat qpp::funm (
    const Eigen::MatrixBase< Derived > & A,
    cplx(*) (const cplx &) f )
```

Functional calculus  $f(A)$

## Parameters

<i>A</i>	Eigen expression
<i>f</i>	Pointer-to-function from complex to complex

## Returns

$f(A)$

## 6.1.3.50 gcd() [1/2]

```
bigint qpp::gcd (
    bigint a,
    bigint b ) [inline]
```

Greatest common divisor of two integers.

## See also

[qpp::lcm\(\)](#)

**Parameters**

<i>a</i>	Integer
<i>b</i>	Integer

**Returns**

Greatest common divisor of *a* and *b*

**6.1.3.51 gcd()** [2/2]

```
bigint qpp::gcd (
    const std::vector< bigint > & as ) [inline]
```

Greatest common divisor of a list of integers.

**See also**

[qpp::lcm\(\)](#)

**Parameters**

<i>as</i>	List of integers
-----------	------------------

**Returns**

Greatest common divisor of all numbers in *as*

**6.1.3.52 gconcurrency()**

```
template<typename Derived >
double qpp::gconcurrency (
    const Eigen::MatrixBase< Derived > & A )
```

G-concurrency of the bi-partite pure state *A*.

**Note**

Both local dimensions must be equal

Uses [qpp::logdet\(\)](#) to avoid overflows

**See also**

[qpp::logdet\(\)](#)

## Parameters

<i>A</i>	Eigen expression
----------	------------------

## Returns

G-concurrence

6.1.3.53 `grams()` [1/3]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::grams (
    const std::vector< Derived > & As )
```

Gram-Schmidt orthogonalization.

## Parameters

<i>As</i>	std::vector of Eigen expressions as column vectors
-----------	--

## Returns

Gram-Schmidt vectors of *As* as columns of a dynamic matrix over the same scalar field as its arguments

6.1.3.54 `grams()` [2/3]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::grams (
    const std::initializer_list< Derived > & As )
```

Gram-Schmidt orthogonalization.

## Parameters

<i>As</i>	std::initializer_list of Eigen expressions as column vectors
-----------	--

## Returns

Gram-Schmidt vectors of *As* as columns of a dynamic matrix over the same scalar field as its arguments

**6.1.3.55** `grams()` [3/3]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::grams (
    const Eigen::MatrixBase< Derived > & A )
```

Gram-Schmidt orthogonalization.

**Parameters**

<i>A</i>	Eigen expression, the input vectors are the columns of <i>A</i>
----------	---

**Returns**

Gram-Schmidt vectors of the columns of *A*, as columns of a dynamic matrix over the same scalar field as *A*

**6.1.3.56** `heig()`

```
template<typename Derived >
std::pair<dyn_col_vect<double>, cmat> qpp::heig (
    const Eigen::MatrixBase< Derived > & A )
```

Full eigen decomposition of Hermitian expression.

**See also**

[qpp::eig\(\)](#)

**Parameters**

<i>A</i>	Eigen expression
----------	------------------

**Returns**

Pair of: 1. Eigenvalues of *A*, as a real dynamic column vector, and 2. Eigenvectors of *A*, as columns of a complex dynamic matrix

**6.1.3.57** `hevals()`

```
template<typename Derived >
dyn_col_vect<double> qpp::hevals (
    const Eigen::MatrixBase< Derived > & A )
```

Hermitian eigenvalues.

**See also**

[qpp::evals\(\)](#)



**Parameters**

$A$	Eigen expression
-----	------------------

**Returns**

Eigenvalues of Hermitian  $A$ , as a real dynamic column vector

**6.1.3.58 hevects()**

```
template<typename Derived >
cmat qpp::hevects (
    const Eigen::MatrixBase< Derived > & A )
```

Hermitian eigenvectors.

**See also**

[qpp::evects\(\)](#)

**Parameters**

$A$	Eigen expression
-----	------------------

**Returns**

Eigenvectors of Hermitian  $A$ , as columns of a complex matrix

**6.1.3.59 inverse()**

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::inverse (
    const Eigen::MatrixBase< Derived > & A )
```

Inverse.

**Parameters**

$A$	Eigen expression
-----	------------------

**Returns**

Inverse of  $A$ , as a dynamic matrix over the same scalar field as  $A$

### 6.1.3.60 invperm()

```
std::vector<idx> qpp::invperm (
    const std::vector< idx > & perm ) [inline]
```

Inverse permutation.

#### Parameters

<i>perm</i>	Permutation
-------------	-------------

#### Returns

Inverse of the permutation *perm*

### 6.1.3.61 ip() [1/2]

```
template<typename Derived >
dyn_col_vect<typename Derived::Scalar> qpp::ip (
    const Eigen::MatrixBase< Derived > & phi,
    const Eigen::MatrixBase< Derived > & psi,
    const std::vector< idx > & subsys,
    const std::vector< idx > & dims )
```

Generalized inner product.

#### Parameters

<i>phi</i>	Column vector Eigen expression
<i>psi</i>	Column vector Eigen expression
<i>subsys</i>	Subsystem indexes over which <i>phi</i> is defined
<i>dims</i>	Dimensions of the multi-partite system

#### Returns

Inner product  $\langle \phi_{subsys} | \psi \rangle$ , as a scalar or column vector over the remaining Hilbert space

### 6.1.3.62 ip() [2/2]

```
template<typename Derived >
dyn_col_vect<typename Derived::Scalar> qpp::ip (
    const Eigen::MatrixBase< Derived > & phi,
    const Eigen::MatrixBase< Derived > & psi,
    const std::vector< idx > & subsys,
    idx d = 2 )
```

Generalized inner product.

## Parameters

<i>phi</i>	Column vector Eigen expression
<i>psi</i>	Column vector Eigen expression
<i>subsys</i>	Subsystem indexes over which <i>phi</i> is defined
<i>d</i>	Subsystem dimensions

## Returns

Inner product  $\langle \phi_{subsys} | \psi \rangle$ , as a scalar or column vector over the remaining Hilbert space

## 6.1.3.63 isprime()

```
bool qpp::isprime (
    bigint p,
    idx k = 80 ) [inline]
```

Primality test based on the Miller-Rabin's algorithm.

## Parameters

<i>p</i>	Integer different from 0, 1 or -1
<i>k</i>	Number of iterations. The probability of a false positive is $2^{-k}$ .

## Returns

True if the number is (most-likely) prime, false otherwise

## 6.1.3.64 kraus2choi()

```
cmat qpp::kraus2choi (
    const std::vector< cmat > & Ks ) [inline]
```

Choi matrix.

## See also

[qpp::choi2kraus\(\)](#)

Constructs the Choi matrix of the channel specified by the set of Kraus operators *Ks* in the standard operator basis  $\{|i\rangle\langle j|\}$  ordered in lexicographical order, i.e.  $|0\rangle\langle 0|$ ,  $|0\rangle\langle 1|$  etc.

## Note

The superoperator matrix *S* and the Choi matrix *C* are related by  $S_{ab,mn} = C_{ma,nb}$

**Parameters**

<i>Ks</i>	Set of Kraus operators
-----------	------------------------

**Returns**

Choi matrix

**6.1.3.65 kraus2super()**

```
cmat qpp::kraus2super (
    const std::vector< cmat > & Ks ) [inline]
```

Superoperator matrix.

Constructs the superoperator matrix of the channel specified by the set of Kraus operators  $K_s$  in the standard operator basis  $\{|i\rangle\langle j|\}$  ordered in lexicographical order, i.e.  $|0\rangle\langle 0|$ ,  $|0\rangle\langle 1|$  etc.

**Parameters**

<i>Ks</i>	Set of Kraus operators
-----------	------------------------

**Returns**

Superoperator matrix

**6.1.3.66 kron()** [1/4]

```
template<typename T >
dyn_mat<typename T::Scalar> qpp::kron (
    const T & head )
```

Kronecker product.

**See also**

[qpp::kronpow\(\)](#)

Used to stop the recursion for the variadic template version of [qpp::kron\(\)](#)

**Parameters**

<i>head</i>	Eigen expression
-------------	------------------

**Returns**

Its argument *head*

**6.1.3.67 kron()** [2/4]

```
template<typename T , typename... Args>
dyn_mat<typename T::Scalar> qpp::kron (
    const T & head,
    const Args &... tail )
```

Kronecker product.

**See also**

[qpp::kronpow\(\)](#)

**Parameters**

<i>head</i>	Eigen expression
<i>tail</i>	Variadic Eigen expression (zero or more parameters)

**Returns**

Kronecker product of all input parameters, evaluated from left to right, as a dynamic matrix over the same scalar field as its arguments

**6.1.3.68 kron()** [3/4]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::kron (
    const std::vector< Derived > & As )
```

Kronecker product.

**See also**

[qpp::kronpow\(\)](#)

**Parameters**

<i>As</i>	std::vector of Eigen expressions
-----------	----------------------------------

**Returns**

Kronecker product of all elements in  $As$ , evaluated from left to right, as a dynamic matrix over the same scalar field as its arguments

**6.1.3.69 kron()** [ 4 / 4 ]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::kron (
    const std::initializer_list< Derived > & As )
```

Kronecker product.

**See also**

[qpp::kronpow\(\)](#)

**Parameters**

$As$	std::initializer_list of Eigen expressions, such as $\{A1, A2, \dots, Ak\}$
------	---

**Returns**

Kronecker product of all elements in  $As$ , evaluated from left to right, as a dynamic matrix over the same scalar field as its arguments

**6.1.3.70 kronpow()**

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::kronpow (
    const Eigen::MatrixBase< Derived > & A,
    idx n )
```

Kronecker power.

**See also**

[qpp::kron\(\)](#)

**Parameters**

$A$	Eigen expression
$n$	Non-negative integer

**Returns**

Kronecker product of  $A$  with itself  $n$  times  $A^{\otimes n}$ , as a dynamic matrix over the same scalar field as  $A$

**6.1.3.71 lcm()** [1/2]

```
bigint qpp::lcm (
    bigint a,
    bigint b ) [inline]
```

Least common multiple of two integers.

**See also**

[qpp::gcd\(\)](#)

**Parameters**

$a$	Integer
$b$	Integer

**Returns**

Least common multiple of  $a$  and  $b$

**6.1.3.72 lcm()** [2/2]

```
bigint qpp::lcm (
    const std::vector< bigint > & as ) [inline]
```

Least common multiple of a list of integers.

**See also**

[qpp::gcd\(\)](#)

**Parameters**

$as$	List of integers
------	------------------

**Returns**

Least common multiple of all numbers in  $as$

### 6.1.3.73 load()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::load (
    const std::string & fname )
```

Loads Eigen matrix from a binary file (internal format) in double precision.

See also

[qpp::save\(\)](#)

The template parameter cannot be automatically deduced and must be explicitly provided, depending on the scalar field of the matrix that is being loaded.

Example:

```
// loads a previously saved Eigen dynamic complex matrix from "input.bin"
cmat mat = load<cmat>("input.bin");
```

Parameters

<i>fname</i>	Output file name
--------------	------------------

### 6.1.3.74 loadMATLAB() [1/2]

```
template<typename Derived >
std::enable_if<std::is_same<typename Derived::Scalar, cplx>::value, dyn_mat<cplx> >::type
qpp::loadMATLAB (
    const std::string & mat_file,
    const std::string & var_name )
```

Loads a complex Eigen dynamic matrix from a MATLAB .mat file,.

See also

[qpp::saveMATLAB\(\)](#)

The template parameter cannot be automatically deduced and must be explicitly provided

Example:

```
// loads a previously saved Eigen ket
// from the MATLAB file "input.mat"
ket psi = loadMATLAB<ket>("input.mat");
```

Template Parameters

<i>Derived</i>	Complex Eigen type
----------------	--------------------



## Parameters

<i>mat_file</i>	MATALB .mat file
<i>var_name</i>	Variable name in the .mat file representing the matrix to be loaded

## Returns

Eigen dynamic matrix

## 6.1.3.75 loadMATLAB() [2/2]

```
template<typename Derived >
std::enable_if<!std::is_same<typename Derived::Scalar, cplx>::value, dyn_mat<typename Derived↵
::Scalar> >::type qpp::loadMATLAB (
    const std::string & mat_file,
    const std::string & var_name )
```

Loads a non-complex Eigen dynamic matrix from a MATLAB .mat file,.

## See also

[qpp::saveMATLAB\(\)](#)

The template parameter cannot be automatically deduced and must be explicitly provided

## Example:

```
// loads a previously saved Eigen dynamic double matrix
// from the MATLAB file "input.mat"
dmat mat = loadMATLAB<dmat>("input.mat");
```

## Template Parameters

<i>Derived</i>	Non-complex Eigen type
----------------	------------------------

## Parameters

<i>mat_file</i>	MATALB .mat file
<i>var_name</i>	Variable name in the .mat file representing the matrix to be loaded

## Returns

Eigen dynamic matrix

**6.1.3.76 logdet()**

```
template<typename Derived >
Derived::Scalar qpp::logdet (
    const Eigen::MatrixBase< Derived > & A )
```

Logarithm of the determinant.

Useful when the determinant overflows/underflows

**Parameters**

<i>A</i>	Eigen expression
----------	------------------

**Returns**

Logarithm of the determinant of *A*, as a scalar over the same scalar field as *A*

**6.1.3.77 logm()**

```
template<typename Derived >
cmat qpp::logm (
    const Eigen::MatrixBase< Derived > & A )
```

Matrix logarithm.

**Parameters**

<i>A</i>	Eigen expression
----------	------------------

**Returns**

Matrix logarithm of *A*

**6.1.3.78 lognegativity()** [1/2]

```
template<typename Derived >
double qpp::lognegativity (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< idx > & dims )
```

Logarithmic negativity of the bi-partite mixed state *A*.

**Parameters**

<i>A</i>	Eigen expression
<i>dims</i>	Dimensions of the bi-partite system

**Returns**

Logarithmic negativity, with the logarithm in base 2

**6.1.3.79 lognegativity()** [2/2]

```
template<typename Derived >
double qpp::lognegativity (
    const Eigen::MatrixBase< Derived > & A,
    idx d = 2 )
```

Logarithmic negativity of the bi-partite mixed state  $A$ .

**Parameters**

$A$	Eigen expression
$d$	Subsystem dimensions

**Returns**

Logarithmic negativity, with the logarithm in base 2

**6.1.3.80 marginalX()**

```
std::vector<double> qpp::marginalX (
    const dmat & probXY ) [inline]
```

Marginal distribution.

**Parameters**

$probXY$	Real matrix representing the joint probability distribution of $X$ and $Y$ in lexicographical order ( $X$ labels the rows, $Y$ labels the columns)
----------	--

**Returns**

Real vector consisting of the marginal distribution of  $X$

**6.1.3.81 marginalY()**

```
std::vector<double> qpp::marginalY (
    const dmat & probXY ) [inline]
```

Marginal distribution.

**Parameters**

<i>probXY</i>	Real matrix representing the joint probability distribution of $X$ and $Y$ in lexicographical order ( $X$ labels the rows, $Y$ labels the columns)
---------------	--

**Returns**

Real vector consisting of the marginal distribution of  $Y$

**6.1.3.82 `measure()`** [1/9]

```
template<typename Derived >
std::tuple<idx, std::vector<double>, std::vector<cmat> > qpp::measure (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< cmat > & Ks )
```

Measures the state  $A$  using the set of Kraus operators  $Ks$ .

**Parameters**

$A$	Eigen expression
$Ks$	Set of Kraus operators

**Returns**

Tuple of: 1. Result of the measurement, 2. Vector of outcome probabilities, and 3. Vector of post-measurement normalized states

**6.1.3.83 `measure()`** [2/9]

```
template<typename Derived >
std::tuple<idx, std::vector<double>, std::vector<cmat> > qpp::measure (
    const Eigen::MatrixBase< Derived > & A,
    const std::initializer_list< cmat > & Ks )
```

Measures the state  $A$  using the set of Kraus operators  $Ks$ .

**Parameters**

$A$	Eigen expression
$Ks$	Set of Kraus operators

## Returns

Tuple of: 1. Result of the measurement, 2. Vector of outcome probabilities, and 3. Vector of post-measurement normalized states

6.1.3.84 `measure()` [3/9]

```
template<typename Derived >
std::tuple<idx, std::vector<double>, std::vector<cmat> > qpp::measure (
    const Eigen::MatrixBase< Derived > & A,
    const cmat & U )
```

Measures the state  $A$  in the orthonormal basis specified by the unitary matrix  $U$ .

## Parameters

$A$	Eigen expression
$U$	Unitary matrix whose columns represent the measurement basis vectors

## Returns

Tuple of: 1. Result of the measurement, 2. Vector of outcome probabilities, and 3. Vector of post-measurement normalized states

6.1.3.85 `measure()` [4/9]

```
template<typename Derived >
std::tuple<idx, std::vector<double>, std::vector<cmat> > qpp::measure (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< cmat > & Ks,
    const std::vector< idx > & subsys,
    const std::vector< idx > & dims )
```

Measures the part *subsys* of the multi-partite state vector or density matrix  $A$  using the set of Kraus operators  $Ks$ .

## See also

[qpp::measure\\_seq\(\)](#)

## Note

The dimension of all  $Ks$  must match the dimension of *subsys*. The measurement is destructive, i.e. the measured subsystems are traced away.

## Parameters

<i>A</i>	Eigen expression
<i>Ks</i>	Set of Kraus operators
<i>subsys</i>	Subsystem indexes that are measured
<i>dims</i>	Dimensions of the multi-partite system

## Returns

Tuple of: 1. Result of the measurement, 2. Vector of outcome probabilities, and 3. Vector of post-measurement normalized states

6.1.3.86 `measure()` [5/9]

```
template<typename Derived >
std::tuple<idx, std::vector<double>, std::vector<cmat> > qpp::measure (
    const Eigen::MatrixBase< Derived > & A,
    const std::initializer_list< cmat > & Ks,
    const std::vector< idx > & subsys,
    const std::vector< idx > & dims )
```

Measures the part *subsys* of the multi-partite state vector or density matrix *A* using the set of Kraus operators *Ks*.

## See also

[qpp::measure\\_seq\(\)](#)

## Note

The dimension of all *Ks* must match the dimension of *subsys*. The measurement is destructive, i.e. the measured subsystems are traced away.

## Parameters

<i>A</i>	Eigen expression
<i>Ks</i>	Set of Kraus operators
<i>subsys</i>	Subsystem indexes that are measured
<i>dims</i>	Dimensions of the multi-partite system

## Returns

Tuple of: 1. Result of the measurement, 2. Vector of outcome probabilities, and 3. Vector of post-measurement normalized states

**6.1.3.87** `measure()` [6/9]

```
template<typename Derived >
std::tuple<idx, std::vector<double>, std::vector<cmat> > qpp::measure (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< cmat > & Ks,
    const std::vector< idx > & subsys,
    idx d = 2 )
```

Measures the part *subsys* of the multi-partite state vector or density matrix *A* using the set of Kraus operators *Ks*.

See also

[qpp::measure\\_seq\(\)](#)

Note

The dimension of all *Ks* must match the dimension of *subsys*. The measurement is destructive, i.e. the measured subsystems are traced away.

Parameters

<i>A</i>	Eigen expression
<i>Ks</i>	Set of Kraus operators
<i>subsys</i>	Subsystem indexes that are measured
<i>d</i>	Subsystem dimensions

Returns

Tuple of: 1. Result of the measurement, 2. Vector of outcome probabilities, and 3. Vector of post-measurement normalized states

**6.1.3.88** `measure()` [7/9]

```
template<typename Derived >
std::tuple<idx, std::vector<double>, std::vector<cmat> > qpp::measure (
    const Eigen::MatrixBase< Derived > & A,
    const std::initializer_list< cmat > & Ks,
    const std::vector< idx > & subsys,
    idx d = 2 )
```

Measures the part *subsys* of the multi-partite state vector or density matrix *A* using the set of Kraus operators *Ks*.

See also

[qpp::measure\\_seq\(\)](#)

Note

The dimension of all *Ks* must match the dimension of *subsys*. The measurement is destructive, i.e. the measured subsystems are traced away.

## Parameters

<i>A</i>	Eigen expression
<i>Ks</i>	Set of Kraus operators
<i>subsys</i>	Subsystem indexes that are measured
<i>d</i>	Subsystem dimensions

## Returns

Tuple of: 1. Result of the measurement, 2. Vector of outcome probabilities, and 3. Vector of post-measurement normalized states

6.1.3.89 `measure()` [8/9]

```
template<typename Derived >
std::tuple<idx, std::vector<double>, std::vector<cmat> > qpp::measure (
    const Eigen::MatrixBase< Derived > & A,
    const cmat & V,
    const std::vector< idx > & subsys,
    const std::vector< idx > & dims )
```

Measures the part *subsys* of the multi-partite state vector or density matrix *A* in the orthonormal basis or rank-1 POVM specified by the matrix *V*.

## See also

[qpp::measure\\_seq\(\)](#)

## Note

The dimension of *V* must match the dimension of *subsys*. The measurement is destructive, i.e. the measured subsystems are traced away.

## Parameters

<i>A</i>	Eigen expression
<i>V</i>	Matrix whose columns represent the measurement basis vectors or the bra parts of the rank-1 POVM
<i>subsys</i>	Subsystem indexes that are measured
<i>dims</i>	Dimensions of the multi-partite system

## Returns

Tuple of: 1. Result of the measurement, 2. Vector of outcome probabilities, and 3. Vector of post-measurement normalized states



**6.1.3.90** `measure()` [9/9]

```
template<typename Derived >
std::tuple<idx, std::vector<double>, std::vector<cmat> > qpp::measure (
    const Eigen::MatrixBase< Derived > & A,
    const cmat & V,
    const std::vector< idx > & subsys,
    idx d = 2 )
```

Measures the part *subsys* of the multi-partite state vector or density matrix *A* in the orthonormal basis or rank-1 POVM specified by the matrix *V*.

See also

[qpp::measure\\_seq\(\)](#)

Note

The dimension of *V* must match the dimension of *subsys*. The measurement is destructive, i.e. the measured subsystems are traced away.

Parameters

<i>A</i>	Eigen expression
<i>V</i>	Matrix whose columns represent the measurement basis vectors or the bra parts of the rank-1 POVM
<i>subsys</i>	Subsystem indexes that are measured
<i>d</i>	Subsystem dimensions

Returns

Tuple of: 1. Result of the measurement, 2. Vector of outcome probabilities, and 3. Vector of post-measurement normalized states

**6.1.3.91** `measure_seq()` [1/2]

```
template<typename Derived >
std::tuple<std::vector<idx>, double, cmat> qpp::measure_seq (
    const Eigen::MatrixBase< Derived > & A,
    std::vector< idx > subsys,
    std::vector< idx > dims )
```

Sequentially measures the part *subsys* of the multi-partite state vector or density matrix *A* in the computational basis.

See also

[qpp::measure\(\)](#)

## Parameters

<i>A</i>	Eigen expression
<i>subsys</i>	Subsystem indexes that are measured
<i>dims</i>	Dimensions of the multi-partite system

## Returns

Tuple of: 1. Vector of outcome results of the measurement (ordered in increasing order with respect to *subsys*, i.e. first measurement result corresponds to the subsystem with the smallest index), 2. Outcome probability, and 3. Post-measurement normalized state

6.1.3.92 `measure_seq()` [2/2]

```
template<typename Derived >
std::tuple<std::vector<idx>, double, cmat> qpp::measure_seq (
    const Eigen::MatrixBase< Derived > & A,
    std::vector< idx > subsys,
    idx d = 2 )
```

Sequentially measures the part *subsys* of the multi-partite state vector or density matrix *A* in the computational basis.

## See also

[qpp::measure\(\)](#)

## Parameters

<i>A</i>	Eigen expression
<i>subsys</i>	Subsystem indexes that are measured
<i>d</i>	Subsystem dimensions

## Returns

Tuple of: 1. Vector of outcome results of the measurement (ordered in increasing order with respect to *subsys*, i.e. first measurement result corresponds to the subsystem with the smallest index), 2. Outcome probability, and 3. Post-measurement normalized state

6.1.3.93 `mket()` [1/2]

```
ket qpp::mket (
    const std::vector< idx > & mask,
    const std::vector< idx > & dims ) [inline]
```

Multi-partite qudit ket.

See also

[ket](#) template<char... Bits> [qpp::operator "" \\_ket\(\)](#)

Constructs the multi-partite qudit ket  $|\text{mask}\rangle$ , where *mask* is a `std::vector` of non-negative integers. Each element in *mask* has to be smaller than the corresponding element in *dims*.

Parameters

<i>mask</i>	<code>std::vector</code> of non-negative integers
<i>dims</i>	Dimensions of the multi-partite system

Returns

Multi-partite qudit state vector, as a complex dynamic column vector

#### 6.1.3.94 `mket()` [2/2]

```
ket qpp::mket (
    const std::vector< idx > & mask,
    idx d = 2 ) [inline]
```

Multi-partite qudit ket.

See also

[ket](#) template<char... Bits> [qpp::operator "" \\_ket\(\)](#)

Constructs the multi-partite qudit ket  $|\text{mask}\rangle$ , all subsystem having equal dimension *d*. *mask* is a `std::vector` of non-negative integers, and each element in *mask* has to be strictly smaller than *d*.

Parameters

<i>mask</i>	<code>std::vector</code> of non-negative integers
<i>d</i>	Subsystem dimensions

Returns

Multi-partite qudit state vector, as a complex dynamic column vector

#### 6.1.3.95 `modinv()`

```
bigint qpp::modinv (
    bigint a,
    bigint p ) [inline]
```

Modular inverse of *a* mod *p*.

See also

[qpp::egcd\(\)](#)

Note

$a$  and  $p$  must be co-prime

Parameters

$a$	Non-negative integer
$p$	Non-negative integer

Returns

Modular inverse  $a^{-1} \bmod p$

#### 6.1.3.96 modmul()

```
bigint qpp::modmul (
    bigint a,
    bigint b,
    bigint p ) [inline]
```

Modular multiplication without overflow.

Computes  $ab \bmod p$  without overflow

Parameters

$a$	Integer
$b$	Integer
$p$	Positive integer

Returns

$ab \bmod p$  avoiding overflow

#### 6.1.3.97 modpow()

```
bigint qpp::modpow (
    bigint a,
    bigint n,
    bigint p ) [inline]
```

Fast integer power modulo  $p$  based on the SQUARE-AND-MULTIPLY algorithm.

**Note**

Uses `qpp::modmul()` that avoids overflows

Computes  $a^n \bmod p$

**Parameters**

$a$	Non-negative integer
$n$	Non-negative integer
$p$	Strictly positive integer

**Returns**

$a^n \bmod p$

**6.1.3.98 mprj()** [1/2]

```
cmat qpp::mprj (
    const std::vector< idx > & mask,
    const std::vector< idx > & dims ) [inline]
```

Projector onto multi-partite qudit ket.

**See also**

`cmat` `template<char... Bits> qpp::operator "" _prj()`

Constructs the projector onto the multi-partite qudit ket  $|\text{mask}\rangle$ , where *mask* is a `std::vector` of non-negative integers. Each element in *mask* has to be smaller than the corresponding element in *dims*.

**Parameters**

<i>mask</i>	std::vector of non-negative integers
<i>dims</i>	Dimensions of the multi-partite system

**Returns**

Projector onto multi-partite qudit state vector, as a complex dynamic matrix

**6.1.3.99 mprj()** [2/2]

```
cmat qpp::mprj (
    const std::vector< idx > & mask,
    idx d = 2 ) [inline]
```

Projector onto multi-partite qudit ket.

See also

[cmat](#) template<char... Bits> [qpp::operator "" \\_prj\(\)](#)

Constructs the projector onto the multi-partite qudit ket  $|\text{mask}\rangle$ , all subsystem having equal dimension  $d$ .  $\text{mask}$  is a `std::vector` of non-negative integers, and each element in  $\text{mask}$  has to be strictly smaller than  $d$ .

Parameters

<i>mask</i>	std::vector of non-negative integers
<i>d</i>	Subsystem dimensions

Returns

Projector onto multi-partite qudit state vector, as a complex dynamic matrix

#### 6.1.3.100 multiidx2n()

```
idx qpp::multiidx2n (
    const std::vector< idx > & midx,
    const std::vector< idx > & dims ) [inline]
```

Multi-index to non-negative integer index.

See also

[qpp::n2multiidx\(\)](#)

Uses standard lexicographical order, i.e. 00...0, 00...1 etc.

Parameters

<i>midx</i>	Multi-index
<i>dims</i>	Dimensions of the multi-partite system

Returns

Non-negative integer index

#### 6.1.3.101 n2multiidx()

```
std::vector<idx> qpp::n2multiidx (
    idx n,
    const std::vector< idx > & dims ) [inline]
```

Non-negative integer index to multi-index.

See also

[qpp::multiidx2n\(\)](#)

Uses standard lexicographical order, i.e. 00...0, 00...1 etc.

Parameters

<i>n</i>	Non-negative integer index
<i>dims</i>	Dimensions of the multi-partite system

Returns

Multi-index of the same size as *dims*

#### 6.1.3.102 negativity() [1/2]

```
template<typename Derived >
double qpp::negativity (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< idx > & dims )
```

Negativity of the bi-partite mixed state *A*.

Parameters

<i>A</i>	Eigen expression
<i>dims</i>	Dimensions of the bi-partite system

Returns

Negativity

#### 6.1.3.103 negativity() [2/2]

```
template<typename Derived >
double qpp::negativity (
    const Eigen::MatrixBase< Derived > & A,
    idx d = 2 )
```

Negativity of the bi-partite mixed state *A*.

Parameters

<i>A</i>	Eigen expression
<i>d</i>	Subsystem dimensions

**Returns**

Negativity

**6.1.3.104 norm()**

```
template<typename Derived >
double qpp::norm (
    const Eigen::MatrixBase< Derived > & A )
```

Frobenius norm.

**Parameters**

$A$	Eigen expression
-----	------------------

**Returns**Frobenius norm of  $A$ **6.1.3.105 omega()**

```
cplx qpp::omega (
    idx D ) [inline]
```

D-th root of unity.

**Parameters**

$D$	Non-negative integer
-----	----------------------

**Returns**D-th root of unity  $\exp(2\pi i/D)$ **6.1.3.106 operator""\_i()**

```
constexpr cplx qpp::operator"" _i (
    long double x ) [inline], [noexcept]
```

User-defined literal for complex  $i = \sqrt{-1}$  (real overload)

Example:

```
cplx z = 4.5_i; // type of z is std::complex<double>
```



## 6.1.3.107 powm()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::powm (
    const Eigen::MatrixBase< Derived > & A,
    idx n )
```

Fast matrix power based on the SQUARE-AND-MULTIPLY algorithm.

See also

[qpp::spectralpowm\(\)](#)

Explicitly multiplies the matrix  $A$  with itself  $n$  times. By convention  $A^0 = I$ .

Parameters

$A$	Eigen expression
$n$	Non-negative integer

Returns

Matrix power  $A^n$ , as a dynamic matrix over the same scalar field as  $A$

## 6.1.3.108 prj()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::prj (
    const Eigen::MatrixBase< Derived > & A )
```

Projector.

Normalized projector onto state vector

Parameters

$A$	Eigen expression
-----	------------------

Returns

Projector onto the state vector  $A$ , or the matrix *Zero* if  $A$  has norm zero (i.e. smaller than [qpp::eps](#)), as a dynamic matrix over the same scalar field as  $A$

**6.1.3.109 prod()** [1/3]

```
template<typename Derived >
Derived::Scalar qpp::prod (
    const Eigen::MatrixBase< Derived > & A )
```

Element-wise product of *A*.

**Parameters**

<i>A</i>	Eigen expression
----------	------------------

**Returns**

Element-wise product of *A*, as a scalar over the same scalar field as *A*

**6.1.3.110 prod()** [2/3]

```
template<typename InputIterator >
std::iterator_traits<InputIterator>::value_type qpp::prod (
    InputIterator first,
    InputIterator last )
```

Element-wise product of an STL-like range.

**Parameters**

<i>first</i>	Iterator to the first element of the range
<i>last</i>	Iterator to the last element of the range

**Returns**

Element-wise product of the range, as a scalar over the same scalar field as the range

**6.1.3.111 prod()** [3/3]

```
template<typename Container >
Container::value_type qpp::prod (
    const Container & c,
    typename std::enable_if< is\_iterable< Container >::value >::type * = nullptr )
```

Element-wise product of the elements of an STL-like container.

**Parameters**

<i>c</i>	STL-like container
----------	--------------------

**Returns**

Element-wise product of the elements of the container, as a scalar over the same scalar field as the container

**6.1.3.112 ptrace()** [1/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::ptrace (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< idx > & subsys,
    const std::vector< idx > & dims )
```

Partial trace.

**See also**

[qpp::ptrace1\(\)](#), [qpp::ptrace2\(\)](#)

Partial trace of the multi-partite state vector or density matrix over a list of subsystems

**Parameters**

<i>A</i>	Eigen expression
<i>subsys</i>	Subsystem indexes
<i>dims</i>	Dimensions of the multi-partite system

**Returns**

Partial trace  $Tr_{subsys}(\cdot)$  over the subsystems *subsys* in a multi-partite system, as a dynamic matrix over the same scalar field as *A*

**6.1.3.113 ptrace()** [2/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::ptrace (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< idx > & subsys,
    idx d = 2 )
```

Partial trace.

**See also**

[qpp::ptrace1\(\)](#), [qpp::ptrace2\(\)](#)

Partial trace of the multi-partite state vector or density matrix over a list of subsystems

## Parameters

$A$	Eigen expression
$subsys$	Subsystem indexes
$d$	Subsystem dimensions

## Returns

Partial trace  $Tr_{subsys}(\cdot)$  over the subsystems  $subsys$  in a multi-partite system, as a dynamic matrix over the same scalar field as  $A$

6.1.3.114 `ptrace1()` [1/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::ptrace1 (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< idx > & dims )
```

Partial trace.

## See also

[qpp::ptrace2\(\)](#)

Partial trace over the first subsystem of bi-partite state vector or density matrix

## Parameters

$A$	Eigen expression
$dims$	Dimensions of the bi-partite system

## Returns

Partial trace  $Tr_A(\cdot)$  over the first subsystem  $A$  in a bi-partite system  $A \otimes B$ , as a dynamic matrix over the same scalar field as  $A$

6.1.3.115 `ptrace1()` [2/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::ptrace1 (
    const Eigen::MatrixBase< Derived > & A,
    idx d = 2 )
```

Partial trace.

See also

[qpp::ptrace2\(\)](#)

Partial trace over the first subsystem of bi-partite state vector or density matrix

## Parameters

$A$	Eigen expression
$d$	Subsystem dimensions

## Returns

Partial trace  $Tr_A(\cdot)$  over the first subsystem  $A$  in a bi-partite system  $A \otimes B$ , as a dynamic matrix over the same scalar field as  $A$

6.1.3.116 `ptrace2()` [1/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::ptrace2 (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< idx > & dims )
```

Partial trace.

## See also

[qpp::ptrace1\(\)](#)

Partial trace over the second subsystem of bi-partite state vector or density matrix

## Parameters

$A$	Eigen expression
$dims$	Dimensions of the bi-partite system

## Returns

Partial trace  $Tr_B(\cdot)$  over the second subsystem  $B$  in a bi-partite system  $A \otimes B$ , as a dynamic matrix over the same scalar field as  $A$

6.1.3.117 `ptrace2()` [2/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::ptrace2 (
    const Eigen::MatrixBase< Derived > & A,
    idx d = 2 )
```

Partial trace.

## See also

[qpp::ptrace1\(\)](#)

Partial trace over the second subsystem of bi-partite state vector or density matrix

## Parameters

$A$	Eigen expression
$d$	Subsystem dimensions

## Returns

Partial trace  $Tr_B(\cdot)$  over the second subsystem  $B$  in a bi-partite system  $A \otimes B$ , as a dynamic matrix over the same scalar field as  $A$

## 6.1.3.118 ptranspose() [1/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::pttranspose (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< idx > & subsys,
    const std::vector< idx > & dims )
```

Partial transpose.

Partial transpose of the multi-partite state vector or density matrix over a list of subsystems

## Parameters

$A$	Eigen expression
$subsys$	Subsystem indexes
$dims$	Dimensions of the multi-partite system

## Returns

Partial transpose  $(\cdot)^{T_{subsys}}$  over the subsystems  $subsys$  in a multi-partite system, as a dynamic matrix over the same scalar field as  $A$

## 6.1.3.119 ptranspose() [2/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::pttranspose (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< idx > & subsys,
    idx d = 2 )
```

Partial transpose.

Partial transpose of the multi-partite state vector or density matrix over a list of subsystems

## Parameters

$A$	Eigen expression
$subsys$	Subsystem indexes
$d$	Subsystem dimensions

## Returns

Partial transpose  $(\cdot)^{T_{subsys}}$  over the subsystems  $subsys$  in a multi-partite system, as a dynamic matrix over the same scalar field as  $A$

6.1.3.120 `qmutualinfo()` [1/2]

```
template<typename Derived >
double qpp::qmutualinfo (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< idx > & subsysA,
    const std::vector< idx > & subsysB,
    const std::vector< idx > & dims )
```

Quantum mutual information between 2 subsystems of a composite system.

## Parameters

$A$	Eigen expression
$subsysA$	Indexes of the first subsystem
$subsysB$	Indexes of the second subsystem
$dims$	Dimensions of the multi-partite system

## Returns

Mutual information between the 2 subsystems

6.1.3.121 `qmutualinfo()` [2/2]

```
template<typename Derived >
double qpp::qmutualinfo (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< idx > & subsysA,
    const std::vector< idx > & subsysB,
    idx d = 2 )
```

Quantum mutual information between 2 subsystems of a composite system.



## Parameters

<i>A</i>	Eigen expression
<i>subsysA</i>	Indexes of the first subsystem
<i>subsysB</i>	Indexes of the second subsystem
<i>d</i>	Subsystem dimensions

## Returns

Mutual information between the 2 subsystems

6.1.3.122 `rand()` [1/5]

```
double qpp::rand (
    double a,
    double b ) [inline]
```

Generates a random real number uniformly distributed in the interval [a, b)

## Parameters

<i>a</i>	Beginning of the interval, belongs to it
<i>b</i>	End of the interval, does not belong to it

## Returns

Random real number (double) uniformly distributed in the interval [a, b)

6.1.3.123 `rand()` [2/5]

```
bigint qpp::rand (
    bigint a,
    bigint b ) [inline]
```

Generates a random big integer uniformly distributed in the interval [a, b].

## Note

To avoid ambiguity with double `qpp::rand(double, double)` cast at least one of the arguments to `qpp::bigint`

## Parameters

<i>a</i>	Beginning of the interval, belongs to it
<i>b</i>	End of the interval, belongs to it

**Returns**

Random big integer uniformly distributed in the interval [a, b]

**6.1.3.124 rand()** [3/5]

```
template<typename Derived >
Derived qpp::rand (
    idx rows,
    idx cols,
    double a = 0,
    double b = 1 )
```

Generates a random matrix with entries uniformly distributed in the interval [a, b)

If complex, then both real and imaginary parts are uniformly distributed in [a, b)

This is the generic version that always throws `qpp::Exception::Type::UNDEFINED_TYPE`. It is specialized only for `qpp::dmat` and `qpp::cmat`

**6.1.3.125 rand()** [4/5]

```
template<>
dmat qpp::rand (
    idx rows,
    idx cols,
    double a,
    double b ) [inline]
```

Generates a random real matrix with entries uniformly distributed in the interval [a, b), specialization for double matrices (`qpp::dmat`)

The template parameter cannot be automatically deduced and must be explicitly provided

Example:

```
// generates a 3 x 3 random Eigen::MatrixXd,
// with entries uniformly distributed in [-1,1)
dmat mat = rand<dmat>(3, 3, -1, 1);
```

**Parameters**

<i>rows</i>	Number of rows of the random generated matrix
<i>cols</i>	Number of columns of the random generated matrix
<i>a</i>	Beginning of the interval, belongs to it
<i>b</i>	End of the interval, does not belong to it

**Returns**

Random real matrix

**6.1.3.126 rand()** [5/5]

```
template<>
cmat qpp::rand (
    idx rows,
    idx cols,
    double a,
    double b ) [inline]
```

Generates a random complex matrix with entries (both real and imaginary) uniformly distributed in the interval [a, b), specialization for complex matrices ([qpp::cmat](#))

The template parameter cannot be automatically deduced and must be explicitly provided

Example:

```
// generates a 3 x 3 random Eigen::MatrixXcd,
// with entries (both real and imaginary) uniformly distributed in [-1,1)
cmat mat = rand<cmat>(3, 3, -1, 1);
```

**Parameters**

<i>rows</i>	Number of rows of the random generated matrix
<i>cols</i>	Number of columns of the random generated matrix
<i>a</i>	Beginning of the interval, belongs to it
<i>b</i>	End of the interval, does not belong to it

**Returns**

Random complex matrix

**6.1.3.127 randH()**

```
cmat qpp::randH (
    idx D = 2 ) [inline]
```

Generates a random Hermitian matrix.

**Parameters**

<i>D</i>	Dimension of the Hilbert space
----------	--------------------------------

**Returns**

Random Hermitian matrix

**6.1.3.128 randidx()**

```
idx qpp::randidx (
    idx a = std::numeric_limits<idx>::min(),
    idx b = std::numeric_limits<idx>::max() ) [inline]
```

Generates a random index (idx) uniformly distributed in the interval [a, b].

**Parameters**

<i>a</i>	Beginning of the interval, belongs to it
<i>b</i>	End of the interval, belongs to it

**Returns**

Random index (idx) uniformly distributed in the interval [a, b]

**6.1.3.129 randket()**

```
ket qpp::randket (
    idx D = 2 ) [inline]
```

Generates a random normalized ket (pure state vector)

**Parameters**

<i>D</i>	Dimension of the Hilbert space
----------	--------------------------------

**Returns**

Random normalized ket

**6.1.3.130 randkraus()**

```
std::vector<cmat> qpp::randkraus (
    idx N,
    idx D = 2 ) [inline]
```

Generates a set of random Kraus operators.

**Note**

The set of Kraus operators satisfy the closure condition  $\sum_i K_i^\dagger K_i = I$

**Parameters**

<i>N</i>	Number of Kraus operators
<i>D</i>	Dimension of the Hilbert space

**Returns**

Set of *N* Kraus operators satisfying the closure condition

**6.1.3.131 randn()** [1/4]

```
template<typename Derived >
Derived qpp::randn (
    idx rows,
    idx cols,
    double mean = 0,
    double sigma = 1 )
```

Generates a random matrix with entries normally distributed in N(mean, sigma)

If complex, then both real and imaginary parts are normally distributed in N(mean, sigma)

This is the generic version that always throws `qpp::Exception::Type::UNDEFINED_TYPE`. It is specialized only for `qpp::dmat` and `qpp::cmat`

**6.1.3.132 randn()** [2/4]

```
template<>
dmat qpp::randn (
    idx rows,
    idx cols,
    double mean,
    double sigma ) [inline]
```

Generates a random real matrix with entries normally distributed in N(mean, sigma), specialization for double matrices (`qpp::dmat`)

The template parameter cannot be automatically deduced and must be explicitly provided

Example:

```
// generates a 3 x 3 random Eigen::MatrixXd,
// with entries normally distributed in N(0,2)
dmat mat = randn<dmat>(3, 3, 0, 2);
```

**Parameters**

<i>rows</i>	Number of rows of the random generated matrix
<i>cols</i>	Number of columns of the random generated matrix
<i>mean</i>	Mean
<i>sigma</i>	Standard deviation

**Returns**

Random real matrix

**6.1.3.133 randn()** [3/4]

```
template<>
cmat qpp::randn (
    idx rows,
    idx cols,
    double mean,
    double sigma ) [inline]
```

Generates a random complex matrix with entries (both real and imaginary) normally distributed in  $N(\text{mean}, \text{sigma})$ , specialization for complex matrices (`qpp::cmat`)

The template parameter cannot be automatically deduced and must be explicitly provided

Example:

```
// generates a 3 x 3 random Eigen::MatrixXcd,
// with entries (both real and imaginary) normally distributed in N(0,2)
cmat mat = randn<cmat>(3, 3, 0, 2);
```

**Parameters**

<i>rows</i>	Number of rows of the random generated matrix
<i>cols</i>	Number of columns of the random generated matrix
<i>mean</i>	Mean
<i>sigma</i>	Standard deviation

**Returns**

Random complex matrix

**6.1.3.134 randn()** [4/4]

```
double qpp::randn (
    double mean = 0,
    double sigma = 1 ) [inline]
```

Generates a random real number (double) normally distributed in  $N(\text{mean}, \text{sigma})$

#### Parameters

<i>mean</i>	Mean
<i>sigma</i>	Standard deviation

#### Returns

Random real number normally distributed in  $N(\text{mean}, \text{sigma})$

#### 6.1.3.135 randperm()

```
std::vector<idx> qpp::randperm (
    idx N ) [inline]
```

Generates a random uniformly distributed permutation.

Uses Knuth shuffle method (as implemented by `std::shuffle`), so that all permutations are equally probable

#### Parameters

<i>N</i>	Size of the permutation
----------	-------------------------

#### Returns

Random permutation of size *N*

#### 6.1.3.136 randprime()

```
bigint qpp::randprime (
    bigint a,
    bigint b,
    idx N = 1000 ) [inline]
```

Generates a random big prime uniformly distributed in the interval  $[a, b]$ .

#### Parameters

<i>a</i>	Beginning of the interval, belongs to it
<i>b</i>	End of the interval, belongs to it
<i>N</i>	Maximum number of candidates

**Returns**

Random big integer uniformly distributed in the interval [a, b]

**6.1.3.137 randprob()**

```
std::vector<double> qpp::randprob (
    idx N ) [inline]
```

Generates a random probability vector uniformly distributed over the probability simplex.

**Parameters**

<i>N</i>	Size of the probability vector
----------	--------------------------------

**Returns**

Random probability vector

**6.1.3.138 randrho()**

```
cmat qpp::randrho (
    idx D = 2 ) [inline]
```

Generates a random density matrix.

**Parameters**

<i>D</i>	Dimension of the Hilbert space
----------	--------------------------------

**Returns**

Random density matrix

**6.1.3.139 randU()**

```
cmat qpp::randU (
    idx D = 2 ) [inline]
```

Generates a random unitary matrix.



## Parameters

<i>D</i>	Dimension of the Hilbert space
----------	--------------------------------

## Returns

Random unitary

## 6.1.3.140 randV()

```
cmat qpp::randV (
    idx Din,
    idx Dout ) [inline]
```

Generates a random isometry matrix.

## Parameters

<i>Din</i>	Size of the input Hilbert space
<i>Dout</i>	Size of the output Hilbert space

## Returns

Random isometry matrix

## 6.1.3.141 renyi() [1/2]

```
template<typename Derived >
double qpp::renyi (
    const Eigen::MatrixBase< Derived > & A,
    double alpha )
```

Renyi-  $\alpha$  entropy of the density matrix  $A$ , for  $\alpha \geq 0$ .

## Note

When  $\alpha \rightarrow 1$  the Renyi entropy converges to the von-Neumann entropy, with the logarithm in base 2

## Parameters

<i>A</i>	Eigen expression
<i>alpha</i>	Non-negative real number, use <a href="#">qpp::infy</a> for $\alpha = \infty$

**Returns**

Renyi-  $\alpha$  entropy, with the logarithm in base 2

**6.1.3.142** `renyi()` [2/2]

```
double qpp::renyi (
    const std::vector< double > & prob,
    double alpha ) [inline]
```

Renyi-  $\alpha$  entropy of the probability distribution *prob*, for  $\alpha \geq 0$ .

**Note**

When  $\alpha \rightarrow 1$  the Renyi entropy converges to the Shannon entropy, with the logarithm in base 2

**Parameters**

<i>prob</i>	Real probability vector
<i>alpha</i>	Non-negative real number, use <code>qpp::infy</code> for $\alpha = \infty$

**Returns**

Renyi-  $\alpha$  entropy, with the logarithm in base 2

**6.1.3.143** `reshape()`

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::reshape (
    const Eigen::MatrixBase< Derived > & A,
    idx rows,
    idx cols )
```

Reshape.

Uses column-major order when reshaping (same as MATLAB)

**Parameters**

<i>A</i>	Eigen expression
<i>rows</i>	Number of rows of the reshaped matrix
<i>cols</i>	Number of columns of the reshaped matrix

**Returns**

Reshaped matrix with *rows* rows and *cols* columns, as a dynamic matrix over the same scalar field as *A*

**6.1.3.144 rho2bloch()**

```
template<typename Derived >
std::vector<double> qpp::rho2bloch (
    const Eigen::MatrixBase< Derived > & A )
```

Computes the 3-dimensional real Bloch vector corresponding to the qubit density matrix *A*.

**See also**

[qpp::bloch2rho\(\)](#)

**Note**

It is implicitly assumed that the density matrix is Hermitian

**Parameters**

<i>A</i>	Eigen expression
----------	------------------

**Returns**

3-dimensional Bloch vector

**6.1.3.145 rho2pure()**

```
template<typename Derived >
dyn_col_vect<typename Derived::Scalar> qpp::rho2pure (
    const Eigen::MatrixBase< Derived > & A )
```

Finds the pure state representation of a matrix proportional to a projector onto a pure state.

**Note**

No purity check is done, the input state *A* must have rank one, otherwise the function returns the first non-zero eigenvector of *A*

**Parameters**

<i>A</i>	Eigen expression, assumed to be proportional to a projector onto a pure state, i.e. <i>A</i> is assumed to have rank one
----------	--

**Returns**

The unique non-zero eigenvector of  $A$  (up to a phase), as a dynamic column vector over the same scalar field as  $A$

**6.1.3.146 save()**

```
template<typename Derived >
void qpp::save (
    const Eigen::MatrixBase< Derived > & A,
    const std::string & fname )
```

Saves Eigen expression to a binary file (internal format) in double precision.

**See also**

[qpp::load\(\)](#)

**Parameters**

$A$	Eigen expression
<i>fname</i>	Output file name

**6.1.3.147 saveMATLAB()** [1/2]

```
template<typename Derived >
std::enable_if< std::is_same<typename Derived::Scalar, cplx>::value>::type qpp::saveMATLAB (
    const Eigen::MatrixBase< Derived > & A,
    const std::string & mat_file,
    const std::string & var_name,
    const std::string & mode )
```

Saves a complex Eigen dynamic matrix to a MATLAB .mat file,.

**See also**

[qpp::loadMATLAB\(\)](#)

**Template Parameters**

<i>Complex</i>	Eigen type
----------------	------------

**Parameters**

$A$	Eigen expression over the complex field
-----	---

## Parameters

<i>mat_file</i>	MATALB .mat file
<i>var_name</i>	Variable name in the .mat file representing the matrix to be saved
<i>mode</i>	Saving mode (append, overwrite etc.), see MATLAB <i>matOpen()</i> documentation for details

## 6.1.3.148 saveMATLAB() [2/2]

```
template<typename Derived >
std::enable_if< !std::is_same<typename Derived::Scalar, cplx>::value>::type qpp::saveMATLAB (
    const Eigen::MatrixBase< Derived > & A,
    const std::string & mat_file,
    const std::string & var_name,
    const std::string & mode )
```

Saves a non-complex Eigen dynamic matrix to a MATLAB .mat file,.

## See also

[qpp::loadMATLAB\(\)](#)

## Template Parameters

<i>Npn-complex</i>	Eigen type
--------------------	------------

## Parameters

<i>A</i>	Non-complex Eigen expression
<i>mat_file</i>	MATALB .mat file
<i>var_name</i>	Variable name in the .mat file representing the matrix to be saved
<i>mode</i>	Saving mode (append, overwrite etc.), see MATLAB <i>matOpen()</i> documentation for details

## 6.1.3.149 Schatten()

```
template<typename Derived >
double qpp::schatten (
    const Eigen::MatrixBase< Derived > & A,
    double p )
```

Schatten matrix norm.

## Parameters

<i>A</i>	Eigen expression
<i>p</i>	Real number, greater or equal to 1, use <a href="#">qpp::infy</a> for $p = \infty$

**Returns**

Schatten- $p$  matrix norm of  $A$

**6.1.3.150 schmidtA()** [1/2]

```
template<typename Derived >
cmat qpp::schmidtA (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< idx > & dims )
```

Schmidt basis on Alice side.

**Parameters**

$A$	Eigen expression
$dims$	Dimensions of the bi-partite system

**Returns**

Unitary matrix  $U$  whose columns represent the Schmidt basis vectors on Alice side.

**6.1.3.151 schmidtA()** [2/2]

```
template<typename Derived >
cmat qpp::schmidtA (
    const Eigen::MatrixBase< Derived > & A,
    idx d = 2 )
```

Schmidt basis on Alice side.

**Parameters**

$A$	Eigen expression
$d$	Subsystem dimensions

**Returns**

Unitary matrix  $U$  whose columns represent the Schmidt basis vectors on Alice side.

**6.1.3.152 schmidtB()** [1/2]

```
template<typename Derived >
cmat qpp::schmidtB (
```

```
const Eigen::MatrixBase< Derived > & A,
const std::vector< idx > & dims )
```

Schmidt basis on Bob side.

#### Parameters

<i>A</i>	Eigen expression
<i>dims</i>	Dimensions of the bi-partite system

#### Returns

Unitary matrix  $V$  whose columns represent the Schmidt basis vectors on Bob side.

#### 6.1.3.153 schmidtB() [2/2]

```
template<typename Derived >
cmat qpp::schmidtB (
    const Eigen::MatrixBase< Derived > & A,
    idx d = 2 )
```

Schmidt basis on Bob side.

#### Parameters

<i>A</i>	Eigen expression
<i>d</i>	Subsystem dimensions

#### Returns

Unitary matrix  $V$  whose columns represent the Schmidt basis vectors on Bob side.

#### 6.1.3.154 schmidtcoeffs() [1/2]

```
template<typename Derived >
dyn_col_vect<double> qpp::schmidtcoeffs (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< idx > & dims )
```

Schmidt coefficients of the bi-partite pure state  $A$ .

#### Note

The sum of the squares of the Schmidt coefficients equals 1

#### See also

[qpp::schmidtprobs\(\)](#)

**Parameters**

<i>A</i>	Eigen expression
<i>dims</i>	Dimensions of the bi-partite system

**Returns**

Schmidt coefficients of *A*, ordered in decreasing order, as a real dynamic column vector

**6.1.3.155 schmidtcoeffs()** [2/2]

```
template<typename Derived >
dyn_col_vect<double> qpp::schmidtcoeffs (
    const Eigen::MatrixBase< Derived > & A,
    idx d = 2 )
```

Schmidt coefficients of the bi-partite pure state *A*.

**Note**

The sum of the squares of the Schmidt coefficients equals 1

**See also**

[qpp::schmidtprobs\(\)](#)

**Parameters**

<i>A</i>	Eigen expression
<i>d</i>	Subsystem dimensions

**Returns**

Schmidt coefficients of *A*, ordered in decreasing order, as a real dynamic column vector

**6.1.3.156 schmidtprobs()** [1/2]

```
template<typename Derived >
std::vector<double> qpp::schmidtprobs (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< idx > & dims )
```

Schmidt probabilities of the bi-partite pure state *A*.

Defined as the squares of the Schmidt coefficients. The sum of the Schmidt probabilities equals 1.



See also

[qpp::schmidtcoeffs\(\)](#)

#### Parameters

<i>A</i>	Eigen expression
<i>dims</i>	Dimensions of the bi-partite system

#### Returns

Real vector consisting of the Schmidt probabilities of *A*, ordered in decreasing order

#### 6.1.3.157 `schmidtprobs()` [2/2]

```
template<typename Derived >
std::vector<double> qpp::schmidtprobs (
    const Eigen::MatrixBase< Derived > & A,
    idx d = 2 )
```

Schmidt probabilities of the bi-partite pure state *A*.

Defined as the squares of the Schmidt coefficients. The sum of the Schmidt probabilities equals 1.

See also

[qpp::schmidtcoeffs\(\)](#)

#### Parameters

<i>A</i>	Eigen expression
<i>d</i>	Subsystem dimensions

#### Returns

Real vector consisting of the Schmidt probabilities of *A*, ordered in decreasing order

#### 6.1.3.158 `sigma()`

```
template<typename Container >
double qpp::sigma (
    const std::vector< double > & prob,
    const Container & X,
    typename std::enable_if< is_iterable< Container >::value >::type * = nullptr )
```

Standard deviation.

## Parameters

<i>prob</i>	Real probability vector representing the probability distribution of $X$
$X$	Real random variable values represented by an STL-like container

## Returns

Standard deviation of  $X$

6.1.3.159 `sinm()`

```
template<typename Derived >
cmat qpp::sinm (
    const Eigen::MatrixBase< Derived > & A )
```

Matrix sin.

## Parameters

$A$	Eigen expression
-----	------------------

## Returns

Matrix sine of  $A$

6.1.3.160 `spectralpowm()`

```
template<typename Derived >
cmat qpp::spectralpowm (
    const Eigen::MatrixBase< Derived > & A,
    const cplx z )
```

Matrix power.

## See also

[`qpp::powm\(\)`](#)

Uses the spectral decomposition of  $A$  to compute the matrix power. By convention  $A^0 = I$ .

## Parameters

$A$	Eigen expression
$z$	Complex number

**Returns**

Matrix power  $A^z$

**6.1.3.161 sqrtm()**

```
template<typename Derived >
cmat qpp::sqrtm (
    const Eigen::MatrixBase< Derived > & A )
```

Matrix square root.

**Parameters**

$A$	Eigen expression
-----	------------------

**Returns**

Matrix square root of  $A$

**6.1.3.162 sum()** [1/3]

```
template<typename Derived >
Derived::Scalar qpp::sum (
    const Eigen::MatrixBase< Derived > & A )
```

Element-wise sum of  $A$ .

**Parameters**

$A$	Eigen expression
-----	------------------

**Returns**

Element-wise sum of  $A$ , as a scalar over the same scalar field as  $A$

**6.1.3.163 sum()** [2/3]

```
template<typename InputIterator >
std::iterator_traits<InputIterator>::value_type qpp::sum (
    InputIterator first,
    InputIterator last )
```

Element-wise sum of an STL-like range.

**Parameters**

<i>first</i>	Iterator to the first element of the range
<i>last</i>	Iterator to the last element of the range

**Returns**

Element-wise sum of the range, as a scalar over the same scalar field as the range

**6.1.3.164 `sum()`** [3/3]

```
template<typename Container >
Container::value_type qpp::sum (
    const Container & c,
    typename std::enable_if< is\_iterable< Container >::value >::type * = nullptr )
```

Element-wise sum of the elements of an STL-like container.

**Parameters**

<i>c</i>	STL-like container
----------	--------------------

**Returns**

Element-wise sum of the elements of the container, as a scalar over the same scalar field as the container

**6.1.3.165 `super2choi()`**

```
cmat qpp::super2choi (
    const cmat & A ) [inline]
```

Converts superoperator matrix to Choi matrix.

**See also**

[qpp::choi2super\(\)](#)

**Parameters**

<i>A</i>	Superoperator matrix
----------	----------------------

**Returns**

Choi matrix

**6.1.3.166 svals()**

```
template<typename Derived >
dyn_col_vect<double> qpp::svals (
    const Eigen::MatrixBase< Derived > & A )
```

Singular values.

**Parameters**

<i>A</i>	Eigen expression
----------	------------------

**Returns**

Singular values of *A*, ordered in decreasing order, as a real dynamic column vector

**6.1.3.167 svd()**

```
template<typename Derived >
std::tuple<cmat, dyn_col_vect<double>, cmat> qpp::svd (
    const Eigen::MatrixBase< Derived > & A )
```

Full singular value decomposition.

**Parameters**

<i>A</i>	Eigen expression
----------	------------------

**Returns**

Tuple of: 1. Left singular vectors of *A*, as columns of a complex dynamic matrix, 2. Singular values of *A*, ordered in decreasing order, as a real dynamic column vector, and 3. Right singular vectors of *A*, as columns of a complex dynamic matrix

**6.1.3.168 svdU()**

```
template<typename Derived >
cmat qpp::svdU (
    const Eigen::MatrixBase< Derived > & A )
```

Left singular vectors.

## Parameters

$A$	Eigen expression
-----	------------------

## Returns

Complex dynamic matrix, whose columns are the left singular vectors of  $A$

6.1.3.169 `svdV()`

```
template<typename Derived >
cmat qpp::svdV (
    const Eigen::MatrixBase< Derived > & A )
```

Right singular vectors.

## Parameters

$A$	Eigen expression
-----	------------------

## Returns

Complex dynamic matrix, whose columns are the right singular vectors of  $A$

6.1.3.170 `syspermute()` [1/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::syspermute (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< idx > & perm,
    const std::vector< idx > & dims )
```

Subsystem permutation.

Permutes the subsystems of a state vector or density matrix. The qubit *perm*[*i*] is permuted to the location *i*.

## Parameters

$A$	Eigen expression
<i>perm</i>	Permutation
<i>dims</i>	Dimensions of the multi-partite system

**Returns**

Permuted system, as a dynamic matrix over the same scalar field as  $A$

**6.1.3.171 syspermute()** [2/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::syspermute (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< idx > & perm,
    idx d = 2 )
```

Subsystem permutation.

Permutes the subsystems of a state vector or density matrix. The qubit  $perm[i]$  is permuted to the location  $i$ .

**Parameters**

$A$	Eigen expression
$perm$	Permutation
$d$	Subsystem dimensions

**Returns**

Permuted system, as a dynamic matrix over the same scalar field as  $A$

**6.1.3.172 trace()**

```
template<typename Derived >
Derived::Scalar qpp::trace (
    const Eigen::MatrixBase< Derived > & A )
```

Trace.

**Parameters**

$A$	Eigen expression
-----	------------------

**Returns**

Trace of  $A$ , as a scalar over the same scalar field as  $A$

**6.1.3.173 transpose()**

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::transpose (
    const Eigen::MatrixBase< Derived > & A )
```

Transpose.

**Parameters**

<i>A</i>	Eigen expression
----------	------------------

**Returns**

Transpose of *A*, as a dynamic matrix over the same scalar field as *A*

**6.1.3.174 tsallis()** [1/2]

```
template<typename Derived >
double qpp::tsallis (
    const Eigen::MatrixBase< Derived > & A,
    double q )
```

Tsallis- *q* entropy of the density matrix *A*, for  $q \geq 0$ .

**Note**

When  $q \rightarrow 1$  the Tsallis entropy converges to the von-Neumann entropy, with the logarithm in base *e*

**Parameters**

<i>A</i>	Eigen expression
<i>q</i>	Non-negative real number

**Returns**

Tsallis- *q* entropy

**6.1.3.175 tsallis()** [2/2]

```
double qpp::tsallis (
    const std::vector< double > & prob,
    double q ) [inline]
```

Tsallis- *q* entropy of the probability distribution *prob*, for  $q \geq 0$ .



**Note**

When  $q \rightarrow 1$  the Tsallis entropy converges to the Shannon entropy, with the logarithm in base  $e$

**Parameters**

<i>prob</i>	Real probability vector
<i>q</i>	Non-negative real number

**Returns**

Tsallis-  $q$  entropy

**6.1.3.176 uniform()**

```
std::vector<double> qpp::uniform (
    idx N ) [inline]
```

Uniform probability distribution vector.

**Parameters**

<i>N</i>	Size of the alphabet
----------	----------------------

**Returns**

Real vector consisting of a uniform distribution of size  $N$

**6.1.3.177 var()**

```
template<typename Container >
double qpp::var (
    const std::vector< double > & prob,
    const Container & X,
    typename std::enable_if< is_iterable< Container >::value >::type * = nullptr )
```

Variance.

**Parameters**

<i>prob</i>	Real probability vector representing the probability distribution of $X$
$X$	Real random variable values represented by an STL-like container

**Returns**

Variance of  $X$

**6.1.3.178 x2contfrac()**

```
std::vector<int> qpp::x2contfrac (
    double x,
    idx N,
    idx cut = 1e5 ) [inline]
```

Simple continued fraction expansion.

**See also**

[qpp::contfrac2x\(\)](#)

**Parameters**

$x$	Real number
$N$	Maximum number of terms in the expansion
$cut$	Stop the expansion when the next term is greater than $cut$

**Returns**

Integer vector containing the simple continued fraction expansion of  $x$ . If there are  $M$  less than  $N$  terms in the expansion, a shorter vector with  $M$  components is returned.

**6.1.4 Variable Documentation****6.1.4.1 chop**

```
constexpr double qpp::chop = 1e-10
```

Used in [qpp::disp\(\)](#) for setting to zero numbers that have their absolute value smaller than [qpp::chop](#).

**6.1.4.2 ee**

```
constexpr double qpp::ee = 2.718281828459045235360287471352662497
```

Base of natural logarithm,  $e$ .

#### 6.1.4.3 eps

```
constexpr double qpp::eps = 1e-12
```

Used to decide whether a number or expression in double precision is zero or not.

Example:

```
if(std::abs(x) < qpp::eps) // x is zero
```

#### 6.1.4.4 infity

```
constexpr double qpp::infity = std::numeric_limits<double>::max()
```

Used to denote infinity in double precision.

#### 6.1.4.5 maxn

```
constexpr idx qpp::maxn = 64
```

Maximum number of allowed qubits/qudits (subsystems)

Used internally to allocate arrays on the stack (for performance reasons):

#### 6.1.4.6 pi

```
constexpr double qpp::pi = 3.141592653589793238462643383279502884
```

$\pi$

## 6.2 qpp::exception Namespace Reference

Quantum++ exception hierarchy namespace.

## Classes

- class [CustomException](#)  
*Custom exception.*
- class [DimsInvalid](#)  
*Invalid dimension(s) exception.*
- class [DimsMismatchCvector](#)  
*Dimension(s) mismatch column vector size exception.*
- class [DimsMismatchMatrix](#)  
*Dimension(s) mismatch matrix size exception.*
- class [DimsMismatchRvector](#)  
*Dimension(s) mismatch row vector size exception.*
- class [DimsMismatchVector](#)  
*Dimension(s) mismatch vector size exception.*
- class [DimsNotEqual](#)  
*Dimensions not equal exception.*
- class [Exception](#)  
*Base class for generating Quantum++ custom exceptions.*
- class [MatrixMismatchSubsys](#)  
*Matrix mismatch subsystems exception.*
- class [MatrixNotCvector](#)  
*Matrix is not a column vector exception.*
- class [MatrixNotRvector](#)  
*Matrix is not a row vector exception.*
- class [MatrixNotSquare](#)  
*Matrix is not square exception.*
- class [MatrixNotSquareNorCvector](#)  
*Matrix is not square nor column vector exception.*
- class [MatrixNotSquareNorRvector](#)  
*Matrix is not square nor row vector exception.*
- class [MatrixNotSquareNorVector](#)  
*Matrix is not square nor vector exception.*
- class [MatrixNotVector](#)  
*Matrix is not a vector exception.*
- class [NoCodeword](#)  
*Codeword does not exist exception.*
- class [NotBipartite](#)  
*Not bi-partite exception.*
- class [NotQubitCvector](#)  
*Column vector is not  $2 \times 1$  exception.*
- class [NotQubitMatrix](#)  
*Matrix is not  $2 \times 2$  exception.*
- class [NotQubitRvector](#)  
*Row vector is not  $1 \times 2$  exception.*
- class [NotQubitSubsys](#)  
*Subsystems are not qubits exception.*
- class [NotQubitVector](#)  
*Vector is not  $2 \times 1$  nor  $1 \times 2$  exception.*
- class [OutOfRange](#)  
*Parameter out of range exception.*
- class [PermInvalid](#)

- Invalid permutation exception.*
- class [PermMismatchDims](#)
  - Permutation mismatch dimensions exception.*
- class [SizeMismatch](#)
  - Size mismatch exception.*
- class [SubsysMismatchDims](#)
  - Subsystems mismatch dimensions exception.*
- class [TypeMismatch](#)
  - Type mismatch exception.*
- class [UndefinedType](#)
  - Not defined for this type exception.*
- class [Unknown](#)
  - Unknown exception.*
- class [ZeroSize](#)
  - Object has zero size exception.*

### 6.2.1 Detailed Description

Quantum++ exception hierarchy namespace.

## 6.3 qpp::experimental Namespace Reference

Experimental/test functions/classes, do not use or modify.

### 6.3.1 Detailed Description

Experimental/test functions/classes, do not use or modify.

## 6.4 qpp::internal Namespace Reference

Internal utility functions, do not use them directly or modify them.

### Classes

- struct [Display\\_Impl\\_](#)
- class [IOManipEigen](#)
- class [IOManipPointer](#)
- class [IOManipRange](#)
- class [Singleton](#)
  - [Singleton](#) policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern)*

## Functions

- void `n2multiidx` (`idx` n, `idx` numdims, const `idx` \*const dims, `idx` \*result) noexcept
- `idx multiidx2n` (const `idx` \*const midx, `idx` numdims, const `idx` \*const dims) noexcept
- template<typename Derived >  
bool `check_square_mat` (const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >  
bool `check_vector` (const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >  
bool `check_rvector` (const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >  
bool `check_cvector` (const Eigen::MatrixBase< Derived > &A)
- template<typename T >  
bool `check_nonzero_size` (const T &x) noexcept
- template<typename T1 , typename T2 >  
bool `check_matching_sizes` (const T1 &lhs, const T2 &rhs) noexcept
- bool `check_dims` (const std::vector< `idx` > &dims)
- template<typename Derived >  
bool `check_dims_match_mat` (const std::vector< `idx` > &dims, const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >  
bool `check_dims_match_cvect` (const std::vector< `idx` > &dims, const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >  
bool `check_dims_match_rvect` (const std::vector< `idx` > &dims, const Eigen::MatrixBase< Derived > &A)
- bool `check_eq_dims` (const std::vector< `idx` > &dims, `idx` dim) noexcept
- bool `check_subsys_match_dims` (const std::vector< `idx` > &subsys, const std::vector< `idx` > &dims)
- template<typename Derived >  
bool `check_qubit_matrix` (const Eigen::MatrixBase< Derived > &A) noexcept
- template<typename Derived >  
bool `check_qubit_cvector` (const Eigen::MatrixBase< Derived > &A) noexcept
- template<typename Derived >  
bool `check_qubit_rvector` (const Eigen::MatrixBase< Derived > &A) noexcept
- template<typename Derived >  
bool `check_qubit_vector` (const Eigen::MatrixBase< Derived > &A) noexcept
- bool `check_perm` (const std::vector< `idx` > &perm)
- template<typename Derived1 , typename Derived2 >  
`dyn_mat`< typename Derived1::Scalar > `kron2` (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)
- template<typename Derived1 , typename Derived2 >  
`dyn_mat`< typename Derived1::Scalar > `dirsum2` (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)
- template<typename T >  
void `variadic_vector_emplace` (std::vector< T > &)
- template<typename T , typename First , typename... Args>  
void `variadic_vector_emplace` (std::vector< T > &v, First &&first, Args &&... args)
- `idx get_num_subsys` (`idx` sz, `idx` d)
- `idx get_dim_subsys` (`idx` sz, `idx` N)

### 6.4.1 Detailed Description

Internal utility functions, do not use them directly or modify them.

### 6.4.2 Function Documentation

#### 6.4.2.1 check\_cvector()

```
template<typename Derived >
bool qpp::internal::check_cvector (
    const Eigen::MatrixBase< Derived > & A )
```

#### 6.4.2.2 check\_dims()

```
bool qpp::internal::check_dims (
    const std::vector< idx > & dims ) [inline]
```

#### 6.4.2.3 check\_dims\_match\_cvect()

```
template<typename Derived >
bool qpp::internal::check_dims_match_cvect (
    const std::vector< idx > & dims,
    const Eigen::MatrixBase< Derived > & A )
```

#### 6.4.2.4 check\_dims\_match\_mat()

```
template<typename Derived >
bool qpp::internal::check_dims_match_mat (
    const std::vector< idx > & dims,
    const Eigen::MatrixBase< Derived > & A )
```

#### 6.4.2.5 check\_dims\_match\_rvect()

```
template<typename Derived >
bool qpp::internal::check_dims_match_rvect (
    const std::vector< idx > & dims,
    const Eigen::MatrixBase< Derived > & A )
```

#### 6.4.2.6 check\_eq\_dims()

```
bool qpp::internal::check_eq_dims (
    const std::vector< idx > & dims,
    idx dim ) [inline], [noexcept]
```

**6.4.2.7 check\_matching\_sizes()**

```
template<typename T1 , typename T2 >
bool qpp::internal::check_matching_sizes (
    const T1 & lhs,
    const T2 & rhs ) [noexcept]
```

**6.4.2.8 check\_nonzero\_size()**

```
template<typename T >
bool qpp::internal::check_nonzero_size (
    const T & x ) [noexcept]
```

**6.4.2.9 check\_perm()**

```
bool qpp::internal::check_perm (
    const std::vector< idx > & perm ) [inline]
```

**6.4.2.10 check\_qubit\_cvector()**

```
template<typename Derived >
bool qpp::internal::check_qubit_cvector (
    const Eigen::MatrixBase< Derived > & A ) [noexcept]
```

**6.4.2.11 check\_qubit\_matrix()**

```
template<typename Derived >
bool qpp::internal::check_qubit_matrix (
    const Eigen::MatrixBase< Derived > & A ) [noexcept]
```

**6.4.2.12 check\_qubit\_rvector()**

```
template<typename Derived >
bool qpp::internal::check_qubit_rvector (
    const Eigen::MatrixBase< Derived > & A ) [noexcept]
```



#### 6.4.2.13 check\_qubit\_vector()

```
template<typename Derived >
bool qpp::internal::check_qubit_vector (
    const Eigen::MatrixBase< Derived > & A ) [noexcept]
```

#### 6.4.2.14 check\_rvector()

```
template<typename Derived >
bool qpp::internal::check_rvector (
    const Eigen::MatrixBase< Derived > & A )
```

#### 6.4.2.15 check\_square\_mat()

```
template<typename Derived >
bool qpp::internal::check_square_mat (
    const Eigen::MatrixBase< Derived > & A )
```

#### 6.4.2.16 check\_subsys\_match\_dims()

```
bool qpp::internal::check_subsys_match_dims (
    const std::vector< idx > & subsys,
    const std::vector< idx > & dims ) [inline]
```

#### 6.4.2.17 check\_vector()

```
template<typename Derived >
bool qpp::internal::check_vector (
    const Eigen::MatrixBase< Derived > & A )
```

#### 6.4.2.18 dirsum2()

```
template<typename Derived1 , typename Derived2 >
dyn_mat<typename Derived1::Scalar> qpp::internal::dirsum2 (
    const Eigen::MatrixBase< Derived1 > & A,
    const Eigen::MatrixBase< Derived2 > & B )
```

**6.4.2.19 get\_dim\_subsys()**

```
idx qpp::internal::get_dim_subsys (
    idx sz,
    idx N ) [inline]
```

**6.4.2.20 get\_num\_subsys()**

```
idx qpp::internal::get_num_subsys (
    idx sz,
    idx d ) [inline]
```

**6.4.2.21 kron2()**

```
template<typename Derived1 , typename Derived2 >
dyn_mat<typename Derived1::Scalar> qpp::internal::kron2 (
    const Eigen::MatrixBase< Derived1 > & A,
    const Eigen::MatrixBase< Derived2 > & B )
```

**6.4.2.22 multiidx2n()**

```
idx qpp::internal::multiidx2n (
    const idx *const midx,
    idx numdims,
    const idx *const dims ) [inline], [noexcept]
```

**6.4.2.23 n2multiidx()**

```
void qpp::internal::n2multiidx (
    idx n,
    idx numdims,
    const idx *const dims,
    idx * result ) [inline], [noexcept]
```

**6.4.2.24 variadic\_vector\_emplace()** [1/2]

```
template<typename T >
void qpp::internal::variadic_vector_emplace (
    std::vector< T > & )
```

## 6.4.2.25 variadic\_vector\_emplace() [2/2]

```
template<typename T , typename First , typename... Args>
void qpp::internal::variadic_vector_emplace (
    std::vector< T > & v,
    First && first,
    Args &&... args )
```

## 6.5 qpp::literals Namespace Reference

### Functions

- constexpr `cplx operator"" _i` (unsigned long long int x) noexcept  
*User-defined literal for complex  $i = \sqrt{-1}$  (integer overload)*
- template<char... Bits>  
`ket operator"" _ket` ()  
*Multi-partite qubit ket user-defined literal.*
- template<char... Bits>  
`bra operator"" _bra` ()  
*Multi-partite qubit bra user-defined literal.*
- template<char... Bits>  
`cmat operator"" _prj` ()  
*Multi-partite qubit projector user-defined literal.*

### 6.5.1 Function Documentation

#### 6.5.1.1 operator"" \_bra()

```
template<char... Bits>
bra qpp::literals::operator"" _bra ( )
```

Multi-partite qubit bra user-defined literal.

See also

`qpp::mket()` and `qpp::adjoint()`

Constructs the multi-partite qubit bra  $\langle \text{Bits} |$

#### Template Parameters

<i>Bits</i>	String of binary numbers representing the qubit bra
-------------	---

**Returns**

Multi-partite qubit bra, as a complex dynamic row vector

**6.5.1.2 operator"" \_i()**

```
constexpr cplx qpp::literals::operator"" _i (
    unsigned long long int x ) [inline], [noexcept]
```

User-defined literal for complex  $i = \sqrt{-1}$  (integer overload)

Example:

```
cplx z = 4_i; // type of z is std::complex<double>
```

**6.5.1.3 operator"" \_ket()**

```
template<char... Bits>
ket qpp::literals::operator"" _ket ( )
```

Multi-partite qubit ket user-defined literal.

See also

[qpp::mket\(\)](#)

Constructs the multi-partite qubit ket  $|\text{Bits}\rangle$

**Template Parameters**

<i>Bits</i>	String of binary numbers representing the qubit ket
-------------	---

**Returns**

Multi-partite qubit ket, as a complex dynamic column vector

**6.5.1.4 operator"" \_prj()**

```
template<char... Bits>
cmat qpp::literals::operator"" _prj ( )
```

Multi-partite qubit projector user-defined literal.

See also

[qpp::mprj\(\)](#)

Constructs the multi-partite qubit projector  $|Bits\rangle\langle Bits|$  (in the computational basis)

#### Template Parameters

<i>Bits</i>	String of binary numbers representing the qubit state to project on
-------------	---

#### Returns

Multi-partite qubit projector, as a complex dynamic matrix



## Chapter 7

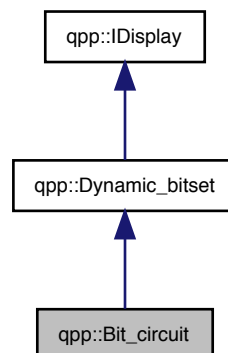
# Class Documentation

### 7.1 qpp::Bit\_circuit Class Reference

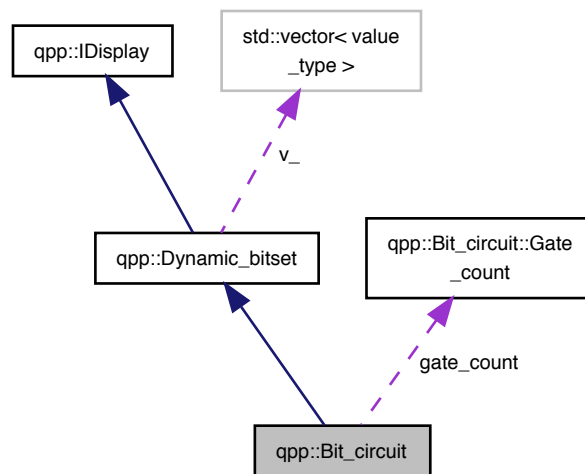
Classical reversible circuit simulator.

```
#include <classes/reversible.h>
```

Inheritance diagram for qpp::Bit\_circuit:



Collaboration diagram for `qpp::Bit_circuit`:



## Classes

- struct [Gate\\_count](#)

## Public Member Functions

- [Bit\\_circuit](#) (const [Dynamic\\_bitset](#) &dynamic\_bitset)  
*Conversion constructor, used to initialize a `qpp::Bit_circuit` with a `qpp::Dynamic_bitset`.*
- [Bit\\_circuit](#) & [X](#) (idx pos)  
*Bit flip.*
- [Bit\\_circuit](#) & [NOT](#) (idx pos)  
*Bit flip.*
- [Bit\\_circuit](#) & [CNOT](#) (const std::vector< idx > &pos)  
*Controlled-NOT.*
- [Bit\\_circuit](#) & [TOF](#) (const std::vector< idx > &pos)  
*Toffoli gate.*
- [Bit\\_circuit](#) & [SWAP](#) (const std::vector< idx > &pos)  
*Swap bits.*
- [Bit\\_circuit](#) & [FRED](#) (const std::vector< idx > &pos)  
*Fredkin gate (Controlled-SWAP)*
- [Bit\\_circuit](#) & [reset](#) () noexcept  
*Reset the circuit all zero, clear all gates.*

## Public Attributes

- struct `qpp::Bit_circuit::Gate_count` [gate\\_count](#)  
*Gate counters.*



## Additional Inherited Members

### 7.1.1 Detailed Description

Classical reversible circuit simulator.

### 7.1.2 Constructor & Destructor Documentation

#### 7.1.2.1 Bit\_circuit()

```
qpp::Bit_circuit::Bit_circuit (
    const Dynamic\_bitset & dynamic_bitset ) [inline]
```

Conversion constructor, used to initialize a [qpp::Bit\\_circuit](#) with a [qpp::Dynamic\\_bitset](#).

#### Parameters

<i>dynamic_bitset</i>	Dynamic bitset
-----------------------	----------------

### 7.1.3 Member Function Documentation

#### 7.1.3.1 CNOT()

```
Bit\_circuit& qpp::Bit_circuit::CNOT (
    const std::vector< idx > & pos ) [inline]
```

Controlled-NOT.

#### Parameters

<i>pos</i>	Bit position in the circuit
------------	-----------------------------

#### Returns

Reference to the current instance

#### 7.1.3.2 FRED()

```
Bit\_circuit& qpp::Bit_circuit::FRED (
    const std::vector< idx > & pos ) [inline]
```

Fredkin gate (Controlled-SWAP)

#### Parameters

<i>pos</i>	Bit positions in the circuit, in the order control-target-target
------------	--

#### Returns

Reference to the current instance

#### 7.1.3.3 NOT()

```
Bit_circuit& qpp::Bit_circuit::NOT (
    idx pos ) [inline]
```

Bit flip.

#### See also

[qpp::Bit\\_circuit::X\(\)](#)

#### Parameters

<i>pos</i>	Bit position in the circuit
------------	-----------------------------

#### Returns

Reference to the current instance

#### 7.1.3.4 reset()

```
Bit_circuit& qpp::Bit_circuit::reset ( ) [inline], [noexcept]
```

Reset the circuit all zero, clear all gates.

#### Returns

Reference to the current instance

#### 7.1.3.5 SWAP()

```
Bit_circuit& qpp::Bit_circuit::SWAP (
    const std::vector< idx > & pos ) [inline]
```

Swap bits.

## Parameters

<i>pos</i>	Bit positions in the circuit
------------	------------------------------

## Returns

Reference to the current instance

## 7.1.3.6 TOF()

```
Bit_circuit& qpp::Bit_circuit::TOF (  
    const std::vector< idx > & pos ) [inline]
```

Toffoli gate.

## Parameters

<i>pos</i>	Bit positions in the circuit, in the order control-control-target
------------	---

## Returns

Reference to the current instance

## 7.1.3.7 X()

```
Bit_circuit& qpp::Bit_circuit::X (  
    idx pos ) [inline]
```

Bit flip.

## See also

[qpp::Bit\\_circuit::NOT\(\)](#)

## Parameters

<i>pos</i>	Bit position in the circuit
------------	-----------------------------

## Returns

Reference to the current instance

## 7.1.4 Member Data Documentation

### 7.1.4.1 gate\_count

```
struct qpp::Bit_circuit::Gate_count qpp::Bit_circuit::gate_count
```

Gate counters.

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

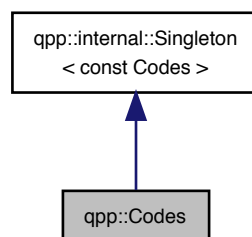
- [classes/reversible.h](#)

## 7.2 qpp::Codes Class Reference

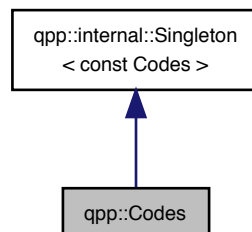
const Singleton class that defines quantum error correcting codes

```
#include <classes/codes.h>
```

Inheritance diagram for qpp::Codes:



Collaboration diagram for qpp::Codes:



## Public Types

- enum [Type](#) { [Type::FIVE\\_QUBIT](#) = 1, [Type::SEVEN\\_QUBIT\\_STEANE](#), [Type::NINE\\_QUBIT\\_SHOR](#) }  
*Code types, add more codes here if needed.*

## Public Member Functions

- [ket codeword](#) ([Type](#) type, [idx](#) i) const  
*Returns the codeword of the specified code type.*

## Private Member Functions

- [Codes](#) ()  
*Default constructor.*
- [~Codes](#) ()=default  
*Default destructor.*

## Friends

- class [internal::Singleton](#)< const [Codes](#) >

## Additional Inherited Members

### 7.2.1 Detailed Description

const Singleton class that defines quantum error correcting codes

### 7.2.2 Member Enumeration Documentation

#### 7.2.2.1 Type

```
enum qpp::Codes::Type    [strong]
```

Code types, add more codes here if needed.

See also

[qpp::Codes::codeword\(\)](#)

#### Enumerator

FIVE_QUBIT	[[5,1,3]] qubit code
SEVEN_QUBIT_STEANE	[[7,1,3]] Steane qubit code
NINE_QUBIT_SHOR	[[9,1,3]] Shor qubit code

## 7.2.3 Constructor & Destructor Documentation

### 7.2.3.1 Codes()

```
qpp::Codes::Codes ( ) [inline], [private]
```

Default constructor.

### 7.2.3.2 ~Codes()

```
qpp::Codes::~~Codes ( ) [private], [default]
```

Default destructor.

## 7.2.4 Member Function Documentation

### 7.2.4.1 codeword()

```
ket qpp::Codes::codeword (
    Type type,
    idx i ) const [inline]
```

Returns the codeword of the specified code type.

See also

[qpp::Codes::Type](#)

#### Parameters

<i>type</i>	Code type
<i>i</i>	Codeword index

#### Returns

*i*-th codeword of the code *type*

## 7.2.5 Friends And Related Function Documentation

## 7.2.5.1 internal::Singleton&lt; const Codes &gt;

```
friend class internal::Singleton< const Codes > [friend]
```

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

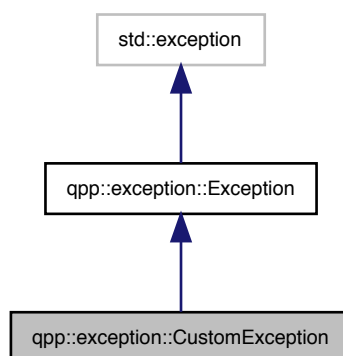
- [classes/codes.h](#)

## 7.3 qpp::exception::CustomException Class Reference

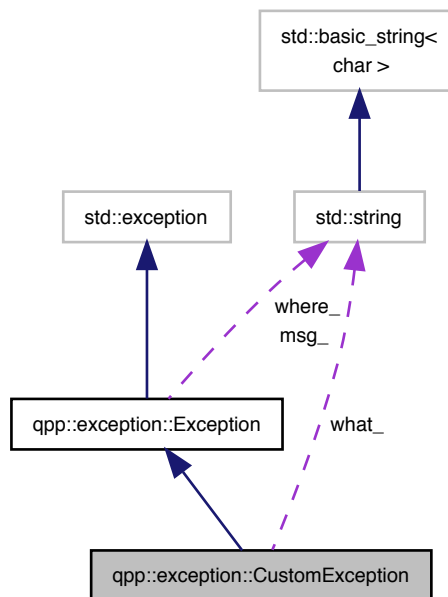
Custom exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::CustomException:



Collaboration diagram for `qpp::exception::CustomException`:



## Public Member Functions

- [CustomException](#) (const std::string &where, const std::string &[what](#))

## Private Member Functions

- std::string [type\\_description](#) () const override  
*[Exception](#) type description.*

## Private Attributes

- std::string [what\\_](#) {}

### 7.3.1 Detailed Description

Custom exception.

Custom exception, the user must provide a custom message

### 7.3.2 Constructor & Destructor Documentation



### 7.3.2.1 CustomException()

```
qpp::exception::CustomException::CustomException (
    const std::string & where,
    const std::string & what ) [inline]
```

## 7.3.3 Member Function Documentation

### 7.3.3.1 type\_description()

```
std::string qpp::exception::CustomException::type_description( ) const [inline], [override],
[private], [virtual]
```

[Exception](#) type description.

#### Returns

[Exception](#) type description

Implements [qpp::exception::Exception](#).

## 7.3.4 Member Data Documentation

### 7.3.4.1 what\_

```
std::string qpp::exception::CustomException::what_ {} [private]
```

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

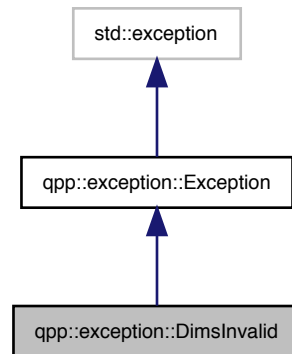
- [classes/exception.h](#)

## 7.4 qpp::exception::DimsInvalid Class Reference

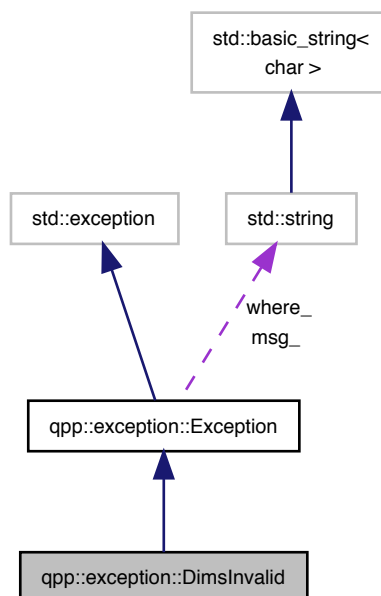
Invalid dimension(s) exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::DimsInvalid:



Collaboration diagram for qpp::exception::DimsInvalid:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.4.1 Detailed Description

Invalid dimension(s) exception.

`std::vector<idx>` of dimensions has zero size or contains zeros

### 7.4.2 Member Function Documentation

#### 7.4.2.1 type\_description()

```
std::string qpp::exception::DimsInvalid::type_description ( ) const [inline], [override],  
[virtual]
```

*Exception* type description.

#### Returns

*Exception* type description

Implements `qpp::exception::Exception`.

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

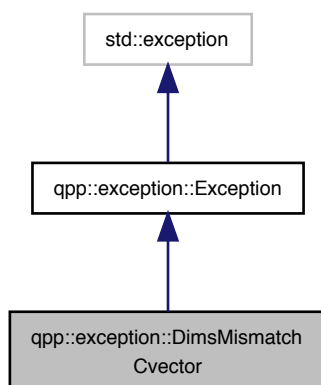
- `classes/exception.h`

## 7.5 qpp::exception::DimsMismatchCvector Class Reference

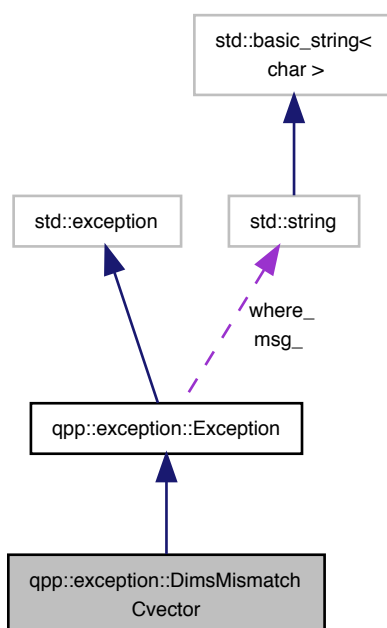
Dimension(s) mismatch column vector size exception.

```
#include <classes/exception.h>
```

Inheritance diagram for `qpp::exception::DimsMismatchCvector`:



Collaboration diagram for `qpp::exception::DimsMismatchCvector`:



## Public Member Functions

- `std::string` [type\\_description](#) () const override  
*[Exception](#) type description.*

### 7.5.1 Detailed Description

Dimension(s) mismatch column vector size exception.

Product of the elements of `std::vector<idx>` of dimensions is not equal to the number of elements of the `Eigen::Matrix` (assumed to be a column vector)

### 7.5.2 Member Function Documentation

#### 7.5.2.1 type\_description()

```
std::string qpp::exception::DimsMismatchCvector::type_description ( ) const [inline], [override], [virtual]
```

[Exception](#) type description.

#### Returns

[Exception](#) type description

Implements [qpp::exception::Exception](#).

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

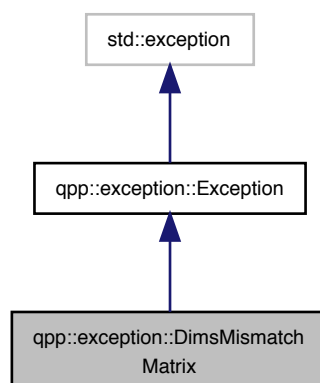
- [classes/exception.h](#)

## 7.6 qpp::exception::DimsMismatchMatrix Class Reference

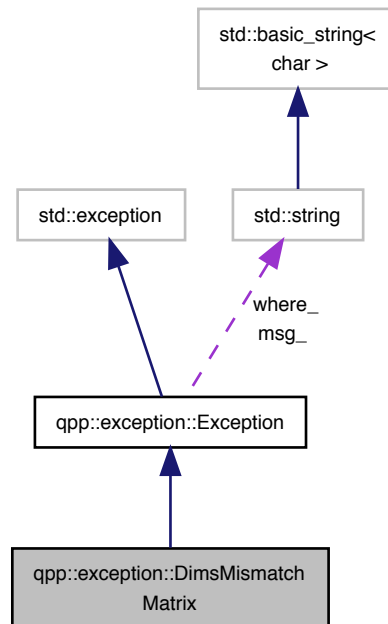
Dimension(s) mismatch matrix size exception.

```
#include <classes/exception.h>
```

Inheritance diagram for `qpp::exception::DimsMismatchMatrix`:



Collaboration diagram for `qpp::exception::DimsMismatchMatrix`:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.6.1 Detailed Description

Dimension(s) mismatch matrix size exception.

Product of the elements of `std::vector<idx>` of dimensions is not equal to the number of rows of the `Eigen::Matrix` (assumed to be a square matrix)

### 7.6.2 Member Function Documentation

### 7.6.2.1 type\_description()

```
std::string qpp::exception::DimsMismatchMatrix::type_description ( ) const [inline], [override],  
[virtual]
```

Exception type description.

#### Returns

Exception type description

Implements [qpp::exception::Exception](#).

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

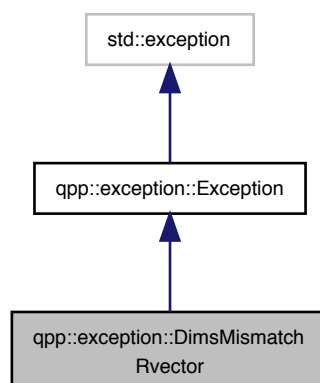
- [classes/exception.h](#)

## 7.7 qpp::exception::DimsMismatchRvector Class Reference

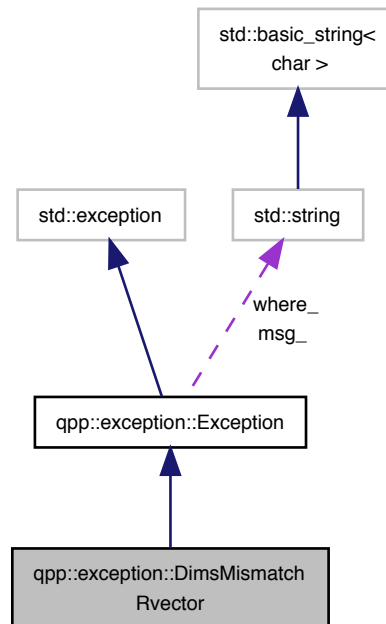
Dimension(s) mismatch row vector size exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::DimsMismatchRvector:



Collaboration diagram for `qpp::exception::DimsMismatchRvector`:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.7.1 Detailed Description

Dimension(s) mismatch row vector size exception.

Product of the elements of `std::vector<idx>` of dimensions is not equal to the number of elements of the Eigen::↔ Matrix (assumed to be a row vector)

### 7.7.2 Member Function Documentation



## 7.7.2.1 type\_description()

```
std::string qpp::exception::DimsMismatchRvector::type_description ( ) const [inline], [override],  
[virtual]
```

Exception type description.

## Returns

Exception type description

Implements [qpp::exception::Exception](#).

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

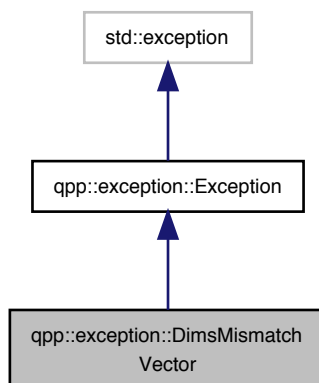
- [classes/exception.h](#)

## 7.8 qpp::exception::DimsMismatchVector Class Reference

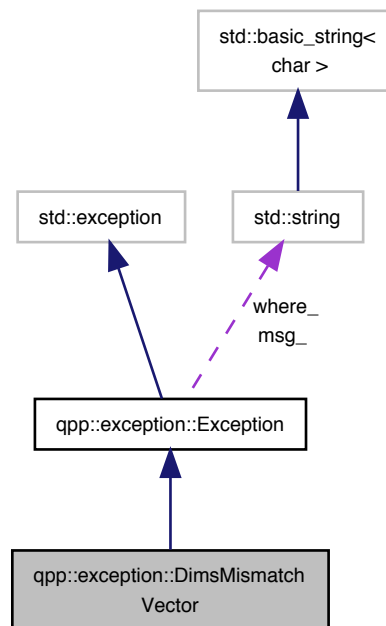
Dimension(s) mismatch vector size exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::DimsMismatchVector:



Collaboration diagram for `qpp::exception::DimsMismatchVector`:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.8.1 Detailed Description

Dimension(s) mismatch vector size exception.

Product of the elements of `std::vector<idx>` of dimensions is not equal to the number of elements of the Eigen::↔ Matrix (assumed to be a row/column vector)

### 7.8.2 Member Function Documentation

### 7.8.2.1 type\_description()

```
std::string qpp::exception::DimsMismatchVector::type_description ( ) const [inline], [override],  
[virtual]
```

Exception type description.

#### Returns

Exception type description

Implements [qpp::exception::Exception](#).

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

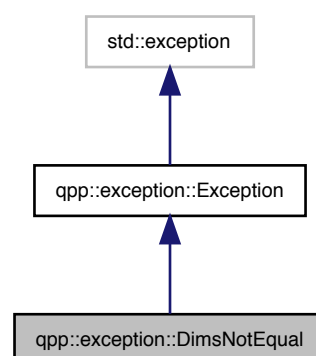
- [classes/exception.h](#)

## 7.9 qpp::exception::DimsNotEqual Class Reference

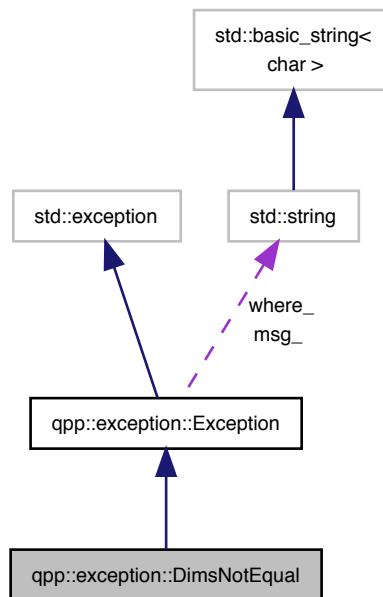
Dimensions not equal exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::DimsNotEqual:



Collaboration diagram for `qpp::exception::DimsNotEqual`:



## Public Member Functions

- `std::string type_description () const` override  
*Exception type description.*

### 7.9.1 Detailed Description

Dimensions not equal exception.

Local/global dimensions are not equal

### 7.9.2 Member Function Documentation

#### 7.9.2.1 type\_description()

```
std::string qpp::exception::DimsNotEqual::type_description ( ) const [inline], [override],
[virtual]
```

*Exception type description.*

## Returns

[Exception](#) type description

Implements [qpp::exception::Exception](#).

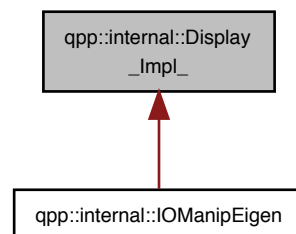
The documentation for this class was generated from the following file:

- [classes/exception.h](#)

## 7.10 qpp::internal::Display\_Impl\_ Struct Reference

```
#include <internal/util.h>
```

Inheritance diagram for qpp::internal::Display\_Impl\_:



### Public Member Functions

- `template<typename T >`  
`std::ostream & display\_impl\_ (const T &A, std::ostream &os, double chop=qpp::chop) const`

### 7.10.1 Member Function Documentation

#### 7.10.1.1 `display_impl_()`

```
template<typename T >
std::ostream& qpp::internal::Display_Impl_::display_impl_ (
    const T & A,
    std::ostream & os,
    double chop = qpp::chop ) const [inline]
```

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

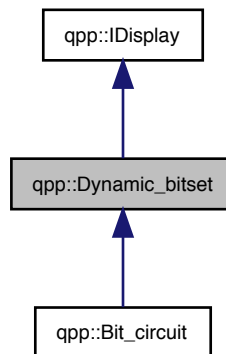
- [internal/util.h](#)

## 7.11 qpp::Dynamic\_bitset Class Reference

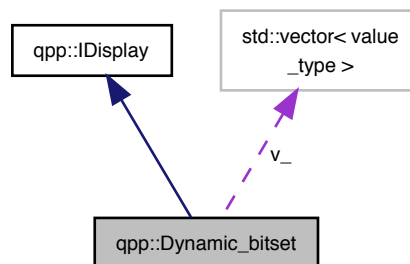
Dynamic bitset class, allows the specification of the number of bits at runtime (unlike `std::bitset<N>`)

```
#include <classes/reversible.h>
```

Inheritance diagram for `qpp::Dynamic_bitset`:



Collaboration diagram for `qpp::Dynamic_bitset`:



### Public Types

- using `value_type` = unsigned int  
*Type of the storage elements.*
- using `storage_type` = `std::vector<value_type>`  
*Type of the storage.*

## Public Member Functions

- [Dynamic\\_bitset](#) ([idx](#) N)  
*Constructor, initializes all bits to false (zero)*
- const [storage\\_type](#) & [data](#) () const  
*Raw storage space of the bitset.*
- [idx](#) [size](#) () const noexcept  
*Number of bits stored in the bitset.*
- [idx](#) [storage\\_size](#) () const noexcept  
*Size of the underlying storage space (in units of value\_type, unsigned int by default)*
- [idx](#) [count](#) () const noexcept  
*Number of bits set to one in the bitset (Hamming weight)*
- bool [get](#) ([idx](#) pos) const noexcept  
*The value of the bit at position pos.*
- bool [none](#) () const noexcept  
*Checks whether none of the bits are set.*
- bool [all](#) () const noexcept  
*Checks whether all bits are set.*
- bool [any](#) () const noexcept  
*Checks whether any bit is set.*
- [Dynamic\\_bitset](#) & [set](#) ([idx](#) pos, bool value=true)  
*Sets the bit at position pos.*
- [Dynamic\\_bitset](#) & [set](#) () noexcept  
*Set all bits to true.*
- [Dynamic\\_bitset](#) & [rand](#) ([idx](#) pos, double p=0.5)  
*Sets the bit at position pos according to a Bernoulli(p) distribution.*
- [Dynamic\\_bitset](#) & [rand](#) (double p=0.5)  
*Sets all bits according to a Bernoulli(p) distribution.*
- [Dynamic\\_bitset](#) & [reset](#) ([idx](#) pos)  
*Sets the bit at position pos to false.*
- [Dynamic\\_bitset](#) & [reset](#) () noexcept  
*Sets all bits to false.*
- [Dynamic\\_bitset](#) & [flip](#) ([idx](#) pos)  
*Flips the bit at position pos.*
- [Dynamic\\_bitset](#) & [flip](#) () noexcept  
*Flips all bits.*
- bool [operator==](#) (const [Dynamic\\_bitset](#) &rhs) const noexcept  
*Equality operator.*
- bool [operator!=](#) (const [Dynamic\\_bitset](#) &rhs) const noexcept  
*Inequality operator.*
- [idx](#) [operator-](#) (const [Dynamic\\_bitset](#) &rhs) const noexcept  
*Number of places the two bitsets differ (Hamming distance)*
- template<class CharT = char, class Traits = std::char\_traits<CharT>, class Allocator = std::allocator<CharT>>  
std::basic\_string< CharT, Traits, Allocator > [to\\_string](#) (CharT zero=CharT('0'), CharT one=CharT('1')) const  
*String representation.*

## Protected Member Functions

- [idx](#) [index\\_](#) ([idx](#) pos) const  
*Index of the pos bit in the storage space.*
- [idx](#) [offset\\_](#) ([idx](#) pos) const  
*Offset of the pos bit in the storage space relative to its index.*

## Protected Attributes

- `idx storage_size_`  
*Storage size.*
- `idx N_`  
*Number of bits.*
- `std::vector< value_type > v_`  
*Storage space.*

## Private Member Functions

- `std::ostream & display (std::ostream &os) const` override  
*`qpp::IDisplay::display()` override, displays the bitset bit by bit*

### 7.11.1 Detailed Description

Dynamic bitset class, allows the specification of the number of bits at runtime (unlike `std::bitset<N>`)

### 7.11.2 Member Typedef Documentation

#### 7.11.2.1 storage\_type

```
using qpp::Dynamic_bitset::storage_type = std::vector<value_type>
```

Type of the storage.

#### 7.11.2.2 value\_type

```
using qpp::Dynamic_bitset::value_type = unsigned int
```

Type of the storage elements.

### 7.11.3 Constructor & Destructor Documentation

#### 7.11.3.1 Dynamic\_bitset()

```
qpp::Dynamic_bitset::Dynamic_bitset (
    idx N ) [inline]
```

Constructor, initializes all bits to false (zero)



**Parameters**

<i>N</i>	Number of bits in the bitset
----------	------------------------------

**7.11.4 Member Function Documentation****7.11.4.1 all()**

```
bool qpp::Dynamic_bitset::all ( ) const [inline], [noexcept]
```

Checks whether all bits are set.

**Returns**

True if all of the bits are set

**7.11.4.2 any()**

```
bool qpp::Dynamic_bitset::any ( ) const [inline], [noexcept]
```

Checks whether any bit is set.

**Returns**

True if any of the bits is set

**7.11.4.3 count()**

```
idx qpp::Dynamic_bitset::count ( ) const [inline], [noexcept]
```

Number of bits set to one in the bitset (Hamming weight)

**Returns**

Hamming weight

#### 7.11.4.4 data()

```
const storage_type& qpp::Dynamic_bitset::data ( ) const [inline]
```

Raw storage space of the bitset.

#### Returns

Const reference to the underlying storage space

#### 7.11.4.5 display()

```
std::ostream& qpp::Dynamic_bitset::display (
    std::ostream & os ) const [inline], [override], [private], [virtual]
```

[qpp::IDisplay::display\(\)](#) override, displays the bitset bit by bit

## Parameters

<i>os</i>	Output stream
-----------	---------------

## Returns

Reference to the output stream

Implements [qpp::IDisplay](#).

## 7.11.4.6 flip() [1/2]

```
Dynamic_bitset& qpp::Dynamic_bitset::flip (  
    idx pos ) [inline]
```

Flips the bit at position *pos*.

## Parameters

<i>pos</i>	Position in the bitset
------------	------------------------

## Returns

Reference to the current instance

## 7.11.4.7 flip() [2/2]

```
Dynamic_bitset& qpp::Dynamic_bitset::flip ( ) [inline], [noexcept]
```

Flips all bits.

## Returns

Reference to the current instance

## 7.11.4.8 get()

```
bool qpp::Dynamic_bitset::get (  
    idx pos ) const [inline], [noexcept]
```

The value of the bit at position *pos*.

**Parameters**

<i>pos</i>	Position in the bitset
------------	------------------------

**Returns**

The value of the bit at position *pos*

**7.11.4.9 index\_()**

```
idx qpp::Dynamic_bitset::index_ (
    idx pos ) const [inline], [protected]
```

Index of the *pos* bit in the storage space.

**Parameters**

<i>pos</i>	Bit location
------------	--------------

**Returns**

Index of the *pos* bit in the storage space

**7.11.4.10 none()**

```
bool qpp::Dynamic_bitset::none ( ) const [inline], [noexcept]
```

Checks whether none of the bits are set.

**Returns**

True if none of the bits are set

**7.11.4.11 offset\_()**

```
idx qpp::Dynamic_bitset::offset_ (
    idx pos ) const [inline], [protected]
```

Offset of the *pos* bit in the storage space relative to its index.

## Parameters

<i>pos</i>	Bit location
------------	--------------

## Returns

Offset of the *pos* bit in the storage space relative to its index

## 7.11.4.12 operator!=(())

```
bool qpp::Dynamic_bitset::operator!= (
    const Dynamic_bitset & rhs ) const [inline], [noexcept]
```

Inequality operator.

## Parameters

<i>rhs</i>	<a href="#">Dynamic_bitset</a> against which the inequality is being tested
------------	---

## Returns

True if the bitsets are not equal (bit by bit), false otherwise

## 7.11.4.13 operator-()

```
idx qpp::Dynamic_bitset::operator- (
    const Dynamic_bitset & rhs ) const [inline], [noexcept]
```

Number of places the two bitsets differ (Hamming distance)

## Parameters

<i>rhs</i>	<a href="#">Dynamic_bitset</a> against which the the Hamming distance is computed
------------	---

## Returns

Hamming distance

## 7.11.4.14 operator==(())

```
bool qpp::Dynamic_bitset::operator== (
    const Dynamic_bitset & rhs ) const [inline], [noexcept]
```

Equality operator.

## Parameters

<i>rhs</i>	<a href="#">Dynamic_bitset</a> against which the equality is being tested
------------	---

## Returns

True if the bitsets are equal (bit by bit), false otherwise

7.11.4.15 `rand()` [1/2]

```
Dynamic_bitset& qpp::Dynamic_bitset::rand (
    idx pos,
    double p = 0.5 ) [inline]
```

Sets the bit at position *pos* according to a Bernoulli(*p*) distribution.

## Parameters

<i>pos</i>	Position in the bitset
<i>p</i>	Probability

## Returns

Reference to the current instance

7.11.4.16 `rand()` [2/2]

```
Dynamic_bitset& qpp::Dynamic_bitset::rand (
    double p = 0.5 ) [inline]
```

Sets all bits according to a Bernoulli(*p*) distribution.

## Parameters

<i>p</i>	Probability
----------	-------------

## Returns

Reference to the current instance

#### 7.11.4.17 `reset()` [1/2]

```
Dynamic_bitset& qpp::Dynamic_bitset::reset (
    idx pos ) [inline]
```

Sets the bit at position *pos* to false.

##### Parameters

<i>pos</i>	Position in the bitset
------------	------------------------

##### Returns

Reference to the current instance

#### 7.11.4.18 `reset()` [2/2]

```
Dynamic_bitset& qpp::Dynamic_bitset::reset ( ) [inline], [noexcept]
```

Sets all bits to false.

##### Returns

Reference to the current instance

#### 7.11.4.19 `set()` [1/2]

```
Dynamic_bitset& qpp::Dynamic_bitset::set (
    idx pos,
    bool value = true ) [inline]
```

Sets the bit at position *pos*.

##### Parameters

<i>pos</i>	Position in the bitset
<i>value</i>	Bit value

##### Returns

Reference to the current instance



7.11.4.20 `set()` [2/2]

```
Dynamic_bitset& qpp::Dynamic_bitset::set ( ) [inline], [noexcept]
```

Set all bits to true.

**Returns**

Reference to the current instance

7.11.4.21 `size()`

```
idx qpp::Dynamic_bitset::size ( ) const [inline], [noexcept]
```

Number of bits stored in the bitset.

**Returns**

Number of bits stored in the bitset

7.11.4.22 `storage_size()`

```
idx qpp::Dynamic_bitset::storage_size ( ) const [inline], [noexcept]
```

Size of the underlying storage space (in units of `value_type`, unsigned int by default)

**Returns**

Size of the underlying storage space

7.11.4.23 `to_string()`

```
template<class CharT = char, class Traits = std::char_traits<CharT>, class Allocator = std::
::allocator<CharT>>
std::basic_string<CharT, Traits, Allocator> qpp::Dynamic_bitset::to_string (
    CharT zero = CharT('0'),
    CharT one = CharT('1') ) const [inline]
```

String representation.

**Template Parameters**

<i>CharT</i>	String character type
<i>Traits</i>	String traits
<i>Allocator</i>	String Allocator

**Parameters**

<i>zero</i>	Character representing the zero
<i>one</i>	Character representing the one

**Returns**

The bitset as a string

**7.11.5 Member Data Documentation****7.11.5.1 N\_**

`idx` `qpp::Dynamic_bitset::N_` [protected]

Number of bits.

**7.11.5.2 storage\_size\_**

`idx` `qpp::Dynamic_bitset::storage_size_` [protected]

Storage size.

**7.11.5.3 v\_**

`std::vector<value_type>` `qpp::Dynamic_bitset::v_` [protected]

Storage space.

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

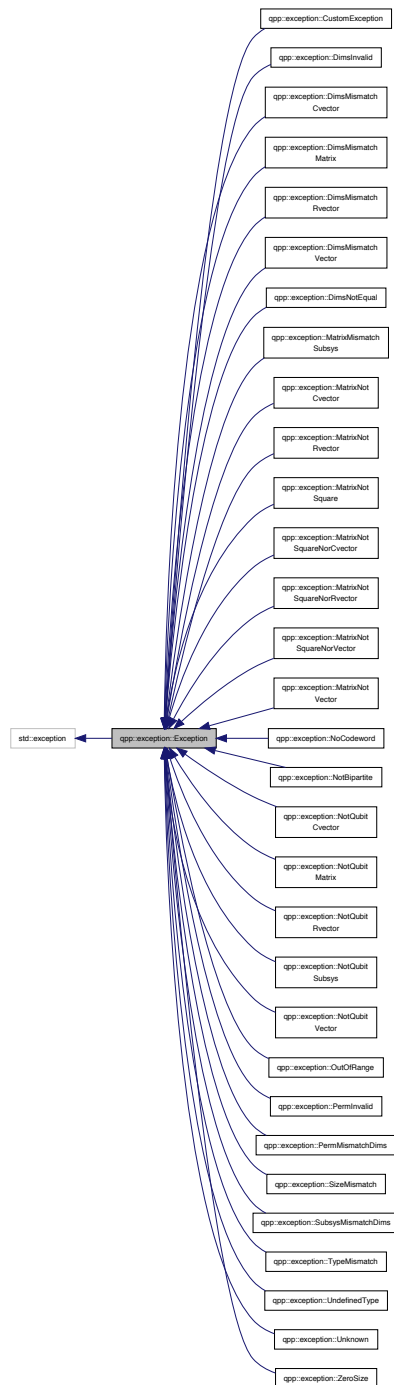
- [classes/reversible.h](#)

## 7.12 qpp::exception::Exception Class Reference

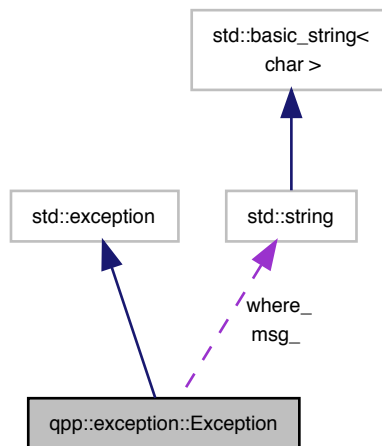
Base class for generating Quantum++ custom exceptions.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::Exception:



Collaboration diagram for `qpp::exception::Exception`:



## Public Member Functions

- [Exception](#) (const std::string &where)  
*Constructs an exception.*
- virtual const char \* [what](#) () const noexcept override  
*Overrides std::exception::what()*
- virtual std::string [type\\_description](#) () const =0  
*Exception type description.*

## Private Attributes

- std::string [where\\_](#)
- std::string [msg\\_](#)

### 7.12.1 Detailed Description

Base class for generating Quantum++ custom exceptions.

Derive from this class if more exceptions are needed, making sure to override [qpp::exception::Exception::type\\_description\(\)](#) in the derived class and to inherit the constructor [qpp::exception::Exception::Exception\(\)](#). Preferably keep your newly defined exception classes in the namespace [qpp::exception](#).

Example:

```

namespace qpp
{
namespace exception
{
    class ZeroSize : public Exception
    {
    public:
        std::string type_description() const override
        {
            return "Object has zero size";
        }

        // inherit the base class' qpp::exception::Exception constructor
        using Exception::Exception;
    };
} // namespace exception
} // namespace qpp

```

## 7.12.2 Constructor & Destructor Documentation

### 7.12.2.1 Exception()

```

qpp::exception::Exception::Exception (
    const std::string & where ) [inline]

```

Constructs an exception.

#### Parameters

<i>where</i>	Text representing where the exception occurred
--------------	--

## 7.12.3 Member Function Documentation

### 7.12.3.1 type\_description()

```

std::string qpp::exception::Exception::type_description ( ) const [inline], [pure virtual]

```

[Exception](#) type description.

#### Returns

[Exception](#) type description

Implemented in [qpp::exception::CustomException](#), [qpp::exception::UndefinedType](#), [qpp::exception::SizeMismatch](#), [qpp::exception::TypeMismatch](#), [qpp::exception::OutOfRange](#), [qpp::exception::NoCodeword](#), [qpp::exception::NotBipartite](#), [qpp::exception::NotQubitSubsys](#), [qpp::exception::NotQubitVector](#), [qpp::exception::NotQubitRvector](#), [qpp::exception::NotQubitCvector](#), [qpp::exception::NotQubitMatrix](#), [qpp::exception::PermMismatchDims](#), [qpp::exception::PermInvalid](#), [qpp::exception::SubsysMismatchDims](#), [qpp::exception::DimsMismatchVector](#), [qpp::exception::DimsMismatchRvector](#), [qpp::exception::DimsMismatchCvector](#), [qpp::exception::DimsMismatchMatrix](#), [qpp::exception::DimsNotEqual](#), [qpp::exception::DimsInvalid](#), [qpp::exception::MatrixMismatchSubsys](#), [qpp::exception::MatrixNotSquareNorVector](#), [qpp::exception::MatrixNotSquareNorRvector](#), [qpp::exception::MatrixNotSquareNorCvector](#), [qpp::exception::MatrixNotVector](#), [qpp::exception::MatrixNotRvector](#), [qpp::exception::MatrixNotCvector](#), [qpp::exception::MatrixNotSquare](#), [qpp::exception::ZeroSize](#), and [qpp::exception::Unknown](#).

### 7.12.3.2 what()

```
virtual const char* qpp::exception::Exception::what ( ) const [inline], [override], [virtual],  
[noexcept]
```

Overrides `std::exception::what()`

#### Returns

[Exception](#) description

## 7.12.4 Member Data Documentation

### 7.12.4.1 msg\_

```
std::string qpp::exception::Exception::msg_ [mutable], [private]
```

### 7.12.4.2 where\_

```
std::string qpp::exception::Exception::where_ [private]
```

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

- [classes/exception.h](#)

## 7.13 qpp::Bit\_circuit::Gate\_count Struct Reference

```
#include <classes/reversible.h>
```

### Public Attributes

- [idx NOT](#) = 0
- [idx & X](#) = NOT
- [idx CNOT](#) = 0
- [idx SWAP](#) = 0
- [idx FRED](#) = 0
- [idx TOF](#) = 0

### 7.13.1 Member Data Documentation

#### 7.13.1.1 CNOT

```
idx qpp::Bit_circuit::Gate_count::CNOT = 0
```

#### 7.13.1.2 FRED

```
idx qpp::Bit_circuit::Gate_count::FRED = 0
```

#### 7.13.1.3 NOT

```
idx qpp::Bit_circuit::Gate_count::NOT = 0
```

#### 7.13.1.4 SWAP

```
idx qpp::Bit_circuit::Gate_count::SWAP = 0
```

#### 7.13.1.5 TOF

```
idx qpp::Bit_circuit::Gate_count::TOF = 0
```

#### 7.13.1.6 X

```
idx& qpp::Bit_circuit::Gate_count::X = NOT
```

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

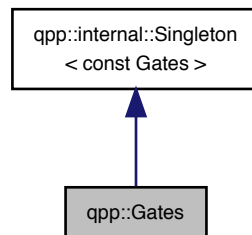
- [classes/reversible.h](#)

## 7.14 qpp::Gates Class Reference

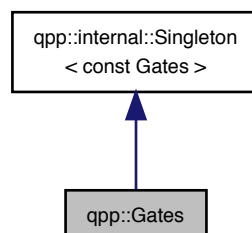
const Singleton class that implements most commonly used gates

```
#include <classes/gates.h>
```

Inheritance diagram for qpp::Gates:



Collaboration diagram for qpp::Gates:



### Public Member Functions

- `cmat Rn` (double theta, const std::vector< double > &n) const  
*Qubit rotation of theta about the 3-dimensional real (unit) vector n.*
- `cmat Zd` (idx D=2) const  
*Generalized Z gate for qudits.*
- `cmat Fd` (idx D=2) const  
*Fourier transform gate for qudits.*
- `cmat Xd` (idx D=2) const  
*Generalized X gate for qudits.*
- `template<typename Derived = Eigen::MatrixXcd>`  
`Derived Id` (idx D=2) const



*Identity gate.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > CTRL (const Eigen::MatrixBase< Derived > &A, const std::vector<`  
`idx > &ctrl, const std::vector< idx > &subsys, idx N, idx d=2) const`

*Generates the multi-partite multiple-controlled-A gate in matrix form.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > expandout (const Eigen::MatrixBase< Derived > &A, idx pos, const`  
`std::vector< idx > &dims) const`

*Expands out.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > expandout (const Eigen::MatrixBase< Derived > &A, idx pos, const`  
`std::initializer_list< idx > &dims) const`

*Expands out.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > expandout (const Eigen::MatrixBase< Derived > &A, idx pos, idx N,`  
`idx d=2) const`

*Expands out.*

## Public Attributes

- `cmat Id2 {cmat::Identity(2, 2)}`

*Identity gate.*

- `cmat H {cmat::Zero(2, 2)}`

*Hadamard gate.*

- `cmat X {cmat::Zero(2, 2)}`

*Pauli Sigma-X gate.*

- `cmat Y {cmat::Zero(2, 2)}`

*Pauli Sigma-Y gate.*

- `cmat Z {cmat::Zero(2, 2)}`

*Pauli Sigma-Z gate.*

- `cmat S {cmat::Zero(2, 2)}`

*S gate.*

- `cmat T {cmat::Zero(2, 2)}`

*T gate.*

- `cmat CNOT {cmat::Identity(4, 4)}`

*Controlled-NOT control target gate.*

- `cmat CZ {cmat::Identity(4, 4)}`

*Controlled-Phase gate.*

- `cmat CNOTba {cmat::Zero(4, 4)}`

*Controlled-NOT target control gate.*

- `cmat SWAP {cmat::Identity(4, 4)}`

*SWAP gate.*

- `cmat TOF {cmat::Identity(8, 8)}`

*Toffoli gate.*

- `cmat FRED {cmat::Identity(8, 8)}`

*Fredkin gate.*

## Private Member Functions

- `Gates ()`

*Initializes the gates.*

- `~Gates ()=default`

*Default destructor.*

## Friends

- class [internal::Singleton< const Gates >](#)

## Additional Inherited Members

### 7.14.1 Detailed Description

const Singleton class that implements most commonly used gates

### 7.14.2 Constructor & Destructor Documentation

#### 7.14.2.1 Gates()

```
qpp::Gates::Gates ( ) [inline], [private]
```

Initializes the gates.

#### 7.14.2.2 ~Gates()

```
qpp::Gates::~~Gates ( ) [private], [default]
```

Default destructor.

### 7.14.3 Member Function Documentation

#### 7.14.3.1 CTRL()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::Gates::CTRL (
    const Eigen::MatrixBase< Derived > & A,
    const std::vector< idx > & ctrl,
    const std::vector< idx > & subsys,
    idx N,
    idx d = 2 ) const [inline]
```

Generates the multi-partite multiple-controlled-*A* gate in matrix form.

See also

[qpp::applyCTRL\(\)](#)

Note

The dimension of the gate *A* must match the dimension of *subsys*

## Parameters

<i>A</i>	Eigen expression
<i>ctrl</i>	Control subsystem indexes
<i>subsys</i>	Subsystem indexes where the gate <i>A</i> is applied
<i>N</i>	Total number of subsystems
<i>d</i>	Subsystem dimensions

## Returns

CTRL-A gate, as a matrix over the same scalar field as *A*

7.14.3.2 `expandout()` [1/3]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::Gates::expandout (
    const Eigen::MatrixBase< Derived > & A,
    idx pos,
    const std::vector< idx > & dims ) const [inline]
```

Expands out.

See also

[qpp::kron\(\)](#)

Expands out *A* as a matrix in a multi-partite system. Faster than using [qpp::kron\(I, I, ..., I, A, I, ..., I\)](#).

## Parameters

<i>A</i>	Eigen expression
<i>pos</i>	Position
<i>dims</i>	Dimensions of the multi-partite system

## Returns

Tensor product  $I \otimes \dots \otimes I \otimes A \otimes I \otimes \dots \otimes I$ , with *A* on position *pos*, as a dynamic matrix over the same scalar field as *A*

7.14.3.3 `expandout()` [2/3]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::Gates::expandout (
    const Eigen::MatrixBase< Derived > & A,
    idx pos,
    const std::initializer_list< idx > & dims ) const [inline]
```

Expands out.

See also

[qpp::kron\(\)](#)

Expands out  $A$  as a matrix in a multi-partite system. Faster than using [qpp::kron](#)( $I, I, \dots, I, A, I, \dots, I$ ).

Note

The `std::initializer_list` overload exists because otherwise, in the degenerate case when *dims* has only one element, the one element list is implicitly converted to the element's underlying type, i.e. [qpp::idx](#), which has the net effect of picking the wrong (non-vector) `qpp::expandout()` overload

Parameters

$A$	Eigen expression
$pos$	Position
$dims$	Dimensions of the multi-partite system

Returns

Tensor product  $I \otimes \dots \otimes I \otimes A \otimes I \otimes \dots \otimes I$ , with  $A$  on position  $pos$ , as a dynamic matrix over the same scalar field as  $A$

#### 7.14.3.4 `expandout()` [3/3]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::Gates::expandout (
    const Eigen::MatrixBase< Derived > & A,
    idx pos,
    idx N,
    idx d = 2 ) const [inline]
```

Expands out.

See also

[qpp::kron\(\)](#)

Expands out  $A$  as a matrix in a multi-partite system. Faster than using [qpp::kron](#)( $I, I, \dots, I, A, I, \dots, I$ ).

Parameters

$A$	Eigen expression
$pos$	Position
$N$	Number of subsystems
$d$	Subsystem dimension

**Returns**

Tensor product  $I \otimes \cdots \otimes I \otimes A \otimes I \otimes \cdots \otimes I$ , with  $A$  on position  $pos$ , as a dynamic matrix over the same scalar field as  $A$

**7.14.3.5 Fd()**

```
cmat qpp::Gates::Fd (
    idx D = 2 ) const [inline]
```

Fourier transform gate for qudits.

**Note**

Defined as  $F = \sum_{j,k=0}^{D-1} \exp(2\pi i j k / D) |j\rangle \langle k|$

**Parameters**

$D$	Dimension of the Hilbert space
-----	--------------------------------

**Returns**

Fourier transform gate for qudits

**7.14.3.6 Id()**

```
template<typename Derived = Eigen::MatrixXcd>
Derived qpp::Gates::Id (
    idx D = 2 ) const [inline]
```

Identity gate.

**Note**

Can change the return type from complex matrix (default) by explicitly specifying the template parameter

**Parameters**

$D$	Dimension of the Hilbert space
-----	--------------------------------

**Returns**

Identity gate on a Hilbert space of dimension  $D$

## 7.14.3.7 Rn()

```

cmat qpp::Gates::Rn (
    double theta,
    const std::vector< double > & n ) const [inline]

```

Qubit rotation of *theta* about the 3-dimensional real (unit) vector *n*.

## Parameters

<i>theta</i>	Rotation angle
<i>n</i>	3-dimensional real (unit) vector

## Returns

Rotation gate

## 7.14.3.8 Xd()

```

cmat qpp::Gates::Xd (
    idx D = 2 ) const [inline]

```

Generalized X gate for qudits.

## Note

Defined as  $X = \sum_{j=0}^{D-1} |j \oplus 1\rangle \langle j|$ , i.e. raising operator  $X|j\rangle = |j \oplus 1\rangle$

## Parameters

<i>D</i>	Dimension of the Hilbert space
----------	--------------------------------

## Returns

Generalized X gate for qudits

## 7.14.3.9 Zd()

```

cmat qpp::Gates::Zd (
    idx D = 2 ) const [inline]

```

Generalized Z gate for qudits.

## Note

Defined as  $Z = \sum_{j=0}^{D-1} \exp(2\pi i j/D) |j\rangle \langle j|$

## Parameters

<i>D</i>	Dimension of the Hilbert space
----------	--------------------------------

## Returns

Generalized Z gate for qudits

## 7.14.4 Friends And Related Function Documentation

## 7.14.4.1 internal::Singleton&lt; const Gates &gt;

```
friend class internal::Singleton< const Gates > [friend]
```

## 7.14.5 Member Data Documentation

## 7.14.5.1 CNOT

```
cmat qpp::Gates::CNOT {cmat::Identity(4, 4)}
```

Controlled-NOT control target gate.

## 7.14.5.2 CNOTba

```
cmat qpp::Gates::CNOTba {cmat::Zero(4, 4)}
```

Controlled-NOT target control gate.

## 7.14.5.3 CZ

```
cmat qpp::Gates::CZ {cmat::Identity(4, 4)}
```

Controlled-Phase gate.

#### 7.14.5.4 FRED

```
cmat qpp::Gates::FRED {cmat::Identity(8, 8)}
```

Fredkin gate.

#### 7.14.5.5 H

```
cmat qpp::Gates::H {cmat::Zero(2, 2)}
```

Hadamard gate.

#### 7.14.5.6 Id2

```
cmat qpp::Gates::Id2 {cmat::Identity(2, 2)}
```

Identity gate.

#### 7.14.5.7 S

```
cmat qpp::Gates::S {cmat::Zero(2, 2)}
```

S gate.

#### 7.14.5.8 SWAP

```
cmat qpp::Gates::SWAP {cmat::Identity(4, 4)}
```

SWAP gate.

#### 7.14.5.9 T

```
cmat qpp::Gates::T {cmat::Zero(2, 2)}
```

T gate.



#### 7.14.5.10 TOF

```
cmat qpp::Gates::TOF {cmat::Identity(8, 8)}
```

Toffoli gate.

#### 7.14.5.11 X

```
cmat qpp::Gates::X {cmat::Zero(2, 2)}
```

Pauli Sigma-X gate.

#### 7.14.5.12 Y

```
cmat qpp::Gates::Y {cmat::Zero(2, 2)}
```

Pauli Sigma-Y gate.

#### 7.14.5.13 Z

```
cmat qpp::Gates::Z {cmat::Zero(2, 2)}
```

Pauli Sigma-Z gate.

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

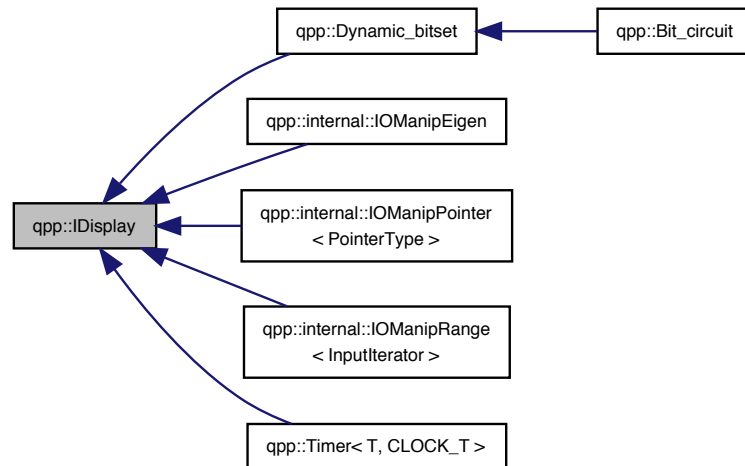
- [classes/gates.h](#)

## 7.15 qpp::IDisplay Class Reference

Abstract class (interface) that mandates the definition of virtual `std::ostream& display(std::ostream& os) const`.

```
#include <classes/ideisplay.h>
```

Inheritance diagram for `qpp::IDisplay`:



### Public Member Functions

- `IDisplay()`=default  
*Default constructor.*
- `IDisplay(const IDisplay &)=default`  
*Default copy constructor.*
- `IDisplay(IDisplay &&)=default`  
*Default move constructor.*
- `IDisplay & operator= (const IDisplay &)=default`  
*Default copy assignment operator.*
- `IDisplay & operator= (IDisplay &&)=default`  
*Default move assignment operator.*
- `virtual ~IDisplay()`=default  
*Default virtual destructor.*

### Private Member Functions

- `virtual std::ostream & display(std::ostream &os) const =0`  
*Must be overridden by all derived classes.*

## Friends

- `std::ostream & operator<< (std::ostream &os, const IDisplay &rhs)`  
*Overloads the extraction operator.*

### 7.15.1 Detailed Description

Abstract class (interface) that mandates the definition of virtual `std::ostream& display(std::ostream& os) const`.

This class defines friend inline `std::ostream& operator<< (std::ostream& os, const qpp::IDisplay& rhs)`. The latter delegates the work to the pure private virtual function `qpp::IDisplay::display()` which has to be overridden by all derived classes.

### 7.15.2 Constructor & Destructor Documentation

#### 7.15.2.1 IDisplay() [1/3]

```
qpp::IDisplay::IDisplay ( ) [default]
```

Default constructor.

#### 7.15.2.2 IDisplay() [2/3]

```
qpp::IDisplay::IDisplay (
    const IDisplay & ) [default]
```

Default copy constructor.

#### 7.15.2.3 IDisplay() [3/3]

```
qpp::IDisplay::IDisplay (
    IDisplay && ) [default]
```

Default move constructor.

#### 7.15.2.4 ~IDisplay()

```
virtual qpp::IDisplay::~IDisplay ( ) [virtual], [default]
```

Default virtual destructor.

## 7.15.3 Member Function Documentation

### 7.15.3.1 display()

```
virtual std::ostream& qpp::IDisplay::display (
    std::ostream & os ) const [private], [pure virtual]
```

Must be overridden by all derived classes.

The actual stream extraction processing is performed by the overridden member function in the derived class. This function is automatically invoked by friend inline `std::ostream& operator<<(std::ostream& os, const IDisplay& rhs)`.

Implemented in [qpp::Dynamic\\_bitset](#), [qpp::internal::IOManipEigen](#), [qpp::Timer< T, CLOCK\\_T >](#), [qpp::internal::IOManipPointer< PointerType >](#), and [qpp::internal::IOManipRange< InputIterator >](#).

### 7.15.3.2 operator=() [1/2]

```
IDisplay& qpp::IDisplay::operator= (
    const IDisplay & ) [default]
```

Default copy assignment operator.

### 7.15.3.3 operator=() [2/2]

```
IDisplay& qpp::IDisplay::operator= (
    IDisplay && ) [default]
```

Default move assignment operator.

## 7.15.4 Friends And Related Function Documentation

### 7.15.4.1 operator<<

```
std::ostream& operator<< (
    std::ostream & os,
    const IDisplay & rhs ) [friend]
```

Overloads the extraction operator.

Delegates the work to the virtual function [qpp::IDisplay::display\(\)](#)

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

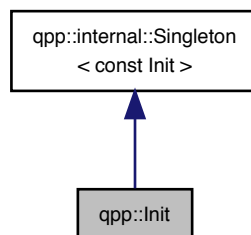
- [classes/ideplay.h](#)

## 7.16 qpp::Init Class Reference

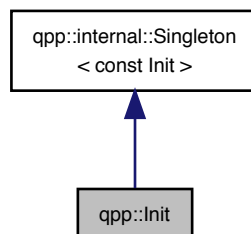
const Singleton class that performs additional initializations/cleanups

```
#include <classes/init.h>
```

Inheritance diagram for qpp::Init:



Collaboration diagram for qpp::Init:



### Private Member Functions

- [Init \(\)](#)  
*Additional initializations.*
- [~Init \(\)](#)  
*Cleanups.*

### Friends

- class [internal::Singleton< const Init >](#)

## Additional Inherited Members

### 7.16.1 Detailed Description

const Singleton class that performs additional initializations/cleanups

### 7.16.2 Constructor & Destructor Documentation

#### 7.16.2.1 Init()

```
qpp::Init::Init ( ) [inline], [private]
```

Additional initializations.

#### 7.16.2.2 ~Init()

```
qpp::Init::~~Init ( ) [inline], [private]
```

Cleanups.

### 7.16.3 Friends And Related Function Documentation

#### 7.16.3.1 internal::Singleton< const Init >

```
friend class internal::Singleton< const Init > [friend]
```

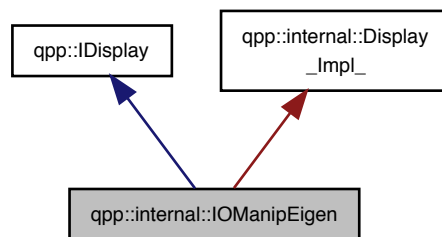
The documentation for this class was generated from the following file:

- [classes/init.h](#)

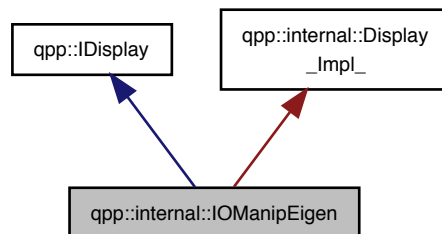
## 7.17 qpp::internal::IOManipEigen Class Reference

```
#include <internal/classes/iomanip.h>
```

Inheritance diagram for qpp::internal::IOManipEigen:



Collaboration diagram for qpp::internal::IOManipEigen:



### Public Member Functions

- `template<typename Derived >`  
`IOManipEigen` (const `Eigen::MatrixBase< Derived >` &A, double `chop`=`qpp::chop`)
- `IOManipEigen` (const `cplx` z, double `chop`=`qpp::chop`)

### Private Member Functions

- `std::ostream & display` (`std::ostream &os`) const override  
*Must be overridden by all derived classes.*

## Private Attributes

- [cmat A\\_](#)
- double [chop\\_](#)

## 7.17.1 Constructor & Destructor Documentation

### 7.17.1.1 IManipEigen() [1/2]

```
template<typename Derived >
qpp::internal::IManipEigen::IManipEigen (
    const Eigen::MatrixBase< Derived > & A,
    double chop = qpp::chop ) [inline], [explicit]
```

### 7.17.1.2 IManipEigen() [2/2]

```
qpp::internal::IManipEigen::IManipEigen (
    const cplx z,
    double chop = qpp::chop ) [inline], [explicit]
```

## 7.17.2 Member Function Documentation

### 7.17.2.1 display()

```
std::ostream& qpp::internal::IManipEigen::display (
    std::ostream & os ) const [inline], [override], [private], [virtual]
```

Must be overridden by all derived classes.

The actual stream extraction processing is performed by the overridden member function in the derived class. This function is automatically invoked by friend inline `std::ostream& operator<<(std::ostream& os, const IDisplay& rhs)`.

Implements [qpp::IDisplay](#).

## 7.17.3 Member Data Documentation



## 7.17.3.1 A\_

```
cmat qpp::internal::IOManipEigen::A_ [private]
```

## 7.17.3.2 chop\_

```
double qpp::internal::IOManipEigen::chop_ [private]
```

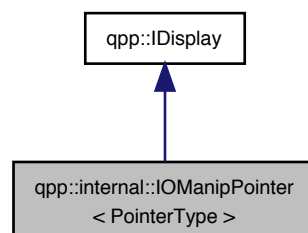
The documentation for this class was generated from the following file:

- [internal/classes/iomanip.h](#)

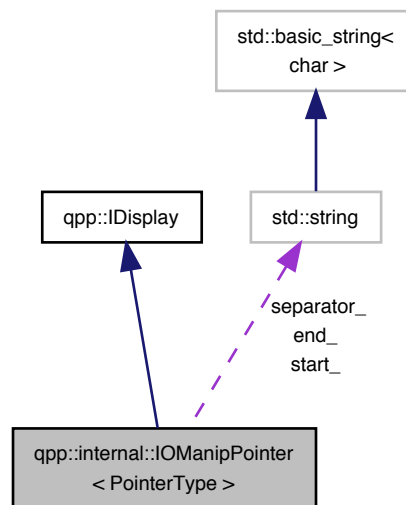
## 7.18 qpp::internal::IOManipPointer&lt; PointerType &gt; Class Template Reference

```
#include <internal/classes/iomanip.h>
```

Inheritance diagram for qpp::internal::IOManipPointer< PointerType >:



Collaboration diagram for `qpp::internal::IOManipPointer< PointerType >`:



## Public Member Functions

- `IOManipPointer` (const PointerType \*p, idx N, const std::string &separator, const std::string &start="[" , const std::string &end="]")
- `IOManipPointer` (const IOManipPointer &)=default
- `IOManipPointer & operator=` (const IOManipPointer &)=default

## Private Member Functions

- `std::ostream & display` (std::ostream &os) const override  
*Must be overridden by all derived classes.*

## Private Attributes

- const PointerType \* `p_`
- idx `N_`
- std::string `separator_`
- std::string `start_`
- std::string `end_`

### 7.18.1 Constructor & Destructor Documentation

## 7.18.1.1 IOManipPointer() [1/2]

```
template<typename PointerType>
qpp::internal::IOManipPointer< PointerType >::IOManipPointer (
    const PointerType * p,
    idx N,
    const std::string & separator,
    const std::string & start = "[",
    const std::string & end = "]" ) [inline], [explicit]
```

## 7.18.1.2 IOManipPointer() [2/2]

```
template<typename PointerType>
qpp::internal::IOManipPointer< PointerType >::IOManipPointer (
    const IOManipPointer< PointerType > & ) [default]
```

## 7.18.2 Member Function Documentation

## 7.18.2.1 display()

```
template<typename PointerType>
std::ostream& qpp::internal::IOManipPointer< PointerType >::display (
    std::ostream & os ) const [inline], [override], [private], [virtual]
```

Must be overridden by all derived classes.

The actual stream extraction processing is performed by the overridden member function in the derived class. This function is automatically invoked by friend inline `std::ostream& operator<<(std::ostream& os, const IDisplay& rhs)`.

Implements [qpp::IDisplay](#).

## 7.18.2.2 operator=()

```
template<typename PointerType>
IOManipPointer& qpp::internal::IOManipPointer< PointerType >::operator= (
    const IOManipPointer< PointerType > & ) [default]
```

## 7.18.3 Member Data Documentation

#### 7.18.3.1 end\_

```
template<typename PointerType>
std::string qpp::internal::IManipPointer< PointerType >::end_ [private]
```

#### 7.18.3.2 N\_

```
template<typename PointerType>
idx qpp::internal::IManipPointer< PointerType >::N_ [private]
```

#### 7.18.3.3 p\_

```
template<typename PointerType>
const PointerType* qpp::internal::IManipPointer< PointerType >::p_ [private]
```

#### 7.18.3.4 separator\_

```
template<typename PointerType>
std::string qpp::internal::IManipPointer< PointerType >::separator_ [private]
```

#### 7.18.3.5 start\_

```
template<typename PointerType>
std::string qpp::internal::IManipPointer< PointerType >::start_ [private]
```

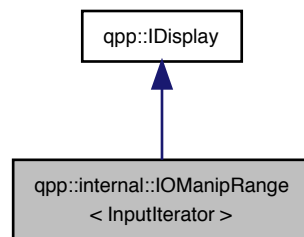
The documentation for this class was generated from the following file:

- [internal/classes/iomanip.h](#)

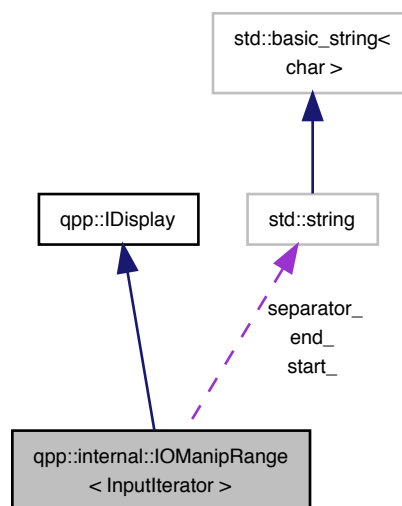
## 7.19 qpp::internal::IOManipRange< InputIterator > Class Template Reference

```
#include <internal/classes/iomanip.h>
```

Inheritance diagram for qpp::internal::IOManipRange< InputIterator >:



Collaboration diagram for qpp::internal::IOManipRange< InputIterator >:



### Public Member Functions

- [IOManipRange](#) (InputIterator first, InputIterator last, const std::string &separator, const std::string &start="[, const std::string &end="]")
- [IOManipRange](#) (const [IOManipRange](#) &)=default
- [IOManipRange](#) & [operator=](#) (const [IOManipRange](#) &)=default

## Private Member Functions

- `std::ostream & display (std::ostream &os)` const override

*Must be overridden by all derived classes.*

## Private Attributes

- InputIterator `first_`
- InputIterator `last_`
- `std::string` `separator_`
- `std::string` `start_`
- `std::string` `end_`

## 7.19.1 Constructor & Destructor Documentation

### 7.19.1.1 IOManipRange() [1/2]

```
template<typename InputIterator>
qpp::internal::IOManipRange< InputIterator >::IOManipRange (
    InputIterator first,
    InputIterator last,
    const std::string & separator,
    const std::string & start = "[",
    const std::string & end = "]" ) [inline], [explicit]
```

### 7.19.1.2 IOManipRange() [2/2]

```
template<typename InputIterator>
qpp::internal::IOManipRange< InputIterator >::IOManipRange (
    const IOManipRange< InputIterator > & ) [default]
```

## 7.19.2 Member Function Documentation

### 7.19.2.1 display()

```
template<typename InputIterator>
std::ostream& qpp::internal::IOManipRange< InputIterator >::display (
    std::ostream & os ) const [inline], [override], [private], [virtual]
```

Must be overridden by all derived classes.

The actual stream extraction processing is performed by the overridden member function in the derived class. This function is automatically invoked by friend inline `std::ostream& operator<<(std::ostream& os, const IDisplay& rhs)`.

Implements `qpp::IDisplay`.

### 7.19.2.2 operator=()

```
template<typename InputIterator>
IOManipRange& qpp::internal::IOManipRange< InputIterator >::operator= (
    const IOManipRange< InputIterator > & ) [default]
```

## 7.19.3 Member Data Documentation

### 7.19.3.1 end\_

```
template<typename InputIterator>
std::string qpp::internal::IOManipRange< InputIterator >::end_ [private]
```

### 7.19.3.2 first\_

```
template<typename InputIterator>
InputIterator qpp::internal::IOManipRange< InputIterator >::first_ [private]
```

### 7.19.3.3 last\_

```
template<typename InputIterator>
InputIterator qpp::internal::IOManipRange< InputIterator >::last_ [private]
```

### 7.19.3.4 separator\_

```
template<typename InputIterator>
std::string qpp::internal::IOManipRange< InputIterator >::separator_ [private]
```

### 7.19.3.5 start\_

```
template<typename InputIterator>
std::string qpp::internal::IOManipRange< InputIterator >::start_ [private]
```

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

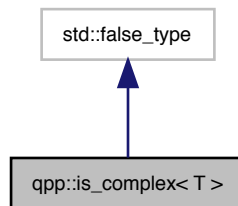
- [internal/classes/iomanip.h](#)

## 7.20 qpp::is\_complex< T > Struct Template Reference

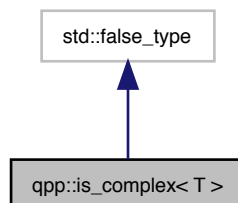
Checks whether the type is a complex type.

```
#include <traits.h>
```

Inheritance diagram for qpp::is\_complex< T >:



Collaboration diagram for qpp::is\_complex< T >:



### 7.20.1 Detailed Description

```
template<typename T>  
struct qpp::is_complex< T >
```

Checks whether the type is a complex type.

Provides the constant member *value* which is equal to *true*, if the type is a complex type, i.e. *std::complex< T >*

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

- [traits.h](#)

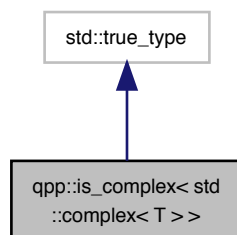


## 7.21 qpp::is\_complex< std::complex< T > > Struct Template Reference

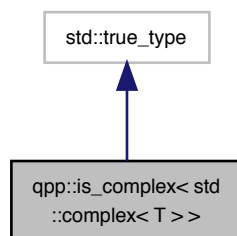
Checks whether the type is a complex number type, specialization for complex types.

```
#include <traits.h>
```

Inheritance diagram for qpp::is\_complex< std::complex< T > >:



Collaboration diagram for qpp::is\_complex< std::complex< T > >:



### 7.21.1 Detailed Description

```
template<typename T>
struct qpp::is_complex< std::complex< T > >
```

Checks whether the type is a complex number type, specialization for complex types.

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

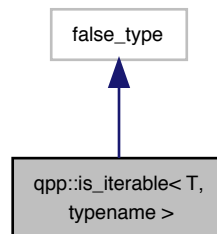
- [traits.h](#)

## 7.22 qpp::is\_iterable< T, typename > Struct Template Reference

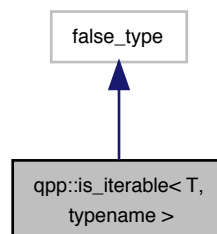
Checks whether *T* is compatible with an STL-like iterable container.

```
#include <traits.h>
```

Inheritance diagram for qpp::is\_iterable< T, typename >:



Collaboration diagram for qpp::is\_iterable< T, typename >:



### 7.22.1 Detailed Description

```
template<typename T, typename = void>
struct qpp::is_iterable< T, typename >
```

Checks whether *T* is compatible with an STL-like iterable container.

Provides the constant member *value* which is equal to *true*, if *T* is compatible with an iterable container, i.e. provides at least *begin()* and *end()* member functions. Otherwise, *value* is equal to *false*.

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

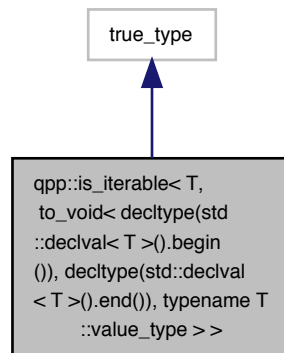
- [traits.h](#)

## 7.23 `qpp::is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), typename T::value_type > >` Struct Template Reference

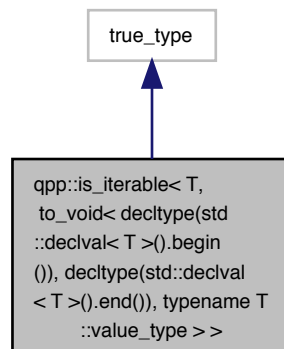
Checks whether *T* is compatible with an STL-like iterable container, specialization for STL-like iterable containers.

```
#include <traits.h>
```

Inheritance diagram for `qpp::is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), typename T::value_type > >`:



Collaboration diagram for `qpp::is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), typename T::value_type > >`:



### 7.23.1 Detailed Description

```
template<typename T>
struct qpp::is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), typename T::value_type > >
```

Checks whether *T* is compatible with an STL-like iterable container, specialization for STL-like iterable containers.

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

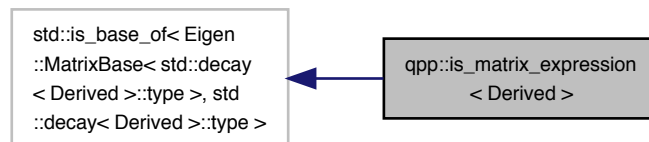
- [traits.h](#)

## 7.24 qpp::is\_matrix\_expression< Derived > Struct Template Reference

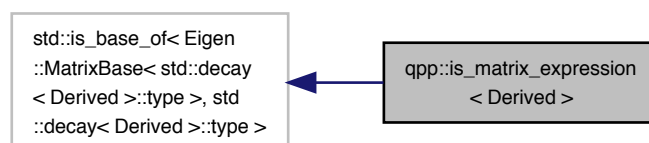
Checks whether the type is an Eigen matrix expression.

```
#include <traits.h>
```

Inheritance diagram for qpp::is\_matrix\_expression< Derived >:



Collaboration diagram for qpp::is\_matrix\_expression< Derived >:



### 7.24.1 Detailed Description

```
template<typename Derived>
struct qpp::is_matrix_expression< Derived >
```

Checks whether the type is an Eigen matrix expression.

Provides the constant member *value* which is equal to *true*, if the type is an Eigen matrix expression of type *EigenMatrixBase<Derived>*. Otherwise, *value* is equal to *false*.

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

- [traits.h](#)

## 7.25 `qpp::make_void< Ts >` Struct Template Reference

Helper for [qpp::to\\_void<>](#) alias template.

```
#include <traits.h>
```

### Public Types

- typedef void [type](#)

### 7.25.1 Detailed Description

```
template<typename... Ts>
struct qpp::make_void< Ts >
```

Helper for [qpp::to\\_void<>](#) alias template.

See also

[qpp::to\\_void<>](#)

### 7.25.2 Member Typedef Documentation

#### 7.25.2.1 `type`

```
template<typename... Ts>
typedef void qpp::make\_void< Ts >::type
```

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

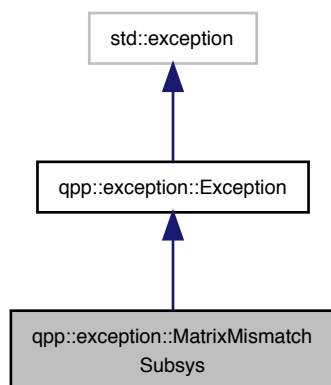
- [traits.h](#)

## 7.26 qpp::exception::MatrixMismatchSubsys Class Reference

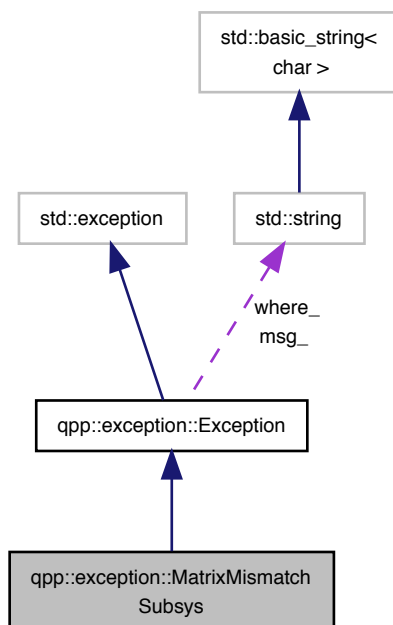
Matrix mismatch subsystems exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::MatrixMismatchSubsys:



Collaboration diagram for qpp::exception::MatrixMismatchSubsys:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.26.1 Detailed Description

Matrix mismatch subsystems exception.

Matrix size mismatch subsystem sizes (e.g. in `qpp::apply()`)

### 7.26.2 Member Function Documentation

#### 7.26.2.1 type\_description()

```
std::string qpp::exception::MatrixMismatchSubsys::type_description ( ) const [inline], [override],  
[virtual]
```

*Exception* type description.

#### Returns

*Exception* type description

Implements `qpp::exception::Exception`.

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

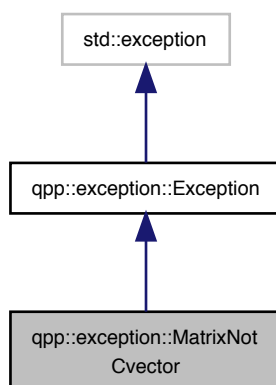
- `classes/exception.h`

## 7.27 qpp::exception::MatrixNotCvector Class Reference

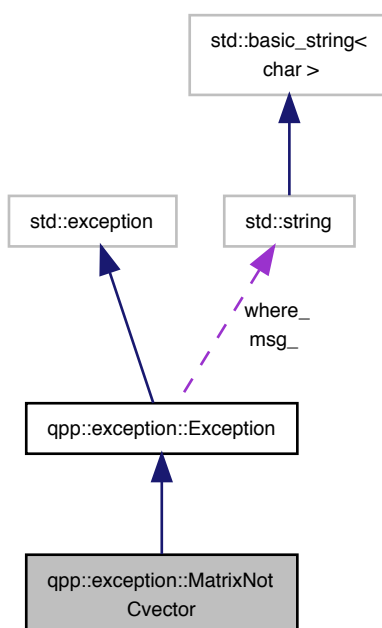
Matrix is not a column vector exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::MatrixNotCvector:



Collaboration diagram for qpp::exception::MatrixNotCvector:



## Public Member Functions

- `std::string` [type\\_description](#) () const override  
*[Exception](#) type description.*



### 7.27.1 Detailed Description

Matrix is not a column vector exception.

Eigen::Matrix is not a column vector

### 7.27.2 Member Function Documentation

#### 7.27.2.1 type\_description()

```
std::string qpp::exception::MatrixNotCvector::type_description ( ) const [inline], [override],  
[virtual]
```

Exception type description.

#### Returns

Exception type description

Implements [qpp::exception::Exception](#).

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

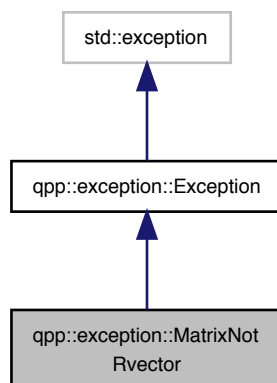
- [classes/exception.h](#)

## 7.28 qpp::exception::MatrixNotRvector Class Reference

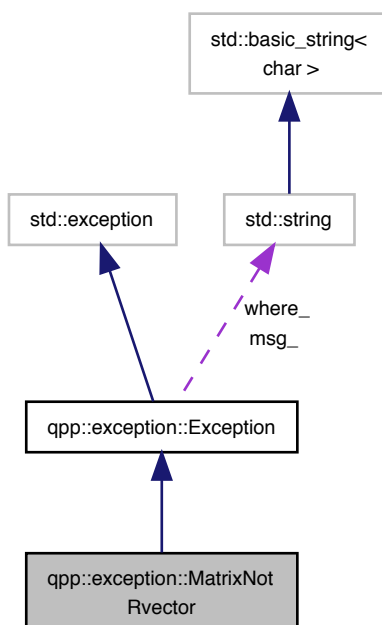
Matrix is not a row vector exception.

```
#include <classes/exception.h>
```

Inheritance diagram for `qpp::exception::MatrixNotRvector`:



Collaboration diagram for `qpp::exception::MatrixNotRvector`:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.28.1 Detailed Description

Matrix is not a row vector exception.

Eigen::Matrix is not a row vector

### 7.28.2 Member Function Documentation

## 7.28.2.1 type\_description()

```
std::string qpp::exception::MatrixNotRvector::type_description ( ) const [inline], [override],  
[virtual]
```

Exception type description.

## Returns

Exception type description

Implements [qpp::exception::Exception](#).

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

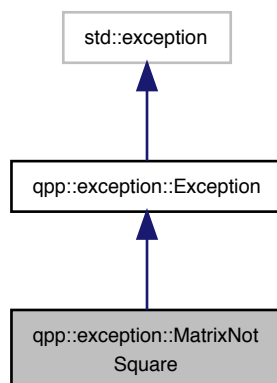
- [classes/exception.h](#)

## 7.29 qpp::exception::MatrixNotSquare Class Reference

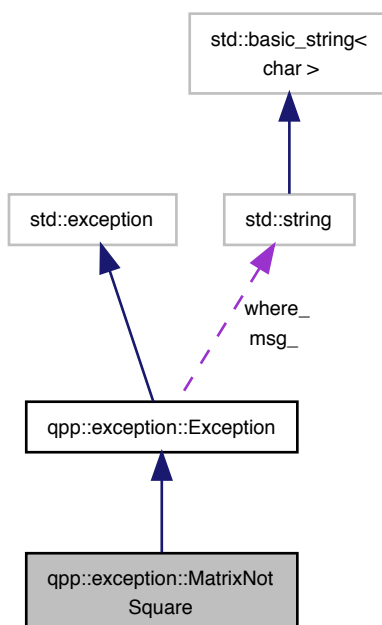
Matrix is not square exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::MatrixNotSquare:



Collaboration diagram for qpp::exception::MatrixNotSquare:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.29.1 Detailed Description

Matrix is not square exception.

Eigen::Matrix is not a square matrix

### 7.29.2 Member Function Documentation

## 7.29.2.1 type\_description()

```
std::string qpp::exception::MatrixNotSquare::type_description ( ) const [inline], [override],  
[virtual]
```

Exception type description.

## Returns

Exception type description

Implements [qpp::exception::Exception](#).

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

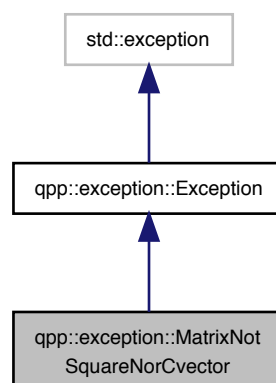
- [classes/exception.h](#)

## 7.30 qpp::exception::MatrixNotSquareNorCvector Class Reference

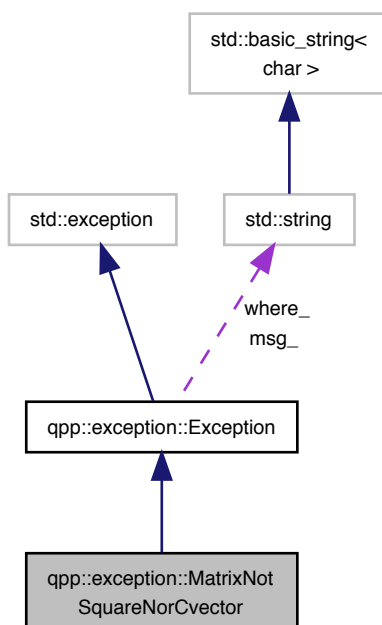
Matrix is not square nor column vector exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::MatrixNotSquareNorCvector:



Collaboration diagram for `qpp::exception::MatrixNotSquareNorCvector`:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.30.1 Detailed Description

Matrix is not square nor column vector exception.

Eigen::Matrix is not a square matrix nor a column vector

### 7.30.2 Member Function Documentation

## 7.30.2.1 type\_description()

```
std::string qpp::exception::MatrixNotSquareNorCvector::type_description ( ) const [inline],  
[override], [virtual]
```

Exception type description.

## Returns

Exception type description

Implements [qpp::exception::Exception](#).

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

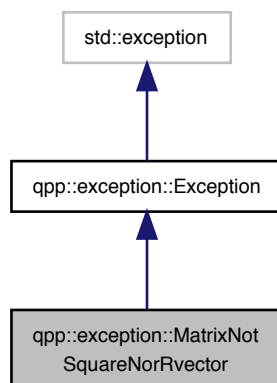
- [classes/exception.h](#)

## 7.31 qpp::exception::MatrixNotSquareNorRvector Class Reference

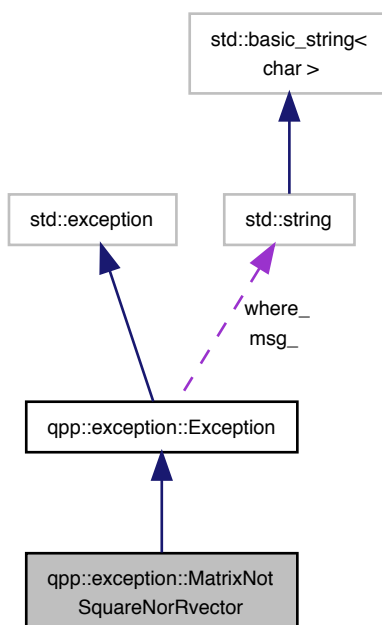
Matrix is not square nor row vector exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::MatrixNotSquareNorRvector:



Collaboration diagram for `qpp::exception::MatrixNotSquareNorRvector`:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.31.1 Detailed Description

Matrix is not square nor row vector exception.

Eigen::Matrix is not a square matrix nor a row vector

### 7.31.2 Member Function Documentation



## 7.31.2.1 type\_description()

```
std::string qpp::exception::MatrixNotSquareNorRvector::type_description ( ) const [inline],  
[override], [virtual]
```

Exception type description.

## Returns

Exception type description

Implements [qpp::exception::Exception](#).

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

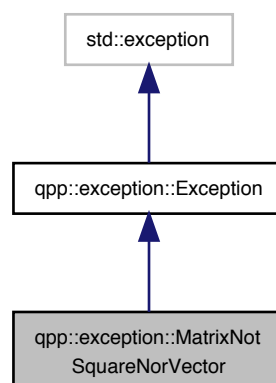
- [classes/exception.h](#)

## 7.32 qpp::exception::MatrixNotSquareNorVector Class Reference

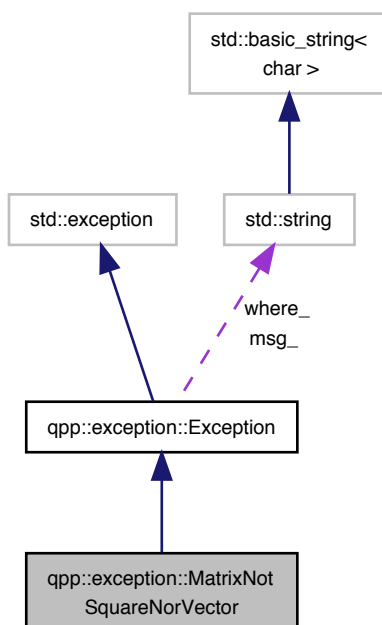
Matrix is not square nor vector exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::MatrixNotSquareNorVector:



Collaboration diagram for qpp::exception::MatrixNotSquareNorVector:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.32.1 Detailed Description

Matrix is not square nor vector exception.

Eigen::Matrix is not a square matrix nor a row/column vector

### 7.32.2 Member Function Documentation

## 7.32.2.1 type\_description()

```
std::string qpp::exception::MatrixNotSquareNorVector::type_description ( ) const [inline],  
[override], [virtual]
```

Exception type description.

## Returns

Exception type description

Implements [qpp::exception::Exception](#).

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

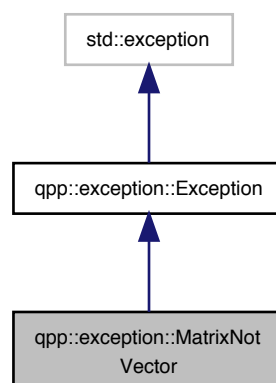
- [classes/exception.h](#)

## 7.33 qpp::exception::MatrixNotVector Class Reference

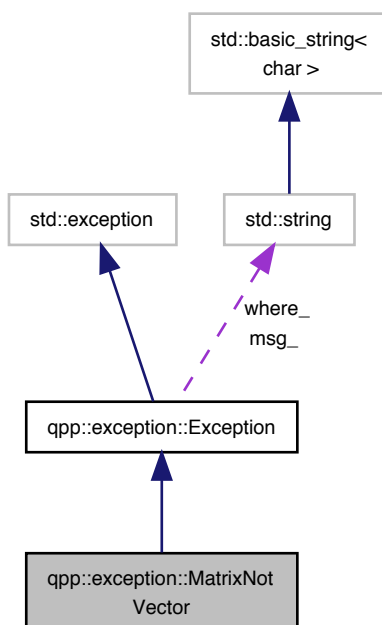
Matrix is not a vector exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::MatrixNotVector:



Collaboration diagram for qpp::exception::MatrixNotVector:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.33.1 Detailed Description

Matrix is not a vector exception.

Eigen::Matrix is not a row or column vector

### 7.33.2 Member Function Documentation

## 7.33.2.1 type\_description()

```
std::string qpp::exception::MatrixNotVector::type_description ( ) const [inline], [override],  
[virtual]
```

Exception type description.

## Returns

Exception type description

Implements [qpp::exception::Exception](#).

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

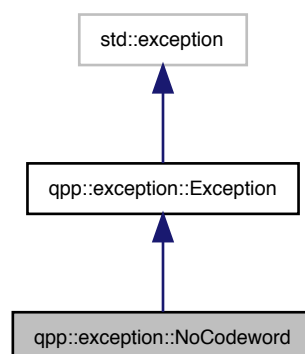
- [classes/exception.h](#)

## 7.34 qpp::exception::NoCodeword Class Reference

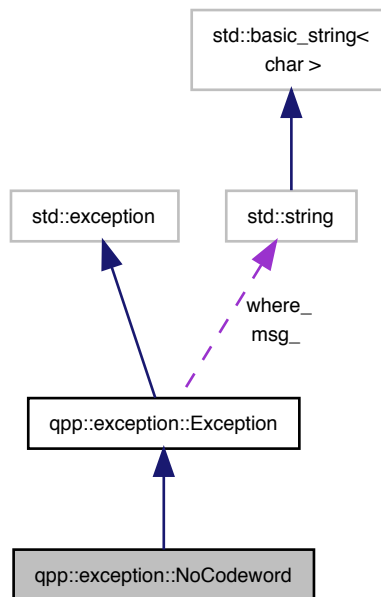
Codeword does not exist exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::NoCodeword:



Collaboration diagram for `qpp::exception::NoCodeword`:



## Public Member Functions

- `std::string type_description () const` override  
*Exception type description.*

### 7.34.1 Detailed Description

Codeword does not exist exception.

Codeword does not exist, thrown when calling `qpp::Codes::codeword()` with an invalid index

### 7.34.2 Member Function Documentation

#### 7.34.2.1 type\_description()

```
std::string qpp::exception::NoCodeword::type_description ( ) const [inline], [override],
[virtual]
```

*Exception type description.*

## Returns

[Exception](#) type description

Implements [qpp::exception::Exception](#).

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

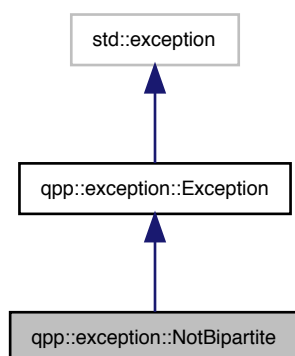
- [classes/exception.h](#)

## 7.35 qpp::exception::NotBipartite Class Reference

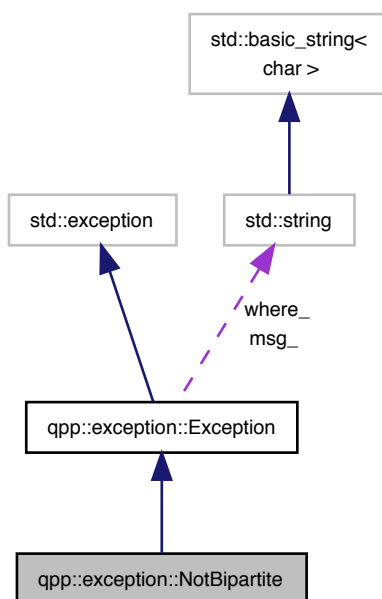
Not bi-partite exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::NotBipartite:



Collaboration diagram for qpp::exception::NotBipartite:



## Public Member Functions

- `std::string type_description () const` override  
*Exception type description.*

### 7.35.1 Detailed Description

Not bi-partite exception.

`std::vector<idx>` of dimensions has size different from 2

### 7.35.2 Member Function Documentation

#### 7.35.2.1 type\_description()

```
std::string qpp::exception::NotBipartite::type_description ( ) const [inline], [override],
[virtual]
```

*Exception type description.*



## Returns

[Exception](#) type description

Implements [qpp::exception::Exception](#).

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

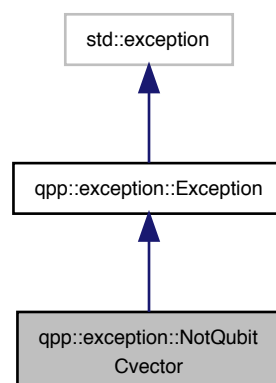
- [classes/exception.h](#)

## 7.36 qpp::exception::NotQubitCvector Class Reference

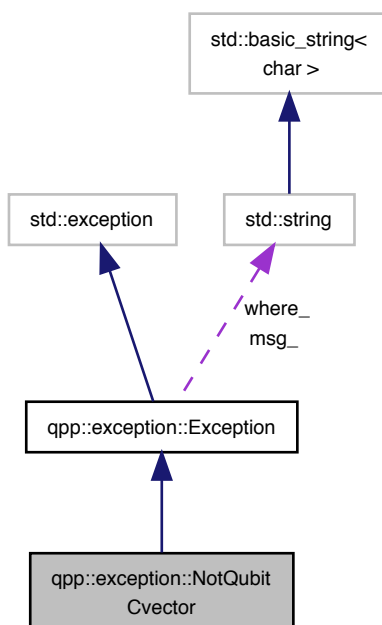
Column vector is not 2 x 1 exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::NotQubitCvector:



Collaboration diagram for `qpp::exception::NotQubitCvector`:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.36.1 Detailed Description

Column vector is not 2 x 1 exception.

Eigen::Matrix is not 2 x 1

### 7.36.2 Member Function Documentation

## 7.36.2.1 type\_description()

```
std::string qpp::exception::NotQubitCvector::type_description ( ) const [inline], [override],  
[virtual]
```

Exception type description.

## Returns

Exception type description

Implements [qpp::exception::Exception](#).

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

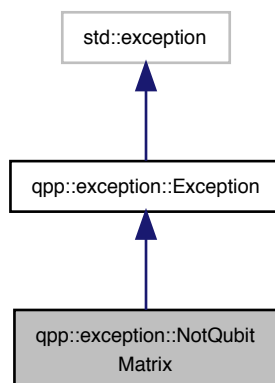
- [classes/exception.h](#)

## 7.37 qpp::exception::NotQubitMatrix Class Reference

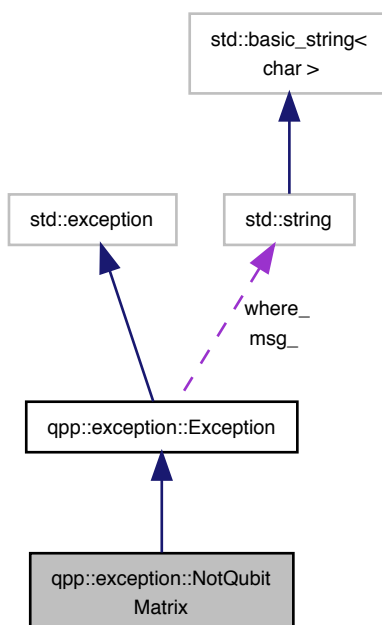
Matrix is not 2 x 2 exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::NotQubitMatrix:



Collaboration diagram for qpp::exception::NotQubitMatrix:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.37.1 Detailed Description

Matrix is not 2 x 2 exception.

Eigen::Matrix is not 2 x 2

### 7.37.2 Member Function Documentation

## 7.37.2.1 type\_description()

```
std::string qpp::exception::NotQubitMatrix::type_description ( ) const [inline], [override],  
[virtual]
```

Exception type description.

## Returns

Exception type description

Implements [qpp::exception::Exception](#).

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

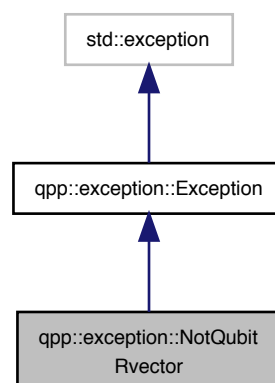
- [classes/exception.h](#)

## 7.38 qpp::exception::NotQubitRvector Class Reference

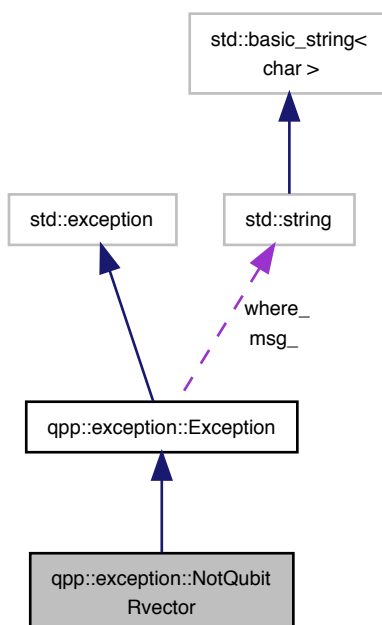
Row vector is not 1 x 2 exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::NotQubitRvector:



Collaboration diagram for `qpp::exception::NotQubitRvector`:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.38.1 Detailed Description

Row vector is not 1 x 2 exception.

Eigen::Matrix is not 1 x 2

### 7.38.2 Member Function Documentation

## 7.38.2.1 type\_description()

```
std::string qpp::exception::NotQubitRvector::type_description ( ) const [inline], [override],  
[virtual]
```

Exception type description.

## Returns

Exception type description

Implements [qpp::exception::Exception](#).

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

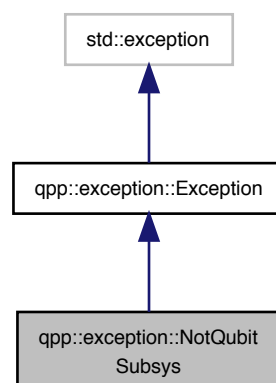
- [classes/exception.h](#)

## 7.39 qpp::exception::NotQubitSubsys Class Reference

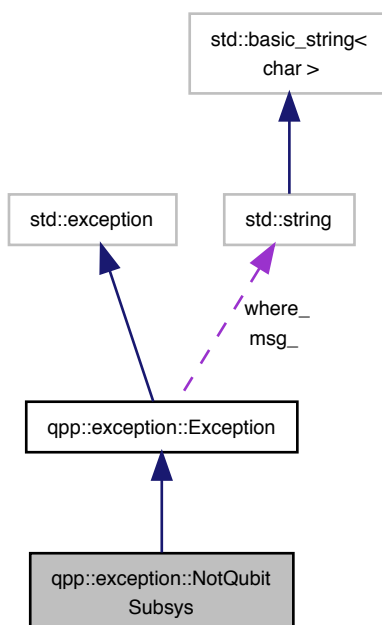
Subsystems are not qubits exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::NotQubitSubsys:



Collaboration diagram for qpp::exception::NotQubitSubsys:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.39.1 Detailed Description

Subsystems are not qubits exception.

Subsystems are not 2-dimensional (qubits)

### 7.39.2 Member Function Documentation



## 7.39.2.1 type\_description()

```
std::string qpp::exception::NotQubitSubsys::type_description ( ) const [inline], [override],  
[virtual]
```

Exception type description.

## Returns

Exception type description

Implements [qpp::exception::Exception](#).

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

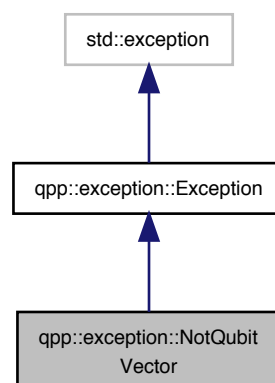
- [classes/exception.h](#)

## 7.40 qpp::exception::NotQubitVector Class Reference

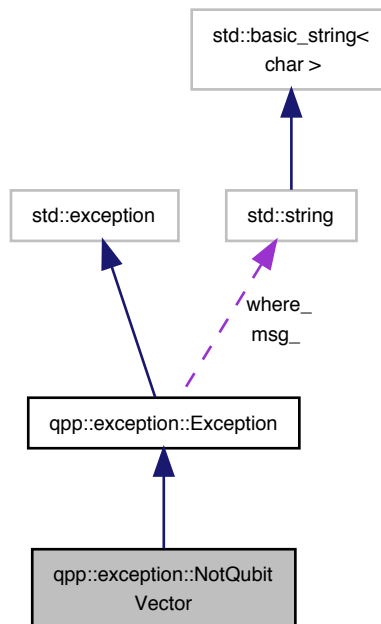
Vector is not 2 x 1 nor 1 x 2 exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::NotQubitVector:



Collaboration diagram for qpp::exception::NotQubitVector:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.40.1 Detailed Description

Vector is not 2 x 1 nor 1 x 2 exception.

Eigen::Matrix is not 2 x 1 nor 1 x 2

### 7.40.2 Member Function Documentation

## 7.40.2.1 type\_description()

```
std::string qpp::exception::NotQubitVector::type_description ( ) const [inline], [override],  
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements [qpp::exception::Exception](#).

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

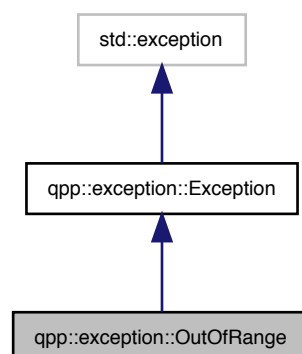
- [classes/exception.h](#)

## 7.41 qpp::exception::OutOfRange Class Reference

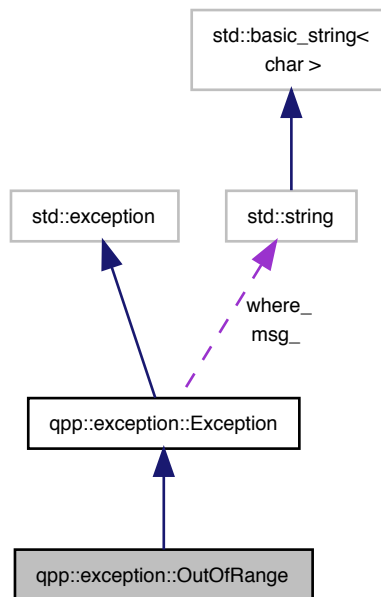
Parameter out of range exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::OutOfRange:



Collaboration diagram for `qpp::exception::OutOfRange`:



## Public Member Functions

- `std::string type_description () const` override  
*Exception type description.*

### 7.41.1 Detailed Description

Parameter out of range exception.

Parameter out of range

### 7.41.2 Member Function Documentation

#### 7.41.2.1 type\_description()

```
std::string qpp::exception::OutOfRange::type_description ( ) const [inline], [override],
[virtual]
```

*Exception type description.*

## Returns

[Exception](#) type description

Implements [qpp::exception::Exception](#).

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

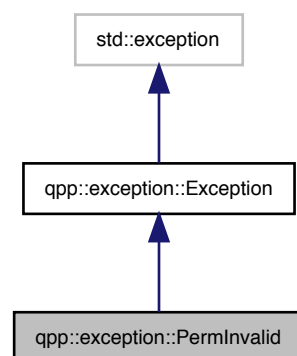
- [classes/exception.h](#)

## 7.42 qpp::exception::PermInvalid Class Reference

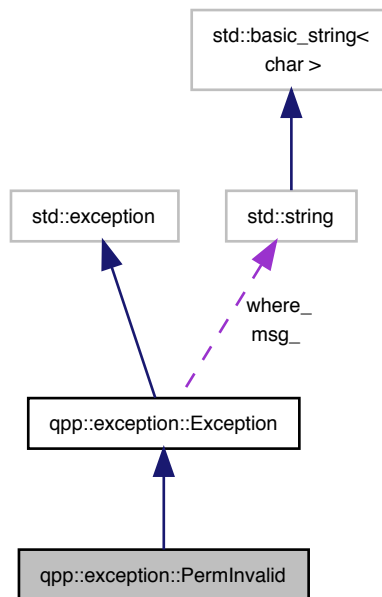
Invalid permutation exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::PermInvalid:



Collaboration diagram for qpp::exception::PermInvalid:



## Public Member Functions

- `std::string type_description () const` override  
*Exception type description.*

### 7.42.1 Detailed Description

Invalid permutation exception.

`std::vector<idx>` does not represent a valid permutation

### 7.42.2 Member Function Documentation

#### 7.42.2.1 type\_description()

```
std::string qpp::exception::PermInvalid::type_description ( ) const [inline], [override],
[virtual]
```

*Exception type description.*

## Returns

[Exception](#) type description

Implements [qpp::exception::Exception](#).

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

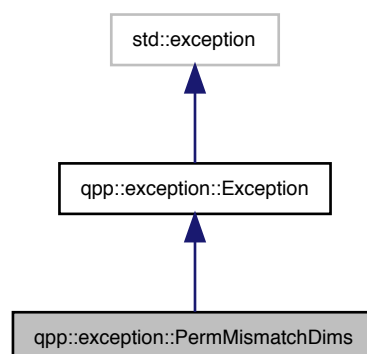
- [classes/exception.h](#)

## 7.43 qpp::exception::PermMismatchDims Class Reference

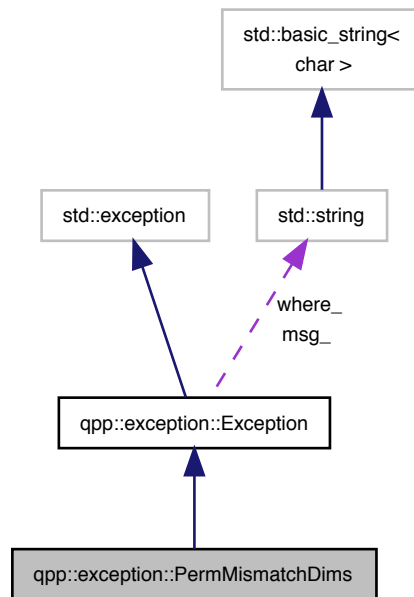
Permutation mismatch dimensions exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::PermMismatchDims:



Collaboration diagram for `qpp::exception::PermMismatchDims`:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.43.1 Detailed Description

Permutation mismatch dimensions exception.

Size of the `std::vector<idx>` representing the permutation is different from the size of the `std::vector<idx>` of dimensions

### 7.43.2 Member Function Documentation



## 7.43.2.1 type\_description()

```
std::string qpp::exception::PermMismatchDims::type_description ( ) const [inline], [override],  
[virtual]
```

Exception type description.

## Returns

Exception type description

Implements [qpp::exception::Exception](#).

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

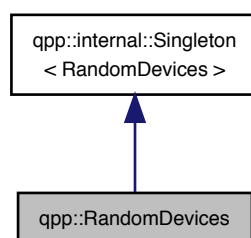
- [classes/exception.h](#)

## 7.44 qpp::RandomDevices Class Reference

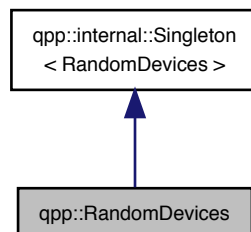
Singleton class that manages the source of randomness in the library.

```
#include <classes/random_devices.h>
```

Inheritance diagram for `qpp::RandomDevices`:



Collaboration diagram for `qpp::RandomDevices`:



## Public Member Functions

- `std::mt19937 & get_prng ()`  
*Returns a reference to the internal PRNG object.*
- `std::istream & load (std::istream &is)`  
*Loads the state of the PRNG from an input stream.*
- `std::ostream & save (std::ostream &os) const`  
*Saves the state of the PRNG to an output stream.*

## Private Member Functions

- `RandomDevices ()`  
*Initializes and seeds the random number generators.*
- `~RandomDevices ()=default`  
*Default destructor.*

## Private Attributes

- `std::random_device rd_`  
*used to seed std::mt19937 prng\_*
- `std::mt19937 prng_`  
*Mersenne twister random number generator.*

## Friends

- class `internal::Singleton< RandomDevices >`

## Additional Inherited Members

### 7.44.1 Detailed Description

Singleton class that manages the source of randomness in the library.

Consists of a wrapper around an `std::mt19937` Mersenne twister random number generator engine and an `std::random_device` engine. The latter is used to seed the Mersenne twister.

#### Warning

This class DOES NOT seed the standard C number generator used by `Eigen::Matrix::Random()`, since it is not thread safe. Do not use `Eigen::Matrix::Random()` or functions that depend on the C style random number engine, but use `qpp::rand()` instead!

### 7.44.2 Constructor & Destructor Documentation

#### 7.44.2.1 RandomDevices()

```
qpp::RandomDevices::RandomDevices ( ) [inline], [private]
```

Initializes and seeds the random number generators.

#### 7.44.2.2 ~RandomDevices()

```
qpp::RandomDevices::~~RandomDevices ( ) [private], [default]
```

Default destructor.

### 7.44.3 Member Function Documentation

#### 7.44.3.1 get\_prng()

```
std::mt19937& qpp::RandomDevices::get_prng ( ) [inline]
```

Returns a reference to the internal PRNG object.

##### Returns

Reference to the internal PRNG object

#### 7.44.3.2 load()

```
std::istream& qpp::RandomDevices::load (
    std::istream & is ) [inline]
```

Loads the state of the PRNG from an input stream.

##### Parameters

<i>is</i>	Input stream
-----------	--------------

##### Returns

The input stream

#### 7.44.3.3 save()

```
std::ostream& qpp::RandomDevices::save (
    std::ostream & os ) const [inline]
```

Saves the state of the PRNG to an output stream.

##### Parameters

<i>os</i>	Output stream
-----------	---------------

##### Returns

The output stream

### 7.44.4 Friends And Related Function Documentation

#### 7.44.4.1 internal::Singleton< RandomDevices >

```
friend class internal::Singleton< RandomDevices > [friend]
```

### 7.44.5 Member Data Documentation

#### 7.44.5.1 prng\_

```
std::mt19937 qpp::RandomDevices::prng_ [private]
```

Mersenne twister random number generator.

#### 7.44.5.2 rd\_

```
std::random_device qpp::RandomDevices::rd_ [private]
```

used to seed std::mt19937 prng\_

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

- [classes/random\\_devices.h](#)

## 7.45 qpp::internal::Singleton< T > Class Template Reference

[Singleton](#) policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern)

```
#include <internal/classes/singleton.h>
```

### Static Public Member Functions

- static T & [get\\_instance](#) () noexcept(std::is\_nothrow\_constructible< T >::value)
- static T & [get\\_thread\\_local\\_instance](#) () noexcept(std::is\_nothrow\_constructible< T >::value)

### Protected Member Functions

- [Singleton](#) () noexcept=default
- [Singleton](#) (const [Singleton](#) &)=delete
- [Singleton](#) & [operator=](#) (const [Singleton](#) &)=delete
- virtual [~Singleton](#) ()=default

#### 7.45.1 Detailed Description

```
template<typename T>
class qpp::internal::Singleton< T >
```

[Singleton](#) policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern)

To implement a singleton, derive your class from [qpp::internal::Singleton](#), make [qpp::internal::Singleton](#) a friend of your class, then declare the constructor and destructor of your class as private. To get an instance, use the static member function [qpp::internal::Singleton::get\\_instance\(\)](#) ([qpp::internal::Singleton::get\\_thread\\_local\\_instance\(\)](#)), which returns a reference (thread\_local reference) to your newly created singleton (thread-safe in C++11).

Example:

```
class MySingleton: public qpp::internal::Singleton<MySingleton>
{
    friend class qpp::internal::Singleton<MySingleton>;
public:
    // Declare all public members here
private:
    MySingleton()
    {
        // Implement the constructor here
    }
    ~MySingleton()
    {
        // Implement the destructor here
    }
};

MySingleton& mySingleton = MySingleton::get_instance(); // Get an instance
thread_local MySingleton& tls = MySingleton::get_thread_local_instance();
// Get a thread_local instance
```

### See also

Code of [qpp::Codes](#), [qpp::Gates](#), [qpp::Init](#), [qpp::RandomDevices](#), [qpp::States](#) or [qpp.h](#) for real world examples of usage.

## 7.45.2 Constructor & Destructor Documentation

### 7.45.2.1 Singleton() [1/2]

```
template<typename T>
qpp::internal::Singleton< T >::Singleton ( ) [protected], [default], [noexcept]
```

### 7.45.2.2 Singleton() [2/2]

```
template<typename T>
qpp::internal::Singleton< T >::Singleton (
    const Singleton< T > & ) [protected], [delete]
```

### 7.45.2.3 ~Singleton()

```
template<typename T>
virtual qpp::internal::Singleton< T >::~~Singleton ( ) [protected], [virtual], [default]
```

## 7.45.3 Member Function Documentation

### 7.45.3.1 get\_instance()

```
template<typename T>
static T& qpp::internal::Singleton< T >::get_instance ( ) [inline], [static], [noexcept]
```

### 7.45.3.2 get\_thread\_local\_instance()

```
template<typename T>
static T& qpp::internal::Singleton< T >::get_thread_local_instance ( ) [inline], [static],
[noexcept]
```

## 7.45.3.3 operator=()

```
template<typename T>
Singleton& qpp::internal::Singleton< T >::operator= (
    const Singleton< T > & ) [protected], [delete]
```

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

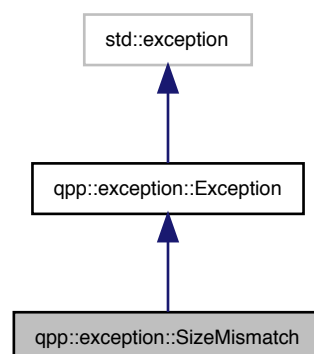
- [internal/classes/singleton.h](#)

## 7.46 qpp::exception::SizeMismatch Class Reference

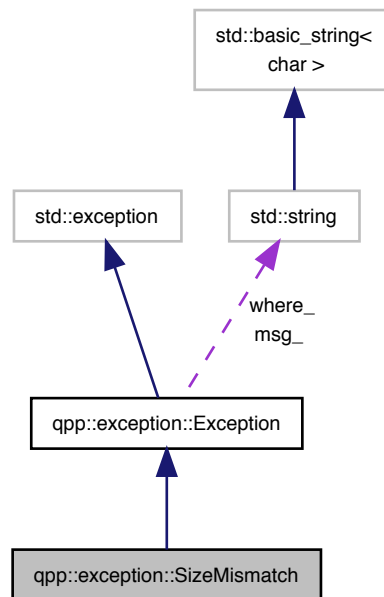
Size mismatch exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::SizeMismatch:



Collaboration diagram for `qpp::exception::SizeMismatch`:



## Public Member Functions

- `std::string type_description () const` override  
*Exception type description.*

### 7.46.1 Detailed Description

Size mismatch exception.

Sizes do not match

### 7.46.2 Member Function Documentation

#### 7.46.2.1 type\_description()

```
std::string qpp::exception::SizeMismatch::type_description ( ) const [inline], [override],
[virtual]
```

*Exception type description.*



## Returns

[Exception](#) type description

Implements [qpp::exception::Exception](#).

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

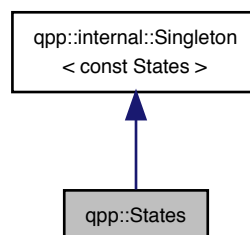
- [classes/exception.h](#)

## 7.47 qpp::States Class Reference

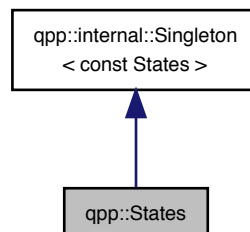
const Singleton class that implements most commonly used states

```
#include <classes/states.h>
```

Inheritance diagram for qpp::States:



Collaboration diagram for qpp::States:



## Public Member Functions

- `ket mes (idx d=2) const`  
*Maximally entangled state of 2 qudits.*
- `ket zero (idx n, idx d=2) const`  
*Zero state of n qudits.*
- `ket one (idx n, idx d=2) const`  
*One state of n qudits.*
- `ket jn (idx j, idx n, idx d=2) const`  
 $|j\rangle^{\otimes n}$  *state of n qudits*
- `ket plus (idx n) const`  
*Plus state of n qubits.*
- `ket minus (idx n) const`  
*Minus state of n qubits.*

## Public Attributes

- `ket x0 {ket::Zero(2)}`  
*Pauli Sigma-X 0-eigenstate  $|+\rangle$*
- `ket x1 {ket::Zero(2)}`  
*Pauli Sigma-X 1-eigenstate  $|-\rangle$*
- `ket y0 {ket::Zero(2)}`  
*Pauli Sigma-Y 0-eigenstate  $|y+\rangle$*
- `ket y1 {ket::Zero(2)}`  
*Pauli Sigma-Y 1-eigenstate  $|y-\rangle$*
- `ket z0 {ket::Zero(2)}`  
*Pauli Sigma-Z 0-eigenstate  $|0\rangle$*
- `ket z1 {ket::Zero(2)}`  
*Pauli Sigma-Z 1-eigenstate  $|1\rangle$*
- `cmat px0 {cmat::Zero(2, 2)}`  
*Projector onto the Pauli Sigma-X 0-eigenstate  $|+\rangle\langle+|$ .*
- `cmat px1 {cmat::Zero(2, 2)}`  
*Projector onto the Pauli Sigma-X 1-eigenstate  $|-\rangle\langle-|$ .*
- `cmat py0 {cmat::Zero(2, 2)}`  
*Projector onto the Pauli Sigma-Y 0-eigenstate  $|y+\rangle\langle y+|$ .*
- `cmat py1 {cmat::Zero(2, 2)}`  
*Projector onto the Pauli Sigma-Y 1-eigenstate  $|y-\rangle\langle y-|$ .*
- `cmat pz0 {cmat::Zero(2, 2)}`  
*Projector onto the Pauli Sigma-Z 0-eigenstate  $|0\rangle\langle 0|$ .*
- `cmat pz1 {cmat::Zero(2, 2)}`  
*Projector onto the Pauli Sigma-Z 1-eigenstate  $|1\rangle\langle 1|$ .*
- `ket b00 {ket::Zero(4)}`  
*Bell-00 state (following the convention in Nielsen and Chuang)*
- `ket b01 {ket::Zero(4)}`  
*Bell-01 state (following the convention in Nielsen and Chuang)*
- `ket b10 {ket::Zero(4)}`  
*Bell-10 state (following the convention in Nielsen and Chuang)*
- `ket b11 {ket::Zero(4)}`  
*Bell-11 state (following the convention in Nielsen and Chuang)*
- `cmat pb00 {cmat::Zero(4, 4)}`

- Projector onto the Bell-00 state.*
- [cmat pb01](#) {cmat::Zero(4, 4)}
- Projector onto the Bell-01 state.*
- [cmat pb10](#) {cmat::Zero(4, 4)}
- Projector onto the Bell-10 state.*
- [cmat pb11](#) {cmat::Zero(4, 4)}
- Projector onto the Bell-11 state.*
- [ket GHZ](#) {ket::Zero(8)}
- GHZ state.*
- [ket W](#) {ket::Zero(8)}
- W state.*
- [cmat pGHZ](#) {cmat::Zero(8, 8)}
- Projector onto the GHZ state.*
- [cmat pW](#) {cmat::Zero(8, 8)}
- Projector onto the W state.*

### Private Member Functions

- [States](#) ()
  - [~States](#) ()=default
- Default destructor.*

### Friends

- class [internal::Singleton](#)< [const States](#) >

### Additional Inherited Members

#### 7.47.1 Detailed Description

const Singleton class that implements most commonly used states

#### 7.47.2 Constructor & Destructor Documentation

##### 7.47.2.1 States()

```
qpp::States::States ( ) [inline], [private]
```

Initialize the states

##### 7.47.2.2 ~States()

```
qpp::States::~~States ( ) [private], [default]
```

Default destructor.

### 7.47.3 Member Function Documentation

#### 7.47.3.1 `jn()`

```
ket qpp::States::jn (
    idx j,
    idx n,
    idx d = 2 ) const [inline]
```

$|j\rangle^{\otimes n}$  state of  $n$  qudits

##### Parameters

$j$	Non-negative integer
$n$	Non-negative integer
$d$	Subsystem dimensions

##### Returns

$|j\rangle^{\otimes n}$  state of  $n$  qudits

#### 7.47.3.2 `mes()`

```
ket qpp::States::mes (
    idx d = 2 ) const [inline]
```

Maximally entangled state of 2 qudits.

##### Parameters

$d$	Subsystem dimensions
-----	----------------------

##### Returns

Maximally entangled state  $\frac{1}{\sqrt{d}} \sum_{j=0}^{d-1} |jj\rangle$  of 2 qudits

#### 7.47.3.3 `minus()`

```
ket qpp::States::minus (
    idx n ) const [inline]
```

Minus state of  $n$  qubits.

## Parameters

$n$	Non-negative integer
-----	----------------------

## Returns

Minus state  $|-\rangle^{\otimes n}$  of  $n$  qubits

## 7.47.3.4 one()

```
ket qpp::States::one (
    idx n,
    idx d = 2 ) const [inline]
```

One state of  $n$  qudits.

## Parameters

$n$	Non-negative integer
$d$	Subsystem dimensions

## Returns

One state  $|1\rangle^{\otimes n}$  of  $n$  qudits

## 7.47.3.5 plus()

```
ket qpp::States::plus (
    idx n ) const [inline]
```

Plus state of  $n$  qubits.

## Parameters

$n$	Non-negative integer
-----	----------------------

## Returns

Plus state  $|+\rangle^{\otimes n}$  of  $n$  qubits

### 7.47.3.6 zero()

```
ket qpp::States::zero (
    idx n,
    idx d = 2 ) const [inline]
```

Zero state of  $n$  qudits.

#### Parameters

$n$	Non-negative integer
$d$	Subsystem dimensions

#### Returns

Zero state  $|0\rangle^{\otimes n}$  of  $n$  qudits

## 7.47.4 Friends And Related Function Documentation

### 7.47.4.1 internal::Singleton< const States >

```
friend class internal::Singleton< const States > [friend]
```

## 7.47.5 Member Data Documentation

### 7.47.5.1 b00

```
ket qpp::States::b00 {ket::Zero(4)}
```

Bell-00 state (following the convention in Nielsen and Chuang)

### 7.47.5.2 b01

```
ket qpp::States::b01 {ket::Zero(4)}
```

Bell-01 state (following the convention in Nielsen and Chuang)

### 7.47.5.3 b10

```
ket qpp::States::b10 {ket::Zero(4)}
```

Bell-10 state (following the convention in Nielsen and Chuang)

### 7.47.5.4 b11

```
ket qpp::States::b11 {ket::Zero(4)}
```

Bell-11 state (following the convention in Nielsen and Chuang)

### 7.47.5.5 GHZ

```
ket qpp::States::GHZ {ket::Zero(8)}
```

GHZ state.

### 7.47.5.6 pb00

```
cmat qpp::States::pb00 {cmat::Zero(4, 4)}
```

Projector onto the Bell-00 state.

### 7.47.5.7 pb01

```
cmat qpp::States::pb01 {cmat::Zero(4, 4)}
```

Projector onto the Bell-01 state.

### 7.47.5.8 pb10

```
cmat qpp::States::pb10 {cmat::Zero(4, 4)}
```

Projector onto the Bell-10 state.

#### 7.47.5.9 pb11

```
cmat qpp::States::pb11 {cmat::Zero(4, 4)}
```

Projector onto the Bell-11 state.

#### 7.47.5.10 pGHZ

```
cmat qpp::States::pGHZ {cmat::Zero(8, 8)}
```

Projector onto the GHZ state.

#### 7.47.5.11 pW

```
cmat qpp::States::pW {cmat::Zero(8, 8)}
```

Projector onto the W state.

#### 7.47.5.12 px0

```
cmat qpp::States::px0 {cmat::Zero(2, 2)}
```

Projector onto the Pauli Sigma-X 0-eigenstate  $|+\rangle\langle+|$ .

#### 7.47.5.13 px1

```
cmat qpp::States::px1 {cmat::Zero(2, 2)}
```

Projector onto the Pauli Sigma-X 1-eigenstate  $|-\rangle\langle-|$ .

#### 7.47.5.14 py0

```
cmat qpp::States::py0 {cmat::Zero(2, 2)}
```

Projector onto the Pauli Sigma-Y 0-eigenstate  $|y+\rangle\langle y+|$ .



#### 7.47.5.15 py1

```
cmat qpp::States::py1 {cmat::Zero(2, 2)}
```

Projector onto the Pauli Sigma-Y 1-eigenstate  $|y\rangle\langle y|$ .

#### 7.47.5.16 pz0

```
cmat qpp::States::pz0 {cmat::Zero(2, 2)}
```

Projector onto the Pauli Sigma-Z 0-eigenstate  $|0\rangle\langle 0|$ .

#### 7.47.5.17 pz1

```
cmat qpp::States::pz1 {cmat::Zero(2, 2)}
```

Projector onto the Pauli Sigma-Z 1-eigenstate  $|1\rangle\langle 1|$ .

#### 7.47.5.18 W

```
ket qpp::States::W {ket::Zero(8)}
```

W state.

#### 7.47.5.19 x0

```
ket qpp::States::x0 {ket::Zero(2)}
```

Pauli Sigma-X 0-eigenstate  $|+\rangle$

#### 7.47.5.20 x1

```
ket qpp::States::x1 {ket::Zero(2)}
```

Pauli Sigma-X 1-eigenstate  $|-\rangle$

#### 7.47.5.21 y0

```
ket qpp::States::y0 {ket::Zero(2)}
```

Pauli Sigma-Y 0-eigenstate  $|y+\rangle$

#### 7.47.5.22 y1

```
ket qpp::States::y1 {ket::Zero(2)}
```

Pauli Sigma-Y 1-eigenstate  $|y-\rangle$

#### 7.47.5.23 z0

```
ket qpp::States::z0 {ket::Zero(2)}
```

Pauli Sigma-Z 0-eigenstate  $|0\rangle$

#### 7.47.5.24 z1

```
ket qpp::States::z1 {ket::Zero(2)}
```

Pauli Sigma-Z 1-eigenstate  $|1\rangle$

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

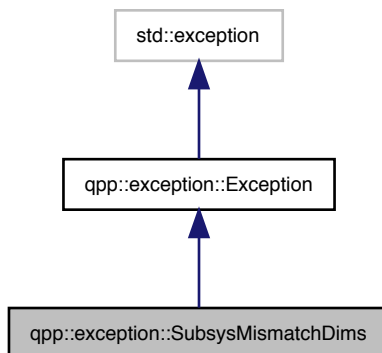
- [classes/states.h](#)

## 7.48 qpp::exception::SubsysMismatchDims Class Reference

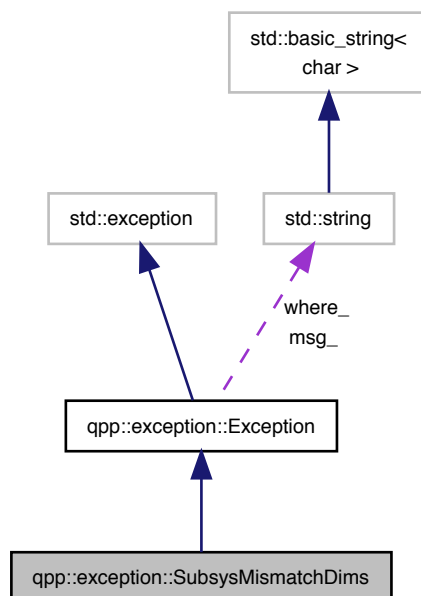
Subsystems mismatch dimensions exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::SubsysMismatchDims:



Collaboration diagram for qpp::exception::SubsysMismatchDims:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.48.1 Detailed Description

Subsystems mismatch dimensions exception.

`std::vector<idx>` of subsystem labels has duplicates, or has entries that are larger than the size of the `std::vector<idx>` of dimensions

### 7.48.2 Member Function Documentation

#### 7.48.2.1 type\_description()

```
std::string qpp::exception::SubsysMismatchDims::type_description ( ) const [inline], [override], [virtual]
```

*Exception type description.*

#### Returns

*Exception type description*

Implements `qpp::exception::Exception`.

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

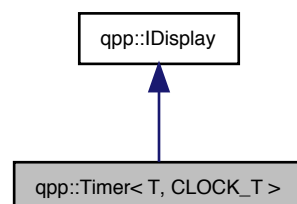
- `classes/exception.h`

## 7.49 qpp::Timer< T, CLOCK\_T > Class Template Reference

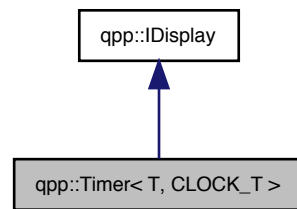
Chronometer.

```
#include <classes/timer.h>
```

Inheritance diagram for `qpp::Timer< T, CLOCK_T >`:



Collaboration diagram for qpp::Timer< T, CLOCK\_T >:



## Public Member Functions

- `Timer ()` noexcept  
*Constructs an instance with the current time as the starting point.*
- `void tic ()` noexcept  
*Resets the chronometer.*
- `const Timer & toc ()` noexcept  
*Stops the chronometer.*
- `double tics ()` const noexcept  
*Time passed in the duration specified by T.*
- `template<typename U = T>`  
  `U get_duration ()` const noexcept  
    *Duration specified by U.*
- `Timer (const Timer &)=default`  
    *Default copy constructor.*
- `Timer (Timer &&)=default`  
    *Default move constructor.*
- `Timer & operator= (const Timer &)=default`  
    *Default copy assignment operator.*
- `Timer & operator= (Timer &&)=default`  
    *Default move assignment operator.*
- `virtual ~Timer ()=default`  
    *Default virtual destructor.*

## Protected Attributes

- `CLOCK_T::time_point start_`
- `CLOCK_T::time_point end_`

## Private Member Functions

- `std::ostream & display (std::ostream &os)` const override  
    *qpp::IDisplay::display() override*

### 7.49.1 Detailed Description

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady_clock>
class qpp::Timer< T, CLOCK_T >
```

Chronometer.

#### Template Parameters

<i>T</i>	Tics duration, default is <code>std::chrono::duration&lt;double, 1&gt;</code> , i.e. seconds in double precision
<i>CLOCK_T</i>	Clock's type, default is <code>std::chrono::steady_clock</code> , not affected by wall clock changes during runtime

### 7.49.2 Constructor & Destructor Documentation

#### 7.49.2.1 Timer() [1/3]

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady_clock>
qpp::Timer< T, CLOCK_T >::Timer ( ) [inline], [noexcept]
```

Constructs an instance with the current time as the starting point.

#### 7.49.2.2 Timer() [2/3]

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady_clock>
qpp::Timer< T, CLOCK_T >::Timer (
    const Timer< T, CLOCK_T > & ) [default]
```

Default copy constructor.

#### 7.49.2.3 Timer() [3/3]

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady_clock>
qpp::Timer< T, CLOCK_T >::Timer (
    Timer< T, CLOCK_T > && ) [default]
```

Default move constructor.

## 7.49.2.4 ~Timer()

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵_clock>
virtual qpp::Timer< T, CLOCK_T >::~~Timer ( ) [virtual], [default]
```

Default virtual destructor.

## 7.49.3 Member Function Documentation

## 7.49.3.1 display()

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵_clock>
std::ostream& qpp::Timer< T, CLOCK_T >::display (
    std::ostream & os ) const [inline], [override], [private], [virtual]
```

[qpp::IDisplay::display\(\)](#) override

## Parameters

<code>os</code>	Output stream
-----------------	---------------

## Returns

Writes to the output stream the number of tics (specified by T) that passed between the instantiation/reset and invocation of [qpp::Timer::toc\(\)](#).

Implements [qpp::IDisplay](#).

## 7.49.3.2 get\_duration()

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵_clock>
template<typename U = T>
U qpp::Timer< T, CLOCK_T >::get_duration ( ) const [inline], [noexcept]
```

Duration specified by U.

## Template Parameters

<code>U</code>	Duration, default is T, which defaults to <code>std::chrono::duration&lt;double, 1&gt;</code> , i.e. seconds in double precision
----------------	--

**Returns**

Duration that passed between the instantiation/reset and invocation of [qpp::Timer::toc\(\)](#)

**7.49.3.3 operator=()** [1/2]

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵
_clock>
Timer& qpp::Timer< T, CLOCK_T >::operator= (
    const Timer< T, CLOCK_T > & ) [default]
```

Default copy assignment operator.

**7.49.3.4 operator=()** [2/2]

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵
_clock>
Timer& qpp::Timer< T, CLOCK_T >::operator= (
    Timer< T, CLOCK_T > && ) [default]
```

Default move assignment operator.

**7.49.3.5 tic()**

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵
_clock>
void qpp::Timer< T, CLOCK_T >::tic ( ) [inline], [noexcept]
```

Resets the chronometer.

Resets the starting/ending point to the current time

**7.49.3.6 tics()**

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵
_clock>
double qpp::Timer< T, CLOCK_T >::tics ( ) const [inline], [noexcept]
```

Time passed in the duration specified by T.

**Returns**

Number of tics (specified by T) that passed between the instantiation/reset and invocation of [qpp::Timer::toc\(\)](#)



#### 7.49.3.7 toc()

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵_clock>
const Timer& qpp::Timer< T, CLOCK_T >::toc ( ) [inline], [noexcept]
```

Stops the chronometer.

Set the current time as the ending point

#### Returns

Reference to the current instance

### 7.49.4 Member Data Documentation

#### 7.49.4.1 end\_

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵_clock>
CLOCK_T::time_point qpp::Timer< T, CLOCK_T >::end_ [protected]
```

#### 7.49.4.2 start\_

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↵_clock>
CLOCK_T::time_point qpp::Timer< T, CLOCK_T >::start_ [protected]
```

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

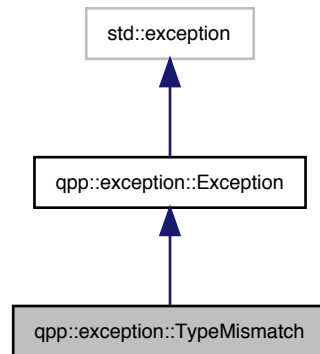
- [classes/timer.h](#)

## 7.50 qpp::exception::TypeMismatch Class Reference

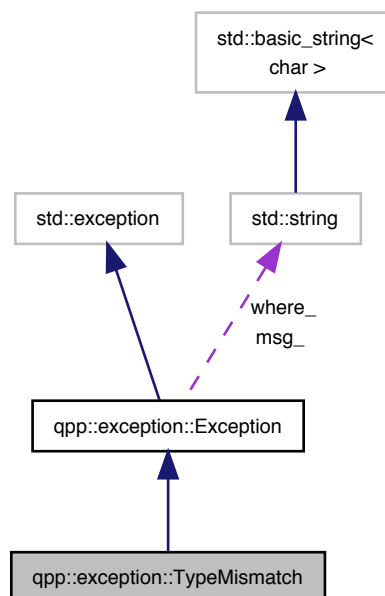
Type mismatch exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::TypeMismatch:



Collaboration diagram for qpp::exception::TypeMismatch:



## Public Member Functions

- `std::string type_description () const` override  
*Exception type description.*

### 7.50.1 Detailed Description

Type mismatch exception.

Scalar types do not match

### 7.50.2 Member Function Documentation

#### 7.50.2.1 type\_description()

```
std::string qpp::exception::TypeMismatch::type_description ( ) const [inline], [override],  
[virtual]
```

*Exception* type description.

#### Returns

*Exception* type description

Implements `qpp::exception::Exception`.

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

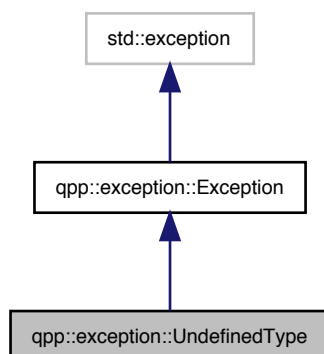
- `classes/exception.h`

## 7.51 qpp::exception::UndefinedType Class Reference

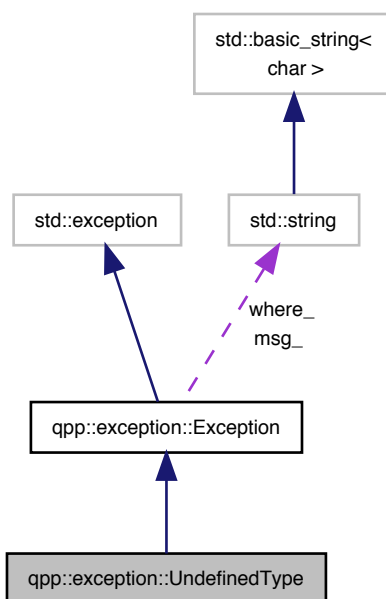
Not defined for this type exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::UndefinedType:



Collaboration diagram for qpp::exception::UndefinedType:



## Public Member Functions

- `std::string type\_description ()` const override  
*[Exception](#) type description.*

### 7.51.1 Detailed Description

Not defined for this type exception.

Templated specialization is not defined for this type

### 7.51.2 Member Function Documentation

#### 7.51.2.1 type\_description()

```
std::string qpp::exception::UndefinedType::type_description ( ) const [inline], [override],  
[virtual]
```

Exception type description.

#### Returns

Exception type description

Implements [qpp::exception::Exception](#).

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

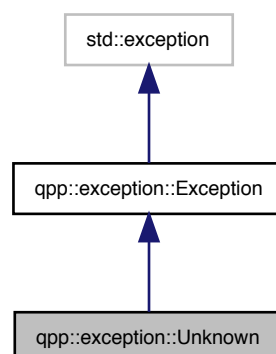
- [classes/exception.h](#)

## 7.52 qpp::exception::Unknown Class Reference

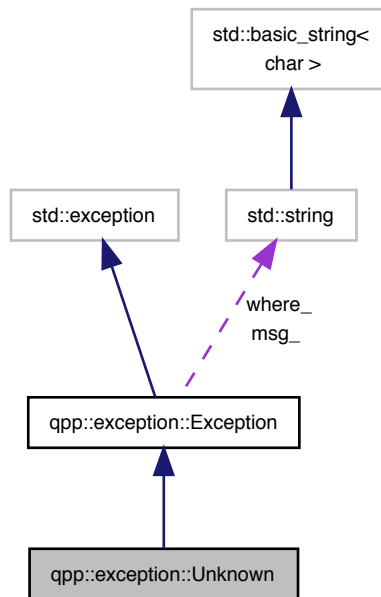
Unknown exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::Unknown:



Collaboration diagram for `qpp::exception::Unknown`:



## Public Member Functions

- `std::string type_description () const` override  
*Exception type description.*

### 7.52.1 Detailed Description

`Unknown` exception.

Thrown when no other exception is suitable (not recommended, it is better to define another suitable exception type)

### 7.52.2 Member Function Documentation

#### 7.52.2.1 type\_description()

```
std::string qpp::exception::Unknown::type_description ( ) const [inline], [override], [virtual]
```

`Exception` type description.

#### Returns

`Exception` type description

Implements `qpp::exception::Exception`.

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

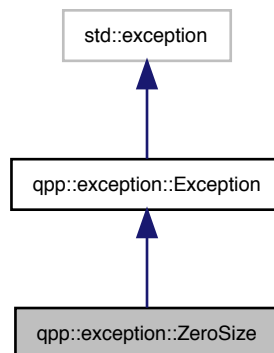
- `classes/exception.h`

## 7.53 qpp::exception::ZeroSize Class Reference

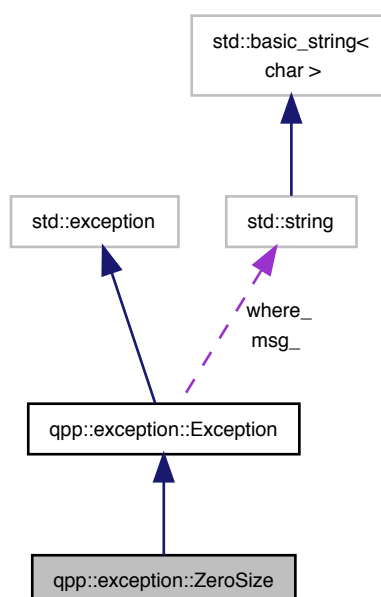
Object has zero size exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::ZeroSize:



Collaboration diagram for qpp::exception::ZeroSize:



## Public Member Functions

- `std::string type_description ()` const override  
*Exception type description.*

### 7.53.1 Detailed Description

Object has zero size exception.

Zero sized object, e.g. empty `Eigen::Matrix` or `std::vector` with no elements

### 7.53.2 Member Function Documentation

#### 7.53.2.1 `type_description()`

```
std::string qpp::exception::ZeroSize::type_description ( ) const [inline], [override], [virtual]
```

*Exception* type description.

#### Returns

*Exception* type description

Implements `qpp::exception::Exception`.

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

- `classes/exception.h`



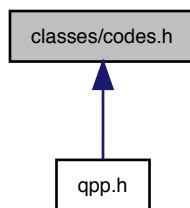
## Chapter 8

# File Documentation

### 8.1 classes/codes.h File Reference

Quantum error correcting codes.

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



#### Classes

- class [qpp::Codes](#)  
*const Singleton class that defines quantum error correcting codes*

#### Namespaces

- [qpp](#)  
*Quantum++ main namespace.*

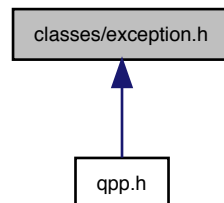
#### 8.1.1 Detailed Description

Quantum error correcting codes.

## 8.2 classes/exception.h File Reference

Exceptions.

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



### Classes

- class [qpp::exception::Exception](#)  
*Base class for generating Quantum++ custom exceptions.*
- class [qpp::exception::Unknown](#)  
*Unknown exception.*
- class [qpp::exception::ZeroSize](#)  
*Object has zero size exception.*
- class [qpp::exception::MatrixNotSquare](#)  
*Matrix is not square exception.*
- class [qpp::exception::MatrixNotCvector](#)  
*Matrix is not a column vector exception.*
- class [qpp::exception::MatrixNotRvector](#)  
*Matrix is not a row vector exception.*
- class [qpp::exception::MatrixNotVector](#)  
*Matrix is not a vector exception.*
- class [qpp::exception::MatrixNotSquareNorCvector](#)  
*Matrix is not square nor column vector exception.*
- class [qpp::exception::MatrixNotSquareNorRvector](#)  
*Matrix is not square nor row vector exception.*
- class [qpp::exception::MatrixNotSquareNorVector](#)  
*Matrix is not square nor vector exception.*
- class [qpp::exception::MatrixMismatchSubsys](#)  
*Matrix mismatch subsystems exception.*
- class [qpp::exception::DimsInvalid](#)  
*Invalid dimension(s) exception.*
- class [qpp::exception::DimsNotEqual](#)  
*Dimensions not equal exception.*
- class [qpp::exception::DimsMismatchMatrix](#)  
*Dimension(s) mismatch matrix size exception.*
- class [qpp::exception::DimsMismatchCvector](#)

- Dimension(s) mismatch column vector size exception.*

  - class [qpp::exception::DimsMismatchRvector](#)
- Dimension(s) mismatch row vector size exception.*

  - class [qpp::exception::DimsMismatchVector](#)
- Dimension(s) mismatch vector size exception.*

  - class [qpp::exception::SubsysMismatchDims](#)
- Subsystems mismatch dimensions exception.*

  - class [qpp::exception::PermlInvalid](#)
- Invalid permutation exception.*

  - class [qpp::exception::PermMismatchDims](#)
- Permutation mismatch dimensions exception.*

  - class [qpp::exception::NotQubitMatrix](#)
- Matrix is not 2 x 2 exception.*

  - class [qpp::exception::NotQubitCvector](#)
- Column vector is not 2 x 1 exception.*

  - class [qpp::exception::NotQubitRvector](#)
- Row vector is not 1 x 2 exception.*

  - class [qpp::exception::NotQubitVector](#)
- Vector is not 2 x 1 nor 1 x 2 exception.*

  - class [qpp::exception::NotQubitSubsys](#)
- Subsystems are not qubits exception.*

  - class [qpp::exception::NotBipartite](#)
- Not bi-partite exception.*

  - class [qpp::exception::NoCodeword](#)
- Codeword does not exist exception.*

  - class [qpp::exception::OutOfRange](#)
- Parameter out of range exception.*

  - class [qpp::exception::TypeMismatch](#)
- Type mismatch exception.*

  - class [qpp::exception::SizeMismatch](#)
- Size mismatch exception.*

  - class [qpp::exception::UndefinedType](#)
- Not defined for this type exception.*

  - class [qpp::exception::CustomException](#)
- Custom exception.*

## Namespaces

- [qpp](#)

*Quantum++ main namespace.*
- [qpp::exception](#)

*Quantum++ exception hierarchy namespace.*

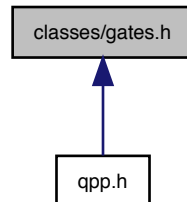
### 8.2.1 Detailed Description

Exceptions.

## 8.3 classes/gates.h File Reference

Quantum gates.

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



### Classes

- class [qpp::Gates](#)  
*const Singleton class that implements most commonly used gates*

### Namespaces

- [qpp](#)  
*Quantum++ main namespace.*

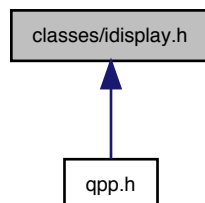
#### 8.3.1 Detailed Description

Quantum gates.

## 8.4 classes/ideisplay.h File Reference

Display interface via the non-virtual interface (NVI)

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



## Classes

- class [qpp::IDisplay](#)

*Abstract class (interface) that mandates the definition of virtual `std::ostream& display(std::ostream& os) const`.*

## Namespaces

- [qpp](#)

*Quantum++ main namespace.*

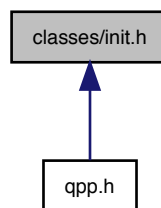
### 8.4.1 Detailed Description

Display interface via the non-virtual interface (NVI)

## 8.5 classes/init.h File Reference

Initialization.

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



## Classes

- class [qpp::Init](#)

*const Singleton class that performs additional initializations/cleanups*

## Namespaces

- [qpp](#)

*Quantum++ main namespace.*

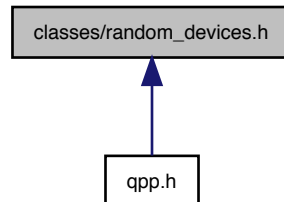
### 8.5.1 Detailed Description

Initialization.

## 8.6 classes/random\_devices.h File Reference

Random devices.

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



### Classes

- class [qpp::RandomDevices](#)  
*Singleton class that manages the source of randomness in the library.*

### Namespaces

- [qpp](#)  
*Quantum++ main namespace.*

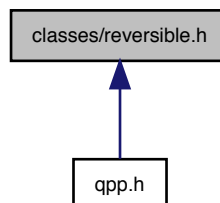
### 8.6.1 Detailed Description

Random devices.

## 8.7 classes/reversible.h File Reference

Support for classical reversible circuits.

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



## Classes

- class [qpp::Dynamic\\_bitset](#)  
*Dynamic bitset class, allows the specification of the number of bits at runtime (unlike `std::bitset<N>`)*
- class [qpp::Bit\\_circuit](#)  
*Classical reversible circuit simulator.*
- struct [qpp::Bit\\_circuit::Gate\\_count](#)

## Namespaces

- [qpp](#)  
*Quantum++ main namespace.*

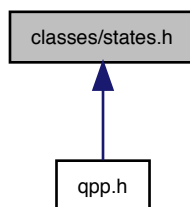
### 8.7.1 Detailed Description

Support for classical reversible circuits.

## 8.8 classes/states.h File Reference

Quantum states.

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



## Classes

- class [qpp::States](#)  
*const Singleton class that implements most commonly used states*

## Namespaces

- [qpp](#)  
*Quantum++ main namespace.*

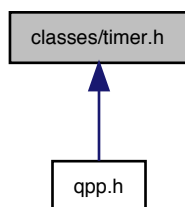
### 8.8.1 Detailed Description

Quantum states.

## 8.9 classes/timer.h File Reference

Timing.

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



### Classes

- class [qpp::Timer< T, CLOCK\\_T >](#)  
*Chronometer.*

### Namespaces

- [qpp](#)  
*Quantum++ main namespace.*

### 8.9.1 Detailed Description

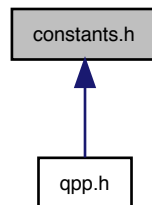
Timing.



## 8.10 constants.h File Reference

Constants.

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



### Namespaces

- [qpp](#)  
*Quantum++ main namespace.*
- [qpp::literals](#)

### Functions

- constexpr cplx [qpp::literals::operator"" \\_i](#) (unsigned long long int x) noexcept  
*User-defined literal for complex  $i = \sqrt{-1}$  (integer overload)*
- constexpr cplx [qpp::operator"" \\_i](#) (long double x) noexcept  
*User-defined literal for complex  $i = \sqrt{-1}$  (real overload)*
- cplx [qpp::omega](#) (idx D)  
*D-th root of unity.*

### Variables

- constexpr double [qpp::chop](#) = 1e-10  
*Used in [qpp::disp\(\)](#) for setting to zero numbers that have their absolute value smaller than [qpp::chop](#).*
- constexpr double [qpp::eps](#) = 1e-12  
*Used to decide whether a number or expression in double precision is zero or not.*
- constexpr idx [qpp::maxn](#) = 64  
*Maximum number of allowed qubits/qudits (subsystems)*
- constexpr double [qpp::pi](#) = 3.141592653589793238462643383279502884  
 $\pi$
- constexpr double [qpp::ee](#) = 2.718281828459045235360287471352662497  
*Base of natural logarithm, e.*
- constexpr double [qpp::infy](#) = std::numeric\_limits<double>::max()  
*Used to denote infinity in double precision.*

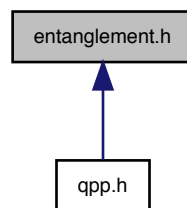
### 8.10.1 Detailed Description

Constants.

## 8.11 entanglement.h File Reference

Entanglement functions.

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



### Namespaces

- [qpp](#)  
*Quantum++ main namespace.*

### Functions

- `template<typename Derived >`  
`dyn_col_vect< double > qpp::schmidtcoeffs (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)`  
*Schmidt coefficients of the bi-partite pure state A.*
- `template<typename Derived >`  
`dyn_col_vect< double > qpp::schmidtcoeffs (const Eigen::MatrixBase< Derived > &A, idx d=2)`  
*Schmidt coefficients of the bi-partite pure state A.*
- `template<typename Derived >`  
`cmat qpp::schmidtA (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)`  
*Schmidt basis on Alice side.*
- `template<typename Derived >`  
`cmat qpp::schmidtA (const Eigen::MatrixBase< Derived > &A, idx d=2)`  
*Schmidt basis on Alice side.*
- `template<typename Derived >`  
`cmat qpp::schmidtB (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)`  
*Schmidt basis on Bob side.*
- `template<typename Derived >`  
`cmat qpp::schmidtB (const Eigen::MatrixBase< Derived > &A, idx d=2)`

*Schmidt basis on Bob side.*

- `template<typename Derived >`  
`std::vector< double > qpp::schmidtprobs (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)`

*Schmidt probabilities of the bi-partite pure state A.*

- `template<typename Derived >`  
`std::vector< double > qpp::schmidtprobs (const Eigen::MatrixBase< Derived > &A, idx d=2)`

*Schmidt probabilities of the bi-partite pure state A.*

- `template<typename Derived >`  
`double qpp::entanglement (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)`

*Entanglement of the bi-partite pure state A.*

- `template<typename Derived >`  
`double qpp::entanglement (const Eigen::MatrixBase< Derived > &A, idx d=2)`

*Entanglement of the bi-partite pure state A.*

- `template<typename Derived >`  
`double qpp::gconcurrence (const Eigen::MatrixBase< Derived > &A)`

*G-concurrence of the bi-partite pure state A.*

- `template<typename Derived >`  
`double qpp::negativity (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)`

*Negativity of the bi-partite mixed state A.*

- `template<typename Derived >`  
`double qpp::negativity (const Eigen::MatrixBase< Derived > &A, idx d=2)`

*Negativity of the bi-partite mixed state A.*

- `template<typename Derived >`  
`double qpp::lognegativity (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)`

*Logarithmic negativity of the bi-partite mixed state A.*

- `template<typename Derived >`  
`double qpp::lognegativity (const Eigen::MatrixBase< Derived > &A, idx d=2)`

*Logarithmic negativity of the bi-partite mixed state A.*

- `template<typename Derived >`  
`double qpp::concurrence (const Eigen::MatrixBase< Derived > &A)`

*Wootters concurrence of the bi-partite qubit mixed state A.*

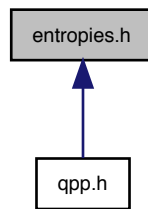
### 8.11.1 Detailed Description

Entanglement functions.

## 8.12 entropies.h File Reference

Entropy functions.

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



## Namespaces

- [qpp](#)  
*Quantum++ main namespace.*

## Functions

- `template<typename Derived >`  
`double qpp::entropy (const Eigen::MatrixBase< Derived > &A)`  
*von-Neumann entropy of the density matrix A*
- `double qpp::entropy (const std::vector< double > &prob)`  
*Shannon entropy of the probability distribution prob.*
- `template<typename Derived >`  
`double qpp::renyi (const Eigen::MatrixBase< Derived > &A, double alpha)`  
*Renyi-  $\alpha$  entropy of the density matrix A, for  $\alpha \geq 0$ .*
- `double qpp::renyi (const std::vector< double > &prob, double alpha)`  
*Renyi-  $\alpha$  entropy of the probability distribution prob, for  $\alpha \geq 0$ .*
- `template<typename Derived >`  
`double qpp::tsallis (const Eigen::MatrixBase< Derived > &A, double q)`  
*Tsallis-  $q$  entropy of the density matrix A, for  $q \geq 0$ .*
- `double qpp::tsallis (const std::vector< double > &prob, double q)`  
*Tsallis-  $q$  entropy of the probability distribution prob, for  $q \geq 0$ .*
- `template<typename Derived >`  
`double qpp::qmutualinfo (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsysA, const std::vector< idx > &subsysB, const std::vector< idx > &dims)`  
*Quantum mutual information between 2 subsystems of a composite system.*
- `template<typename Derived >`  
`double qpp::qmutualinfo (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsysA, const std::vector< idx > &subsysB, idx d=2)`  
*Quantum mutual information between 2 subsystems of a composite system.*

## 8.12.1 Detailed Description

Entropy functions.

## 8.13 experimental/experimental.h File Reference

Experimental/test functions/classes.

### Namespaces

- [qpp](#)  
*Quantum++ main namespace.*
- [qpp::experimental](#)  
*Experimental/test functions/classes, do not use or modify.*

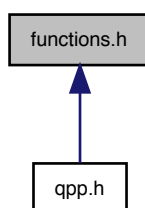
### 8.13.1 Detailed Description

Experimental/test functions/classes.

## 8.14 functions.h File Reference

Generic quantum computing functions.

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



### Namespaces

- [qpp](#)  
*Quantum++ main namespace.*
- [qpp::literals](#)

## Functions

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::transpose (const Eigen::MatrixBase< Derived > &A)`  
*Transpose.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::conjugate (const Eigen::MatrixBase< Derived > &A)`  
*Complex conjugate.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::adjoint (const Eigen::MatrixBase< Derived > &A)`  
*Adjoint.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::inverse (const Eigen::MatrixBase< Derived > &A)`  
*Inverse.*
- `template<typename Derived >`  
`Derived::Scalar qpp::trace (const Eigen::MatrixBase< Derived > &A)`  
*Trace.*
- `template<typename Derived >`  
`Derived::Scalar qpp::det (const Eigen::MatrixBase< Derived > &A)`  
*Determinant.*
- `template<typename Derived >`  
`Derived::Scalar qpp::logdet (const Eigen::MatrixBase< Derived > &A)`  
*Logarithm of the determinant.*
- `template<typename Derived >`  
`Derived::Scalar qpp::sum (const Eigen::MatrixBase< Derived > &A)`  
*Element-wise sum of A.*
- `template<typename Derived >`  
`Derived::Scalar qpp::prod (const Eigen::MatrixBase< Derived > &A)`  
*Element-wise product of A.*
- `template<typename Derived >`  
`double qpp::norm (const Eigen::MatrixBase< Derived > &A)`  
*Frobenius norm.*
- `template<typename Derived >`  
`std::pair< dyn_col_vect< cplx >, cmat > qpp::eig (const Eigen::MatrixBase< Derived > &A)`  
*Full eigen decomposition.*
- `template<typename Derived >`  
`dyn_col_vect< cplx > qpp::evals (const Eigen::MatrixBase< Derived > &A)`  
*Eigenvalues.*
- `template<typename Derived >`  
`cmat qpp::evecs (const Eigen::MatrixBase< Derived > &A)`  
*Eigenvectors.*
- `template<typename Derived >`  
`std::pair< dyn_col_vect< double >, cmat > qpp::heig (const Eigen::MatrixBase< Derived > &A)`  
*Full eigen decomposition of Hermitian expression.*
- `template<typename Derived >`  
`dyn_col_vect< double > qpp::hevals (const Eigen::MatrixBase< Derived > &A)`  
*Hermitian eigenvalues.*
- `template<typename Derived >`  
`cmat qpp::hevecs (const Eigen::MatrixBase< Derived > &A)`  
*Hermitian eigenvectors.*
- `template<typename Derived >`  
`std::tuple< cmat, dyn_col_vect< double >, cmat > qpp::svd (const Eigen::MatrixBase< Derived > &A)`  
*Full singular value decomposition.*

- `template<typename Derived >`  
`dyn_col_vect< double > qpp::svals (const Eigen::MatrixBase< Derived > &A)`  
*Singular values.*
- `template<typename Derived >`  
`cmat qpp::svdU (const Eigen::MatrixBase< Derived > &A)`  
*Left singular vectors.*
- `template<typename Derived >`  
`cmat qpp::svdV (const Eigen::MatrixBase< Derived > &A)`  
*Right singular vectors.*
- `template<typename Derived >`  
`cmat qpp::funm (const Eigen::MatrixBase< Derived > &A, cplx(*f)(const cplx &))`  
*Functional calculus  $f(A)$*
- `template<typename Derived >`  
`cmat qpp::sqrtm (const Eigen::MatrixBase< Derived > &A)`  
*Matrix square root.*
- `template<typename Derived >`  
`cmat qpp::absm (const Eigen::MatrixBase< Derived > &A)`  
*Matrix absolute value.*
- `template<typename Derived >`  
`cmat qpp::expm (const Eigen::MatrixBase< Derived > &A)`  
*Matrix exponential.*
- `template<typename Derived >`  
`cmat qpp::logm (const Eigen::MatrixBase< Derived > &A)`  
*Matrix logarithm.*
- `template<typename Derived >`  
`cmat qpp::sinm (const Eigen::MatrixBase< Derived > &A)`  
*Matrix sin.*
- `template<typename Derived >`  
`cmat qpp::cosm (const Eigen::MatrixBase< Derived > &A)`  
*Matrix cos.*
- `template<typename Derived >`  
`cmat qpp::spectralpowm (const Eigen::MatrixBase< Derived > &A, const cplx z)`  
*Matrix power.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::powm (const Eigen::MatrixBase< Derived > &A, idx n)`  
*Fast matrix power based on the SQUARE-AND-MULTIPLY algorithm.*
- `template<typename Derived >`  
`double qpp::schatten (const Eigen::MatrixBase< Derived > &A, double p)`  
*Schatten matrix norm.*
- `template<typename OutputScalar, typename Derived >`  
`dyn_mat< OutputScalar > qpp::cwise (const Eigen::MatrixBase< Derived > &A, OutputScalar(*f)(const typename Derived::Scalar &))`  
*Functor.*
- `template<typename T >`  
`dyn_mat< typename T::Scalar > qpp::kron (const T &head)`  
*Kronecker product.*
- `template<typename T, typename... Args>`  
`dyn_mat< typename T::Scalar > qpp::kron (const T &head, const Args &... tail)`  
*Kronecker product.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::kron (const std::vector< Derived > &As)`  
*Kronecker product.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::kron (const std::initializer_list< Derived > &As)`  
*Kronecker product.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::kronpow (const Eigen::MatrixBase< Derived > &A, idx n)`  
*Kronecker power.*
- `template<typename T >`  
`dyn_mat< typename T::Scalar > qpp::dirsum (const T &head)`  
*Direct sum.*
- `template<typename T, typename... Args>`  
`dyn_mat< typename T::Scalar > qpp::dirsum (const T &head, const Args &... tail)`  
*Direct sum.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::dirsum (const std::vector< Derived > &As)`  
*Direct sum.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::dirsum (const std::initializer_list< Derived > &As)`  
*Direct sum.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::dirsumpow (const Eigen::MatrixBase< Derived > &A, idx n)`  
*Direct sum power.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::reshape (const Eigen::MatrixBase< Derived > &A, idx rows, idx cols)`  
*Reshape.*
- `template<typename Derived1, typename Derived2 >`  
`dyn_mat< typename Derived1::Scalar > qpp::comm (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)`  
*Commutator.*
- `template<typename Derived1, typename Derived2 >`  
`dyn_mat< typename Derived1::Scalar > qpp::anticomm (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)`  
*Anti-commutator.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::prj (const Eigen::MatrixBase< Derived > &A)`  
*Projector.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::grams (const std::vector< Derived > &As)`  
*Gram-Schmidt orthogonalization.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::grams (const std::initializer_list< Derived > &As)`  
*Gram-Schmidt orthogonalization.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::grams (const Eigen::MatrixBase< Derived > &A)`  
*Gram-Schmidt orthogonalization.*
- `std::vector< idx > qpp::n2multiidx (idx n, const std::vector< idx > &dims)`  
*Non-negative integer index to multi-index.*
- `idx qpp::multiidx2n (const std::vector< idx > &midx, const std::vector< idx > &dims)`  
*Multi-index to non-negative integer index.*
- `ket qpp::mket (const std::vector< idx > &mask, const std::vector< idx > &dims)`  
*Multi-partite qudit ket.*
- `ket qpp::mket (const std::vector< idx > &mask, idx d=2)`  
*Multi-partite qudit ket.*



- `cmat qpp::mprj` (const std::vector< idx > &mask, const std::vector< idx > &dims)  
*Projector onto multi-partite qudit ket.*
- `cmat qpp::mprj` (const std::vector< idx > &mask, idx d=2)  
*Projector onto multi-partite qudit ket.*
- `template<typename InputIterator >`  
`std::vector< double > qpp::abssq` (InputIterator first, InputIterator last)  
*Computes the absolute values squared of an STL-like range of complex numbers.*
- `template<typename Container >`  
`std::vector< double > qpp::abssq` (const Container &c, typename std::enable\_if< is\_iterable< Container >::value >::type \* = nullptr)  
*Computes the absolute values squared of an STL-like container.*
- `template<typename Derived >`  
`std::vector< double > qpp::abssq` (const Eigen::MatrixBase< Derived > &A)  
*Computes the absolute values squared of an Eigen expression.*
- `template<typename InputIterator >`  
`std::iterator_traits< InputIterator >::value_type qpp::sum` (InputIterator first, InputIterator last)  
*Element-wise sum of an STL-like range.*
- `template<typename Container >`  
`Container::value_type qpp::sum` (const Container &c, typename std::enable\_if< is\_iterable< Container >::value >::type \* = nullptr)  
*Element-wise sum of the elements of an STL-like container.*
- `template<typename InputIterator >`  
`std::iterator_traits< InputIterator >::value_type qpp::prod` (InputIterator first, InputIterator last)  
*Element-wise product of an STL-like range.*
- `template<typename Container >`  
`Container::value_type qpp::prod` (const Container &c, typename std::enable\_if< is\_iterable< Container >::value >::type \* = nullptr)  
*Element-wise product of the elements of an STL-like container.*
- `template<typename Derived >`  
`dyn_col_vect< typename Derived::Scalar > qpp::rho2pure` (const Eigen::MatrixBase< Derived > &A)  
*Finds the pure state representation of a matrix proportional to a projector onto a pure state.*
- `template<typename T >`  
`std::vector< T > qpp::complement` (std::vector< T > subsys, idx N)  
*Constructs the complement of a subsystem vector.*
- `template<typename Derived >`  
`std::vector< double > qpp::rho2bloch` (const Eigen::MatrixBase< Derived > &A)  
*Computes the 3-dimensional real Bloch vector corresponding to the qubit density matrix A.*
- `cmat qpp::bloch2rho` (const std::vector< double > &r)  
*Computes the density matrix corresponding to the 3-dimensional real Bloch vector r.*
- `template<char... Bits>`  
`ket qpp::literals::operator"" _ket` ()  
*Multi-partite qubit ket user-defined literal.*
- `template<char... Bits>`  
`bra qpp::literals::operator"" _bra` ()  
*Multi-partite qubit bra user-defined literal.*
- `template<char... Bits>`  
`cmat qpp::literals::operator"" _prj` ()  
*Multi-partite qubit projector user-defined literal.*

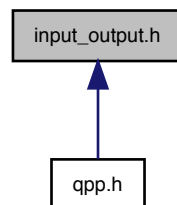
### 8.14.1 Detailed Description

Generic quantum computing functions.

## 8.15 input\_output.h File Reference

Input/output functions.

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



### Namespaces

- [qpp](#)  
*Quantum++ main namespace.*

### Functions

- `template<typename Derived >`  
`internal::IOManipEigen qpp::disp (const Eigen::MatrixBase< Derived > &A, double chop=qpp::chop)`  
*Eigen expression ostream manipulator.*
- `internal::IOManipEigen qpp::disp (cplx z, double chop=qpp::chop)`  
*Complex number ostream manipulator.*
- `template<typename InputIterator >`  
`internal::IOManipRange< InputIterator > qpp::disp (InputIterator first, InputIterator last, const std::string &separator, const std::string &start="[", const std::string &end="]")`  
*Range ostream manipulator.*
- `template<typename Container >`  
`internal::IOManipRange< typename Container::const_iterator > qpp::disp (const Container &c, const std::string &separator, const std::string &start="[", const std::string &end="]", typename std::enable_if< is_iterable< Container >::value >::type != nullptr)`  
*Standard container ostream manipulator. The container must support std::begin(), std::end() and forward iteration.*
- `template<typename PointerType >`  
`internal::IOManipPointer< PointerType > qpp::disp (const PointerType *p, idx N, const std::string &separator, const std::string &start="[", const std::string &end="]")`  
*C-style pointer ostream manipulator.*
- `template<typename Derived >`  
`void qpp::save (const Eigen::MatrixBase< Derived > &A, const std::string &fname)`  
*Saves Eigen expression to a binary file (internal format) in double precision.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::load (const std::string &fname)`  
*Loads Eigen matrix from a binary file (internal format) in double precision.*

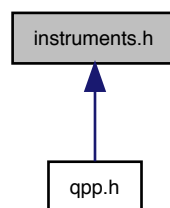
### 8.15.1 Detailed Description

Input/output functions.

## 8.16 instruments.h File Reference

Measurement functions.

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



### Namespaces

- [qpp](#)  
*Quantum++ main namespace.*

### Functions

- `template<typename Derived >`  
`dyn_col_vect< typename Derived::Scalar > qpp::ip (const Eigen::MatrixBase< Derived > &phi, const Eigen::MatrixBase< Derived > &psi, const std::vector< idx > &subsys, const std::vector< idx > &dims)`  
*Generalized inner product.*
- `template<typename Derived >`  
`dyn_col_vect< typename Derived::Scalar > qpp::ip (const Eigen::MatrixBase< Derived > &phi, const Eigen::MatrixBase< Derived > &psi, const std::vector< idx > &subsys, idx d=2)`  
*Generalized inner product.*
- `template<typename Derived >`  
`std::tuple< idx, std::vector< double >, std::vector< cmat > > qpp::measure (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks)`  
*Measures the state A using the set of Kraus operators Ks.*
- `template<typename Derived >`  
`std::tuple< idx, std::vector< double >, std::vector< cmat > > qpp::measure (const Eigen::MatrixBase< Derived > &A, const std::initializer_list< cmat > &Ks)`  
*Measures the state A using the set of Kraus operators Ks.*
- `template<typename Derived >`  
`std::tuple< idx, std::vector< double >, std::vector< cmat > > qpp::measure (const Eigen::MatrixBase< Derived > &A, const cmat &U)`

*Measures the state A in the orthonormal basis specified by the unitary matrix U.*

- `template<typename Derived >`  
`std::tuple< idx, std::vector< double >, std::vector< cmat > > qpp::measure` (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks, const std::vector< idx > &subsys, const std::vector< idx > &dims)

*Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*

- `template<typename Derived >`  
`std::tuple< idx, std::vector< double >, std::vector< cmat > > qpp::measure` (const Eigen::MatrixBase< Derived > &A, const std::initializer\_list< cmat > &Ks, const std::vector< idx > &subsys, const std::vector< idx > &dims)

*Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*

- `template<typename Derived >`  
`std::tuple< idx, std::vector< double >, std::vector< cmat > > qpp::measure` (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks, const std::vector< idx > &subsys, idx d=2)

*Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*

- `template<typename Derived >`  
`std::tuple< idx, std::vector< double >, std::vector< cmat > > qpp::measure` (const Eigen::MatrixBase< Derived > &A, const std::initializer\_list< cmat > &Ks, const std::vector< idx > &subsys, idx d=2)

*Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*

- `template<typename Derived >`  
`std::tuple< idx, std::vector< double >, std::vector< cmat > > qpp::measure` (const Eigen::MatrixBase< Derived > &A, const cmat &V, const std::vector< idx > &subsys, const std::vector< idx > &dims)

*Measures the part subsys of the multi-partite state vector or density matrix A in the orthonormal basis or rank-1 POVM specified by the matrix V.*

- `template<typename Derived >`  
`std::tuple< idx, std::vector< double >, std::vector< cmat > > qpp::measure` (const Eigen::MatrixBase< Derived > &A, const cmat &V, const std::vector< idx > &subsys, idx d=2)

*Measures the part subsys of the multi-partite state vector or density matrix A in the orthonormal basis or rank-1 POVM specified by the matrix V.*

- `template<typename Derived >`  
`std::tuple< std::vector< idx >, double, cmat > qpp::measure_seq` (const Eigen::MatrixBase< Derived > &A, std::vector< idx > subsys, std::vector< idx > dims)

*Sequentially measures the part subsys of the multi-partite state vector or density matrix A in the computational basis.*

- `template<typename Derived >`  
`std::tuple< std::vector< idx >, double, cmat > qpp::measure_seq` (const Eigen::MatrixBase< Derived > &A, std::vector< idx > subsys, idx d=2)

*Sequentially measures the part subsys of the multi-partite state vector or density matrix A in the computational basis.*

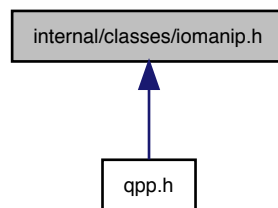
## 8.16.1 Detailed Description

Measurement functions.

## 8.17 internal/classes/iomanip.h File Reference

Input/output manipulators.

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



### Classes

- class [qpp::internal::IOManipRange< InputIterator >](#)
- class [qpp::internal::IOManipPointer< PointerType >](#)
- class [qpp::internal::IOManipEigen](#)

### Namespaces

- [qpp](#)  
*Quantum++ main namespace.*
- [qpp::internal](#)  
*Internal utility functions, do not use them directly or modify them.*

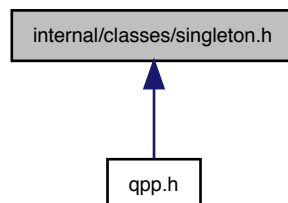
#### 8.17.1 Detailed Description

Input/output manipulators.

## 8.18 internal/classes/singleton.h File Reference

Singleton pattern via CRTP.

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



## Classes

- class [qpp::internal::Singleton< T >](#)  
*Singleton policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern)*

## Namespaces

- [qpp](#)  
*Quantum++ main namespace.*
- [qpp::internal](#)  
*Internal utility functions, do not use them directly or modify them.*

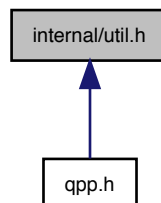
### 8.18.1 Detailed Description

Singleton pattern via CRTP.

## 8.19 internal/util.h File Reference

Internal utility functions.

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



## Classes

- struct [qpp::internal::Display\\_Impl\\_](#)

## Namespaces

- [qpp](#)  
*Quantum++ main namespace.*
- [qpp::internal](#)  
*Internal utility functions, do not use them directly or modify them.*

## Functions

- void [qpp::internal::n2multiidx](#) (idx n, idx numdims, const idx \*const dims, idx \*result) noexcept
- idx [qpp::internal::multiidx2n](#) (const idx \*const midx, idx numdims, const idx \*const dims) noexcept
- template<typename Derived >  
bool [qpp::internal::check\\_square\\_mat](#) (const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >  
bool [qpp::internal::check\\_vector](#) (const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >  
bool [qpp::internal::check\\_rvector](#) (const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >  
bool [qpp::internal::check\\_cvector](#) (const Eigen::MatrixBase< Derived > &A)
- template<typename T >  
bool [qpp::internal::check\\_nonzero\\_size](#) (const T &x) noexcept
- template<typename T1 , typename T2 >  
bool [qpp::internal::check\\_matching\\_sizes](#) (const T1 &lhs, const T2 &rhs) noexcept
- bool [qpp::internal::check\\_dims](#) (const std::vector< idx > &dims)
- template<typename Derived >  
bool [qpp::internal::check\\_dims\\_match\\_mat](#) (const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >  
bool [qpp::internal::check\\_dims\\_match\\_cvect](#) (const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >  
bool [qpp::internal::check\\_dims\\_match\\_rvect](#) (const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &A)
- bool [qpp::internal::check\\_eq\\_dims](#) (const std::vector< idx > &dims, idx dim) noexcept
- bool [qpp::internal::check\\_subsys\\_match\\_dims](#) (const std::vector< idx > &subsys, const std::vector< idx > &dims)
- template<typename Derived >  
bool [qpp::internal::check\\_qubit\\_matrix](#) (const Eigen::MatrixBase< Derived > &A) noexcept
- template<typename Derived >  
bool [qpp::internal::check\\_qubit\\_cvector](#) (const Eigen::MatrixBase< Derived > &A) noexcept
- template<typename Derived >  
bool [qpp::internal::check\\_qubit\\_rvector](#) (const Eigen::MatrixBase< Derived > &A) noexcept
- template<typename Derived >  
bool [qpp::internal::check\\_qubit\\_vector](#) (const Eigen::MatrixBase< Derived > &A) noexcept
- bool [qpp::internal::check\\_perm](#) (const std::vector< idx > &perm)
- template<typename Derived1 , typename Derived2 >  
dyn\_mat< typename Derived1::Scalar > [qpp::internal::kron2](#) (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)
- template<typename Derived1 , typename Derived2 >  
dyn\_mat< typename Derived1::Scalar > [qpp::internal::dirsum2](#) (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)
- template<typename T >  
void [qpp::internal::variadic\\_vector\\_emplace](#) (std::vector< T > &)
- template<typename T , typename First , typename... Args>  
void [qpp::internal::variadic\\_vector\\_emplace](#) (std::vector< T > &v, First &&first, Args &&... args)
- idx [qpp::internal::get\\_num\\_subsys](#) (idx sz, idx d)
- idx [qpp::internal::get\\_dim\\_subsys](#) (idx sz, idx N)

### 8.19.1 Detailed Description

Internal utility functions.

## 8.20 MATLAB/matlab.h File Reference

Input/output interfacing with MATLAB.

```
#include "mat.h"
#include "mex.h"
```

### Namespaces

- [qpp](#)  
*Quantum++ main namespace.*

### Functions

- `template<typename Derived >`  
`std::enable_if< std::is_same< typename Derived::Scalar, cplx >::value, dyn_mat< cplx > >::type qpp::loadMATLAB (const std::string &mat_file, const std::string &var_name)`  
*Loads a complex Eigen dynamic matrix from a MATLAB .mat file,.*
- `template<typename Derived >`  
`std::enable_if<!std::is_same< typename Derived::Scalar, cplx >::value, dyn_mat< typename Derived::Scalar > >::type qpp::loadMATLAB (const std::string &mat_file, const std::string &var_name)`  
*Loads a non-complex Eigen dynamic matrix from a MATLAB .mat file,.*
- `template<typename Derived >`  
`std::enable_if< std::is_same< typename Derived::Scalar, cplx >::value >::type qpp::saveMATLAB (const Eigen::MatrixBase< Derived > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)`  
*Saves a complex Eigen dynamic matrix to a MATLAB .mat file,.*
- `template<typename Derived >`  
`std::enable_if< !std::is_same< typename Derived::Scalar, cplx >::value >::type qpp::saveMATLAB (const Eigen::MatrixBase< Derived > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)`  
*Saves a non-complex Eigen dynamic matrix to a MATLAB .mat file,.*

### 8.20.1 Detailed Description

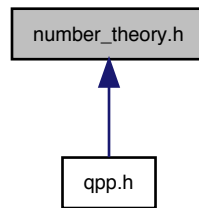
Input/output interfacing with MATLAB.

## 8.21 number\_theory.h File Reference

Number theory functions.



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



## Namespaces

- [qpp](#)  
*Quantum++ main namespace.*

## Functions

- `std::vector< int > qpp::x2contfrac (double x, idx N, idx cut=1e5)`  
*Simple continued fraction expansion.*
- `double qpp::contfrac2x (const std::vector< int > &cf, idx N=idx(-1))`  
*Real representation of a simple continued fraction.*
- `bigint qpp::gcd (bigint a, bigint b)`  
*Greatest common divisor of two integers.*
- `bigint qpp::gcd (const std::vector< bigint > &as)`  
*Greatest common divisor of a list of integers.*
- `bigint qpp::lcm (bigint a, bigint b)`  
*Least common multiple of two integers.*
- `bigint qpp::lcm (const std::vector< bigint > &as)`  
*Least common multiple of a list of integers.*
- `std::vector< idx > qpp::invperm (const std::vector< idx > &perm)`  
*Inverse permutation.*
- `std::vector< idx > qpp::compperm (const std::vector< idx > &perm, const std::vector< idx > &sigma)`  
*Compose permutations.*
- `std::vector< bigint > qpp::factors (bigint a)`  
*Prime factor decomposition.*
- `bigint qpp::modmul (bigint a, bigint b, bigint p)`  
*Modular multiplication without overflow.*
- `bigint qpp::modpow (bigint a, bigint n, bigint p)`  
*Fast integer power modulo p based on the SQUARE-AND-MULTIPLY algorithm.*
- `std::tuple< bigint, bigint, bigint > qpp::egcd (bigint a, bigint b)`  
*Extended greatest common divisor of two integers.*
- `bigint qpp::modinv (bigint a, bigint p)`  
*Modular inverse of a mod p.*
- `bool qpp::isprime (bigint p, idx k=80)`  
*Primality test based on the Miller-Rabin's algorithm.*
- `bigint qpp::randprime (bigint a, bigint b, idx N=1000)`  
*Generates a random big prime uniformly distributed in the interval [a, b].*

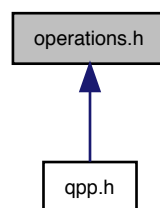
### 8.21.1 Detailed Description

Number theory functions.

## 8.22 operations.h File Reference

Quantum operation functions.

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



### Namespaces

- [qpp](#)  
*Quantum++ main namespace.*

### Functions

- `template<typename Derived1 , typename Derived2 >  
dyn_mat< typename Derived1::Scalar > qpp::applyCTRL (const Eigen::MatrixBase< Derived1 > &state,  
const Eigen::MatrixBase< Derived2 > &A, const std::vector< idx > &ctrl, const std::vector< idx > &subsys,  
const std::vector< idx > &dims)`  
*Applies the controlled-gate A to the part subsys of the multi-partite state vector or density matrix state.*
- `template<typename Derived1 , typename Derived2 >  
dyn_mat< typename Derived1::Scalar > qpp::applyCTRL (const Eigen::MatrixBase< Derived1 > &state,  
const Eigen::MatrixBase< Derived2 > &A, const std::vector< idx > &ctrl, const std::vector< idx > &subsys,  
idx d=2)`  
*Applies the controlled-gate A to the part subsys of the multi-partite state vector or density matrix state.*
- `template<typename Derived1 , typename Derived2 >  
dyn_mat< typename Derived1::Scalar > qpp::apply (const Eigen::MatrixBase< Derived1 > &state, const  
Eigen::MatrixBase< Derived2 > &A, const std::vector< idx > &subsys, const std::vector< idx > &dims)`  
*Applies the gate A to the part subsys of the multi-partite state vector or density matrix state.*
- `template<typename Derived1 , typename Derived2 >  
dyn_mat< typename Derived1::Scalar > qpp::apply (const Eigen::MatrixBase< Derived1 > &state, const  
Eigen::MatrixBase< Derived2 > &A, const std::vector< idx > &subsys, idx d=2)`  
*Applies the gate A to the part subsys of the multi-partite state vector or density matrix state.*

- `template<typename Derived >`  
`cmat qpp::apply` (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks)  
*Applies the channel specified by the set of Kraus operators Ks to the density matrix A.*
- `template<typename Derived >`  
`cmat qpp::apply` (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks, const std::vector< idx > &subsys, const std::vector< idx > &dims)  
*Applies the channel specified by the set of Kraus operators Ks to the part subsys of the multi-partite density matrix A.*
- `template<typename Derived >`  
`cmat qpp::apply` (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks, const std::vector< idx > &subsys, idx d=2)  
*Applies the channel specified by the set of Kraus operators Ks to the part subsys of the multi-partite density matrix A.*
- `cmat qpp::kraus2super` (const std::vector< cmat > &Ks)  
*Superoperator matrix.*
- `cmat qpp::kraus2choi` (const std::vector< cmat > &Ks)  
*Choi matrix.*
- `std::vector< cmat > qpp::choi2kraus` (const cmat &A)  
*Orthogonal Kraus operators from Choi matrix.*
- `cmat qpp::choi2super` (const cmat &A)  
*Converts Choi matrix to superoperator matrix.*
- `cmat qpp::super2choi` (const cmat &A)  
*Converts superoperator matrix to Choi matrix.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::ptrace1` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)  
*Partial trace.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::ptrace1` (const Eigen::MatrixBase< Derived > &A, idx d=2)  
*Partial trace.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::ptrace2` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)  
*Partial trace.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::ptrace2` (const Eigen::MatrixBase< Derived > &A, idx d=2)  
*Partial trace.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::ptrace` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsys, const std::vector< idx > &dims)  
*Partial trace.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::ptrace` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsys, idx d=2)  
*Partial trace.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::ptranspose` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsys, const std::vector< idx > &dims)  
*Partial transpose.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::ptranspose` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsys, idx d=2)  
*Partial transpose.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::syspermute` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &perm, const std::vector< idx > &dims)

*Subsystem permutation.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > qpp::syspermute (const Eigen::MatrixBase< Derived > &A, const`  
`std::vector< idx > &perm, idx d=2)`

*Subsystem permutation.*

### 8.22.1 Detailed Description

Quantum operation functions.

## 8.23 qpp.h File Reference

Quantum++ main header file, includes all other necessary headers.

```
#include <algorithm>
#include <cassert>
#include <chrono>
#include <cmath>
#include <complex>
#include <cstdlib>
#include <cstring>
#include <exception>
#include <fstream>
#include <functional>
#include <initializer_list>
#include <iomanip>
#include <iterator>
#include <limits>
#include <memory>
#include <numeric>
#include <ostream>
#include <random>
#include <sstream>
#include <stdexcept>
#include <string>
#include <tuple>
#include <type_traits>
#include <utility>
#include <vector>
#include <Eigen/Dense>
#include <Eigen/SVD>
#include "types.h"
#include "classes/exception.h"
#include "constants.h"
#include "traits.h"
#include "classes/idevice.h"
#include "internal/util.h"
#include "internal/classes/iomanip.h"
#include "input_output.h"
#include "internal/classes/singleton.h"
#include "classes/init.h"
#include "functions.h"
#include "classes/codes.h"
```

```
#include "classes/gates.h"
#include "classes/states.h"
#include "classes/random_devices.h"
#include "statistics.h"
#include "operations.h"
#include "entropies.h"
#include "entanglement.h"
#include "random.h"
#include "classes/timer.h"
#include "instruments.h"
#include "number_theory.h"
#include "classes/reversible.h"
```

## Namespaces

- [qpp](#)  
*Quantum++ main namespace.*

## Macros

- `#define` [QPP\\_UNUSED\\_](#)

### 8.23.1 Detailed Description

Quantum++ main header file, includes all other necessary headers.

### 8.23.2 Macro Definition Documentation

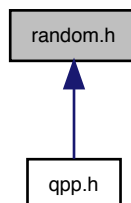
#### 8.23.2.1 QPP\_UNUSED\_

```
#define QPP_UNUSED_
```

## 8.24 random.h File Reference

Randomness-related functions.

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



## Namespaces

- [qpp](#)  
*Quantum++ main namespace.*

## Functions

- double [qpp::rand](#) (double a, double b)  
*Generates a random real number uniformly distributed in the interval [a, b]*
- bigint [qpp::rand](#) (bigint a, bigint b)  
*Generates a random big integer uniformly distributed in the interval [a, b].*
- idx [qpp::randidx](#) (idx a=std::numeric\_limits< idx >::min(), idx b=std::numeric\_limits< idx >::max())  
*Generates a random index (idx) uniformly distributed in the interval [a, b].*
- template<typename Derived >  
Derived [qpp::rand](#) (idx rows, idx cols, double a=0, double b=1)  
*Generates a random matrix with entries uniformly distributed in the interval [a, b]*
- template<>  
dmat [qpp::rand](#) (idx rows, idx cols, double a, double b)  
*Generates a random real matrix with entries uniformly distributed in the interval [a, b], specialization for double matrices ([qpp::dmat](#))*
- template<>  
cmat [qpp::rand](#) (idx rows, idx cols, double a, double b)  
*Generates a random complex matrix with entries (both real and imaginary) uniformly distributed in the interval [a, b], specialization for complex matrices ([qpp::cmat](#))*
- template<typename Derived >  
Derived [qpp::randn](#) (idx rows, idx cols, double mean=0, double sigma=1)  
*Generates a random matrix with entries normally distributed in  $N(\text{mean}, \text{sigma})$*
- template<>  
dmat [qpp::randn](#) (idx rows, idx cols, double mean, double sigma)  
*Generates a random real matrix with entries normally distributed in  $N(\text{mean}, \text{sigma})$ , specialization for double matrices ([qpp::dmat](#))*
- template<>  
cmat [qpp::randn](#) (idx rows, idx cols, double mean, double sigma)  
*Generates a random complex matrix with entries (both real and imaginary) normally distributed in  $N(\text{mean}, \text{sigma})$ , specialization for complex matrices ([qpp::cmat](#))*
- double [qpp::randn](#) (double mean=0, double sigma=1)  
*Generates a random real number (double) normally distributed in  $N(\text{mean}, \text{sigma})$*
- cmat [qpp::randU](#) (idx D=2)  
*Generates a random unitary matrix.*
- cmat [qpp::randV](#) (idx Din, idx Dout)  
*Generates a random isometry matrix.*
- std::vector< cmat > [qpp::randkraus](#) (idx N, idx D=2)  
*Generates a set of random Kraus operators.*
- cmat [qpp::randH](#) (idx D=2)  
*Generates a random Hermitian matrix.*
- ket [qpp::randket](#) (idx D=2)  
*Generates a random normalized ket (pure state vector)*
- cmat [qpp::randrho](#) (idx D=2)  
*Generates a random density matrix.*
- std::vector< idx > [qpp::randperm](#) (idx N)  
*Generates a random uniformly distributed permutation.*
- std::vector< double > [qpp::randprob](#) (idx N)  
*Generates a random probability vector uniformly distributed over the probability simplex.*

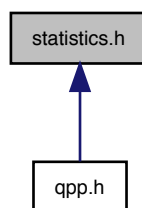
### 8.24.1 Detailed Description

Randomness-related functions.

## 8.25 statistics.h File Reference

Statistics functions.

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



### Namespaces

- [qpp](#)  
*Quantum++ main namespace.*

### Functions

- `std::vector< double > qpp::uniform (idx N)`  
*Uniform probability distribution vector.*
- `std::vector< double > qpp::marginalX (const dmat &probXY)`  
*Marginal distribution.*
- `std::vector< double > qpp::marginalY (const dmat &probXY)`  
*Marginal distribution.*
- `template<typename Container >`  
`double qpp::avg (const std::vector< double > &prob, const Container &X, typename std::enable_if< is_↔`  
`iterable< Container >::value >::type *==nullptr)`  
*Average.*
- `template<typename Container >`  
`double qpp::cov (const dmat &probXY, const Container &X, const Container &Y, typename std::enable_if<`  
`is_iterable< Container >::value >::type *==nullptr)`  
*Covariance.*
- `template<typename Container >`  
`double qpp::var (const std::vector< double > &prob, const Container &X, typename std::enable_if< is_↔`  
`iterable< Container >::value >::type *==nullptr)`  
*Variance.*

- `template<typename Container >`  
`double qpp::sigma (const std::vector< double > &prob, const Container &X, typename std::enable_if< is_↵`  
`iterable< Container >::value >::type !=nullptr)`  
*Standard deviation.*
- `template<typename Container >`  
`double qpp::cor (const dmat &probXY, const Container &X, const Container &Y, typename std::enable_if<`  
`is_iterable< Container >::value >::type !=nullptr)`  
*Correlation.*

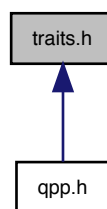
### 8.25.1 Detailed Description

Statistics functions.

## 8.26 traits.h File Reference

Type traits.

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



### Classes

- `struct qpp::make_void< Ts >`  
*Helper for qpp::to\_void<> alias template.*
- `struct qpp::is_iterable< T, typename >`  
*Checks whether T is compatible with an STL-like iterable container.*
- `struct qpp::is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().↵`  
`end()), typename T::value_type > >`  
*Checks whether T is compatible with an STL-like iterable container, specialization for STL-like iterable containers.*
- `struct qpp::is_matrix_expression< Derived >`  
*Checks whether the type is an Eigen matrix expression.*
- `struct qpp::is_complex< T >`  
*Checks whether the type is a complex type.*
- `struct qpp::is_complex< std::complex< T > >`  
*Checks whether the type is a complex number type, specialization for complex types.*



## Namespaces

- [qpp](#)

*Quantum++ main namespace.*

## Typedefs

- `template<typename... Ts>  
using qpp::to\_void = typename make_void< Ts... >::type`

*Alias template that implements the proposal for void\_t.*

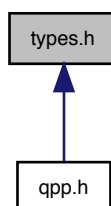
### 8.26.1 Detailed Description

Type traits.

## 8.27 types.h File Reference

Type aliases.

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



## Namespaces

- [qpp](#)

*Quantum++ main namespace.*

## Typedefs

- using `qpp::idx` = `std::size_t`  
*Non-negative integer index.*
- using `qpp::bigint` = `long long int`  
*Big integer.*
- using `qpp::cplx` = `std::complex< double >`  
*Complex number in double precision.*
- using `qpp::ket` = `Eigen::VectorXcd`  
*Complex (double precision) dynamic Eigen column vector.*
- using `qpp::bra` = `Eigen::RowVectorXcd`  
*Complex (double precision) dynamic Eigen row vector.*
- using `qpp::cmat` = `Eigen::MatrixXcd`  
*Complex (double precision) dynamic Eigen matrix.*
- using `qpp::dmat` = `Eigen::MatrixXd`  
*Real (double precision) dynamic Eigen matrix.*
- template<typename Scalar >  
using `qpp::dyn_mat` = `Eigen::Matrix< Scalar, Eigen::Dynamic, Eigen::Dynamic >`  
*Dynamic Eigen matrix over the field specified by Scalar.*
- template<typename Scalar >  
using `qpp::dyn_col_vect` = `Eigen::Matrix< Scalar, Eigen::Dynamic, 1 >`  
*Dynamic Eigen column vector over the field specified by Scalar.*
- template<typename Scalar >  
using `qpp::dyn_row_vect` = `Eigen::Matrix< Scalar, 1, Eigen::Dynamic >`  
*Dynamic Eigen row vector over the field specified by Scalar.*

### 8.27.1 Detailed Description

Type aliases.

## 8.28 /Users/vlad/qpp/README.md File Reference

# Index

/Users/vlad/qpp/README.md, [296](#)

~Codes

qpp::Codes, [132](#)

~Gates

qpp::Gates, [168](#)

~IDisplay

qpp::IDisplay, [177](#)

~Init

qpp::Init, [180](#)

~RandomDevices

qpp::RandomDevices, [233](#)

~Singleton

qpp::internal::Singleton, [236](#)

~States

qpp::States, [241](#)

~Timer

qpp::Timer, [252](#)

A\_

qpp::internal::IOManipEigen, [182](#)

absm

qpp, [28](#)

abssq

qpp, [28](#), [29](#)

adjoint

qpp, [29](#)

all

qpp::Dynamic\_bitset, [151](#)

anticomm

qpp, [30](#)

any

qpp::Dynamic\_bitset, [151](#)

apply

qpp, [30–32](#)

applyCTRL

qpp, [33](#)

avg

qpp, [34](#)

b00

qpp::States, [244](#)

b01

qpp::States, [244](#)

b10

qpp::States, [244](#)

b11

qpp::States, [245](#)

bigint

qpp, [25](#)

Bit\_circuit

qpp::Bit\_circuit, [127](#)

bloch2rho

qpp, [35](#)

bra

qpp, [25](#)

CNOTba

qpp::Gates, [173](#)

CNOT

qpp::Bit\_circuit, [127](#)

qpp::Bit\_circuit::Gate\_count, [164](#)

qpp::Gates, [173](#)

CTRL

qpp::Gates, [168](#)

check\_cvector

qpp::internal, [116](#)

check\_dims

qpp::internal, [117](#)

check\_dims\_match\_cvect

qpp::internal, [117](#)

check\_dims\_match\_mat

qpp::internal, [117](#)

check\_dims\_match\_rvect

qpp::internal, [117](#)

check\_eq\_dims

qpp::internal, [117](#)

check\_matching\_sizes

qpp::internal, [117](#)

check\_nonzero\_size

qpp::internal, [118](#)

check\_perm

qpp::internal, [118](#)

check\_qubit\_cvector

qpp::internal, [118](#)

check\_qubit\_matrix

qpp::internal, [118](#)

check\_qubit\_rvector

qpp::internal, [118](#)

check\_qubit\_vector

qpp::internal, [118](#)

check\_rvector

qpp::internal, [119](#)

check\_square\_mat

qpp::internal, [119](#)

check\_subsys\_match\_dims

qpp::internal, [119](#)

check\_vector

qpp::internal, [119](#)

choi2kraus

qpp, [35](#)

- choi2super
  - qpp, 36
- chop
  - qpp, 112
- chop\_
  - qpp::internal::LOManipEigen, 183
- classes/codes.h, 263
- classes/exception.h, 264
- classes/gates.h, 266
- classes/ideisplay.h, 266
- classes/init.h, 267
- classes/random\_devices.h, 268
- classes/reversible.h, 268
- classes/states.h, 269
- classes/timer.h, 270
- cmat
  - qpp, 26
- Codes
  - qpp::Codes, 132
- codeword
  - qpp::Codes, 132
- comm
  - qpp, 36
- complement
  - qpp, 37
- compperm
  - qpp, 37
- concurrency
  - qpp, 37
- conjugate
  - qpp, 39
- constants.h, 271
- contrac2x
  - qpp, 39
- cor
  - qpp, 40
- cosm
  - qpp, 40
- count
  - qpp::Dynamic\_bitset, 151
- cov
  - qpp, 40
- cplx
  - qpp, 26
- CustomException
  - qpp::exception::CustomException, 134
- cwise
  - qpp, 41
- CZ
  - qpp::Gates, 173
- data
  - qpp::Dynamic\_bitset, 151
- det
  - qpp, 41
- dirsum
  - qpp, 42, 43
- dirsum2
  - qpp::internal, 119
- dirsumpow
  - qpp, 44
- disp
  - qpp, 44–46
- display
  - qpp::Dynamic\_bitset, 152
  - qpp::IDisplay, 178
  - qpp::Timer, 253
  - qpp::internal::LOManipEigen, 182
  - qpp::internal::LOManipPointer, 185
  - qpp::internal::LOManipRange, 188
- display\_impl\_
  - qpp::internal::Display\_Impl\_, 147
- dmat
  - qpp, 26
- dyn\_col\_vect
  - qpp, 26
- dyn\_mat
  - qpp, 26
- dyn\_row\_vect
  - qpp, 27
- Dynamic\_bitset
  - qpp::Dynamic\_bitset, 150
- ee
  - qpp, 112
- egcd
  - qpp, 46
- eig
  - qpp, 47
- end\_
  - qpp::Timer, 255
  - qpp::internal::LOManipPointer, 185
  - qpp::internal::LOManipRange, 189
- entanglement
  - qpp, 47, 48
- entanglement.h, 272
- entropies.h, 273
- entropy
  - qpp, 48, 49
- eps
  - qpp, 112
- evals
  - qpp, 49
- evects
  - qpp, 50
- Exception
  - qpp::exception::Exception, 163
- expandout
  - qpp::Gates, 169, 170
- experimental/experimental.h, 275
- expm
  - qpp, 50
- FRED
  - qpp::Bit\_circuit, 127
  - qpp::Bit\_circuit::Gate\_count, 165
  - qpp::Gates, 173
- factors

- qpp, 50
- Fd
  - qpp::Gates, 171
- first\_
  - qpp::internal::IOManipRange, 189
- flip
  - qpp::Dynamic\_bitset, 153
- functions.h, 275
- funm
  - qpp, 51
- GHZ
  - qpp::States, 245
- gate\_count
  - qpp::Bit\_circuit, 130
- Gates
  - qpp::Gates, 168
- gcd
  - qpp, 51, 52
- gconcurrency
  - qpp, 52
- get
  - qpp::Dynamic\_bitset, 153
- get\_dim\_subsys
  - qpp::internal, 119
- get\_duration
  - qpp::Timer, 253
- get\_instance
  - qpp::internal::Singleton, 236
- get\_num\_subsys
  - qpp::internal, 120
- get\_prng
  - qpp::RandomDevices, 233
- get\_thread\_local\_instance
  - qpp::internal::Singleton, 236
- grams
  - qpp, 53
- H
  - qpp::Gates, 174
- heig
  - qpp, 54
- hevals
  - qpp, 54
- hevects
  - qpp, 55
- IDisplay
  - qpp::IDisplay, 177
- IOManipEigen
  - qpp::internal::IOManipEigen, 182
- IOManipPointer
  - qpp::internal::IOManipPointer, 184, 185
- IOManipRange
  - qpp::internal::IOManipRange, 188
- Id
  - qpp::Gates, 171
- Id2
  - qpp::Gates, 174
- idx
  - qpp, 27
- index\_
  - qpp::Dynamic\_bitset, 154
- infty
  - qpp, 113
- Init
  - qpp::Init, 180
- input\_output.h, 280
- instruments.h, 281
- internal/classes/iomanip.h, 282
- internal/classes/singleton.h, 283
- internal/util.h, 284
- internal::Singleton< const Codes >
  - qpp::Codes, 132
- internal::Singleton< const Gates >
  - qpp::Gates, 173
- internal::Singleton< const Init >
  - qpp::Init, 180
- internal::Singleton< const States >
  - qpp::States, 244
- internal::Singleton< RandomDevices >
  - qpp::RandomDevices, 234
- inverse
  - qpp, 55
- invperm
  - qpp, 55
- ip
  - qpp, 56
- isprime
  - qpp, 57
- jn
  - qpp::States, 242
- ket
  - qpp, 27
- kraus2choi
  - qpp, 57
- kraus2super
  - qpp, 58
- kron
  - qpp, 58–60
- kron2
  - qpp::internal, 120
- kronpow
  - qpp, 60
- last\_
  - qpp::internal::IOManipRange, 189
- lcm
  - qpp, 61
- load
  - qpp, 61
  - qpp::RandomDevices, 233
- loadMATLAB
  - qpp, 62, 63
- logdet
  - qpp, 63

logm  
     qpp, 64  
 lognegativity  
     qpp, 64, 65  
  
 MATLAB/matlab.h, 286  
 marginalX  
     qpp, 65  
 marginalY  
     qpp, 65  
 maxn  
     qpp, 113  
 measure  
     qpp, 66–70  
 measure\_seq  
     qpp, 71, 72  
 mes  
     qpp::States, 242  
 minus  
     qpp::States, 242  
 mket  
     qpp, 72, 73  
 modinv  
     qpp, 73  
 modmul  
     qpp, 74  
 modpow  
     qpp, 74  
 mprj  
     qpp, 75  
 msg\_  
     qpp::exception::Exception, 164  
 multiidx2n  
     qpp, 76  
     qpp::internal, 120  
  
 n2multiidx  
     qpp, 76  
     qpp::internal, 120  
 N\_  
     qpp::Dynamic\_bitset, 160  
     qpp::internal::IOManipPointer, 186  
 NOT  
     qpp::Bit\_circuit, 128  
     qpp::Bit\_circuit::Gate\_count, 165  
 negativity  
     qpp, 77  
 none  
     qpp::Dynamic\_bitset, 154  
 norm  
     qpp, 78  
 number\_theory.h, 286  
  
 offset\_  
     qpp::Dynamic\_bitset, 154  
 omega  
     qpp, 78  
 one  
     qpp::States, 243  
  
 operations.h, 288  
 operator!=  
     qpp::Dynamic\_bitset, 155  
 operator<<  
     qpp::IDisplay, 178  
 operator-  
     qpp::Dynamic\_bitset, 155  
 operator=  
     qpp::IDisplay, 178  
     qpp::Timer, 254  
     qpp::internal::IOManipPointer, 185  
     qpp::internal::IOManipRange, 188  
     qpp::internal::Singleton, 236  
 operator==  
     qpp::Dynamic\_bitset, 155  
 operator""\_bra  
     qpp::literals, 121  
 operator""\_i  
     qpp, 78  
     qpp::literals, 122  
 operator""\_ket  
     qpp::literals, 122  
 operator""\_prj  
     qpp::literals, 122  
  
 p\_  
     qpp::internal::IOManipPointer, 186  
 pGHZ  
     qpp::States, 246  
 pb00  
     qpp::States, 245  
 pb01  
     qpp::States, 245  
 pb10  
     qpp::States, 245  
 pb11  
     qpp::States, 245  
 pi  
     qpp, 113  
 plus  
     qpp::States, 243  
 powm  
     qpp, 78  
 prj  
     qpp, 79  
 prng\_  
     qpp::RandomDevices, 234  
 prod  
     qpp, 79, 80  
 ptrace  
     qpp, 81  
 ptrace1  
     qpp, 82  
 ptrace2  
     qpp, 84  
 ptranspose  
     qpp, 85  
 pW  
     qpp::States, 246

- px0
  - qpp::States, 246
- px1
  - qpp::States, 246
- py0
  - qpp::States, 246
- py1
  - qpp::States, 246
- pz0
  - qpp::States, 247
- pz1
  - qpp::States, 247
- QPP\_UNUSED\_
  - qpp.h, 291
- qmutualinfo
  - qpp, 86
- qpp, 13
  - absm, 28
  - abssq, 28, 29
  - adjoint, 29
  - anticomm, 30
  - apply, 30–32
  - applyCTRL, 33
  - avg, 34
  - bigint, 25
  - bloch2rho, 35
  - bra, 25
  - choi2kraus, 35
  - choi2super, 36
  - chop, 112
  - cmat, 26
  - comm, 36
  - complement, 37
  - comperm, 37
  - concurrence, 37
  - conjugate, 39
  - confrac2x, 39
  - cor, 40
  - cosm, 40
  - cov, 40
  - cplx, 26
  - cwise, 41
  - det, 41
  - dirsum, 42, 43
  - dirsumpow, 44
  - disp, 44–46
  - dmat, 26
  - dyn\_col\_vect, 26
  - dyn\_mat, 26
  - dyn\_row\_vect, 27
  - ee, 112
  - egcd, 46
  - eig, 47
  - entanglement, 47, 48
  - entropy, 48, 49
  - eps, 112
  - evals, 49
  - evects, 50
  - expm, 50
  - factors, 50
  - funm, 51
  - gcd, 51, 52
  - gconcurrency, 52
  - grams, 53
  - heig, 54
  - hevals, 54
  - hevects, 55
  - idx, 27
  - infty, 113
  - inverse, 55
  - invperm, 55
  - ip, 56
  - isprime, 57
  - ket, 27
  - kraus2choi, 57
  - kraus2super, 58
  - kron, 58–60
  - kronpow, 60
  - lcm, 61
  - load, 61
  - loadMATLAB, 62, 63
  - logdet, 63
  - logm, 64
  - lognegativity, 64, 65
  - marginalX, 65
  - marginalY, 65
  - maxn, 113
  - measure, 66–70
  - measure\_seq, 71, 72
  - mket, 72, 73
  - modinv, 73
  - modmul, 74
  - modpow, 74
  - mprj, 75
  - multiidx2n, 76
  - n2multiidx, 76
  - negativity, 77
  - norm, 78
  - omega, 78
  - operator""\_i, 78
  - pi, 113
  - powm, 78
  - prj, 79
  - prod, 79, 80
  - ptrace, 81
  - ptrace1, 82
  - ptrace2, 84
  - ptranspose, 85
  - qmutualinfo, 86
  - rand, 87–89
  - randH, 89
  - randidx, 90
  - randket, 90
  - randkraus, 90
  - randn, 91, 92
  - randperm, 93

- randprime, 93
- randprob, 94
- randrho, 94
- randU, 94
- randV, 95
- renyi, 95, 96
- reshape, 96
- rho2bloch, 97
- rho2pure, 97
- save, 98
- saveMATLAB, 98, 99
- schatten, 99
- schmidtA, 100
- schmidtB, 100, 101
- schmidtcoeffs, 101, 102
- schmidtprobs, 102, 103
- sigma, 103
- sinm, 104
- spectralpowm, 104
- sqrtn, 105
- sum, 105, 106
- super2choi, 106
- svals, 107
- svd, 107
- svdU, 107
- svdV, 108
- syspermute, 108, 109
- to\_void, 27
- trace, 109
- transpose, 109
- tsallis, 110
- uniform, 111
- var, 111
- x2confrac, 112
- qpp.h, 290
  - QPP\_UNUSED\_, 291
- qpp::Bit\_circuit, 125
  - Bit\_circuit, 127
  - CNOT, 127
  - FRED, 127
  - gate\_count, 130
  - NOT, 128
  - reset, 128
  - SWAP, 128
  - TOF, 129
  - X, 129
- qpp::Bit\_circuit::Gate\_count, 164
  - CNOT, 164
  - FRED, 165
  - NOT, 165
  - SWAP, 165
  - TOF, 165
  - X, 165
- qpp::Codes, 130
  - ~Codes, 132
  - Codes, 132
  - codeword, 132
  - internal::Singleton< const Codes >, 132
  - Type, 131
- qpp::Dynamic\_bitset, 148
  - all, 151
  - any, 151
  - count, 151
  - data, 151
  - display, 152
  - Dynamic\_bitset, 150
  - flip, 153
  - get, 153
  - index\_, 154
  - N\_, 160
  - none, 154
  - offset\_, 154
  - operator!=, 155
  - operator-, 155
  - operator==, 155
  - rand, 157
  - reset, 157, 158
  - set, 158
  - size, 159
  - storage\_size, 159
  - storage\_size\_, 160
  - storage\_type, 150
  - to\_string, 159
  - v\_, 160
  - value\_type, 150
- qpp::Gates, 166
  - ~Gates, 168
  - CNOTba, 173
  - CNOT, 173
  - CTRL, 168
  - CZ, 173
  - expandout, 169, 170
  - FRED, 173
  - Fd, 171
  - Gates, 168
  - H, 174
  - Id, 171
  - Id2, 174
  - internal::Singleton< const Gates >, 173
  - Rn, 171
  - S, 174
  - SWAP, 174
  - T, 174
  - TOF, 174
  - X, 175
  - Xd, 172
  - Y, 175
  - Z, 175
  - Zd, 172
- qpp::IDisplay, 176
  - ~IDisplay, 177
  - display, 178
  - IDisplay, 177
  - operator<<, 178
  - operator=, 178
- qpp::Init, 179



- ~Init, 180
  - Init, 180
  - internal::Singleton< const Init >, 180
- qpp::RandomDevices, 231
  - ~RandomDevices, 233
  - get\_prng, 233
  - internal::Singleton< RandomDevices >, 234
  - load, 233
  - prng\_, 234
  - RandomDevices, 232
  - rd\_, 234
  - save, 233
- qpp::States, 239
  - ~States, 241
  - b00, 244
  - b01, 244
  - b10, 244
  - b11, 245
  - GHZ, 245
  - internal::Singleton< const States >, 244
  - jn, 242
  - mes, 242
  - minus, 242
  - one, 243
  - pGHZ, 246
  - pb00, 245
  - pb01, 245
  - pb10, 245
  - pb11, 245
  - plus, 243
  - pW, 246
  - px0, 246
  - px1, 246
  - py0, 246
  - py1, 246
  - pz0, 247
  - pz1, 247
  - States, 241
  - W, 247
  - x0, 247
  - x1, 247
  - y0, 247
  - y1, 248
  - z0, 248
  - z1, 248
  - zero, 243
- qpp::Timer
  - ~Timer, 252
  - display, 253
  - end\_, 255
  - get\_duration, 253
  - operator=, 254
  - start\_, 255
  - tic, 254
  - tics, 254
  - Timer, 252
  - toc, 254
- qpp::Timer< T, CLOCK\_T >, 250
- qpp::exception, 113
- qpp::exception::CustomException, 133
  - CustomException, 134
  - type\_description, 135
  - what\_, 135
- qpp::exception::DimsInvalid, 136
  - type\_description, 137
- qpp::exception::DimsMismatchCvector, 137
  - type\_description, 139
- qpp::exception::DimsMismatchMatrix, 139
  - type\_description, 140
- qpp::exception::DimsMismatchRvector, 141
  - type\_description, 142
- qpp::exception::DimsMismatchVector, 143
  - type\_description, 144
- qpp::exception::DimsNotEqual, 145
  - type\_description, 146
- qpp::exception::Exception, 161
  - Exception, 163
  - msg\_, 164
  - type\_description, 163
  - what, 163
  - where\_, 164
- qpp::exception::MatrixMismatchSubsys, 196
  - type\_description, 197
- qpp::exception::MatrixNotCvector, 197
  - type\_description, 199
- qpp::exception::MatrixNotRvector, 199
  - type\_description, 200
- qpp::exception::MatrixNotSquare, 201
  - type\_description, 202
- qpp::exception::MatrixNotSquareNorCvector, 203
  - type\_description, 204
- qpp::exception::MatrixNotSquareNorRvector, 205
  - type\_description, 206
- qpp::exception::MatrixNotSquareNorVector, 207
  - type\_description, 208
- qpp::exception::MatrixNotVector, 209
  - type\_description, 210
- qpp::exception::NoCodeword, 211
  - type\_description, 212
- qpp::exception::NotBipartite, 213
  - type\_description, 214
- qpp::exception::NotQubitCvector, 215
  - type\_description, 216
- qpp::exception::NotQubitMatrix, 217
  - type\_description, 218
- qpp::exception::NotQubitRvector, 219
  - type\_description, 220
- qpp::exception::NotQubitSubsys, 221
  - type\_description, 222
- qpp::exception::NotQubitVector, 223
  - type\_description, 224
- qpp::exception::OutOfRange, 225
  - type\_description, 226
- qpp::exception::PermInvalid, 227
  - type\_description, 228
- qpp::exception::PermMismatchDims, 229

- type\_description, 230
- qpp::exception::SizeMismatch, 237
  - type\_description, 238
- qpp::exception::SubsysMismatchDims, 249
  - type\_description, 250
- qpp::exception::TypeMismatch, 256
  - type\_description, 257
- qpp::exception::UndefinedType, 257
  - type\_description, 259
- qpp::exception::Unknown, 259
  - type\_description, 260
- qpp::exception::ZeroSize, 261
  - type\_description, 262
- qpp::experimental, 115
- qpp::internal, 115
  - check\_cvector, 116
  - check\_dims, 117
  - check\_dims\_match\_cvect, 117
  - check\_dims\_match\_mat, 117
  - check\_dims\_match\_rvect, 117
  - check\_eq\_dims, 117
  - check\_matching\_sizes, 117
  - check\_nonzero\_size, 118
  - check\_perm, 118
  - check\_qubit\_cvector, 118
  - check\_qubit\_matrix, 118
  - check\_qubit\_rvector, 118
  - check\_qubit\_vector, 118
  - check\_rvector, 119
  - check\_square\_mat, 119
  - check\_subsys\_match\_dims, 119
  - check\_vector, 119
  - dirsum2, 119
  - get\_dim\_subsys, 119
  - get\_num\_subsys, 120
  - kron2, 120
  - multiidx2n, 120
  - n2multiidx, 120
  - variadic\_vector\_emplace, 120
- qpp::internal::Display\_Impl\_, 147
  - display\_impl\_, 147
- qpp::internal::IOManipEigen, 181
  - A\_, 182
  - chop\_, 183
  - display, 182
  - IOManipEigen, 182
- qpp::internal::IOManipPointer
  - display, 185
  - end\_, 185
  - IOManipPointer, 184, 185
  - N\_, 186
  - operator=, 185
  - p\_, 186
  - separator\_, 186
  - start\_, 186
- qpp::internal::IOManipPointer< PointerType >, 183
- qpp::internal::IOManipRange
  - display, 188
  - end\_, 189
  - first\_, 189
  - IOManipRange, 188
  - last\_, 189
  - operator=, 188
  - separator\_, 189
  - start\_, 189
- qpp::internal::IOManipRange< InputIterator >, 187
- qpp::internal::Singleton
  - ~Singleton, 236
  - get\_instance, 236
  - get\_thread\_local\_instance, 236
  - operator=, 236
  - Singleton, 236
- qpp::internal::Singleton< T >, 235
- qpp::is\_complex< std::complex< T > >, 191
- qpp::is\_complex< T >, 190
- qpp::is\_iterable< T, to\_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().begin()), typename T::value\_type > >, 193
- qpp::is\_iterable< T, typename >, 192
- qpp::is\_matrix\_expression< Derived >, 194
- qpp::literals, 121
  - operator""\_bra, 121
  - operator""\_i, 122
  - operator""\_ket, 122
  - operator""\_prj, 122
- qpp::make\_void
  - type, 195
- qpp::make\_void< Ts >, 195
- rand
  - qpp, 87–89
  - qpp::Dynamic\_bitset, 157
- randH
  - qpp, 89
- randidx
  - qpp, 90
- randket
  - qpp, 90
- randkraus
  - qpp, 90
- randn
  - qpp, 91, 92
- random.h, 291
- RandomDevices
  - qpp::RandomDevices, 232
- randperm
  - qpp, 93
- randprime
  - qpp, 93
- randprob
  - qpp, 94
- randrho
  - qpp, 94
- randU
  - qpp, 94
- randV
  - qpp, 95

- rd\_
  - qpp::RandomDevices, 234
- renyi
  - qpp, 95, 96
- reset
  - qpp::Bit\_circuit, 128
  - qpp::Dynamic\_bitset, 157, 158
- reshape
  - qpp, 96
- rho2bloch
  - qpp, 97
- rho2pure
  - qpp, 97
- Rn
  - qpp::Gates, 171
- S
  - qpp::Gates, 174
- SWAP
  - qpp::Bit\_circuit, 128
  - qpp::Bit\_circuit::Gate\_count, 165
  - qpp::Gates, 174
- save
  - qpp, 98
  - qpp::RandomDevices, 233
- saveMATLAB
  - qpp, 98, 99
- schatten
  - qpp, 99
- schmidtA
  - qpp, 100
- schmidtB
  - qpp, 100, 101
- schmidtcoeffs
  - qpp, 101, 102
- schmidtprobs
  - qpp, 102, 103
- separator\_
  - qpp::internal::IOManipPointer, 186
  - qpp::internal::IOManipRange, 189
- set
  - qpp::Dynamic\_bitset, 158
- sigma
  - qpp, 103
- Singleton
  - qpp::internal::Singleton, 236
- sinm
  - qpp, 104
- size
  - qpp::Dynamic\_bitset, 159
- spectralpowm
  - qpp, 104
- sqrtn
  - qpp, 105
- start\_
  - qpp::Timer, 255
  - qpp::internal::IOManipPointer, 186
  - qpp::internal::IOManipRange, 189
- States
  - qpp::States, 241
- statistics.h, 293
- storage\_size
  - qpp::Dynamic\_bitset, 159
- storage\_size\_
  - qpp::Dynamic\_bitset, 160
- storage\_type
  - qpp::Dynamic\_bitset, 150
- sum
  - qpp, 105, 106
- super2choi
  - qpp, 106
- svals
  - qpp, 107
- svd
  - qpp, 107
- svdU
  - qpp, 107
- svdV
  - qpp, 108
- syspermute
  - qpp, 108, 109
- T
  - qpp::Gates, 174
- TOF
  - qpp::Bit\_circuit, 129
  - qpp::Bit\_circuit::Gate\_count, 165
  - qpp::Gates, 174
- tic
  - qpp::Timer, 254
- tics
  - qpp::Timer, 254
- Timer
  - qpp::Timer, 252
- to\_string
  - qpp::Dynamic\_bitset, 159
- to\_void
  - qpp, 27
- toc
  - qpp::Timer, 254
- trace
  - qpp, 109
- traits.h, 294
- transpose
  - qpp, 109
- tsallis
  - qpp, 110
- Type
  - qpp::Codes, 131
- type
  - qpp::make\_void, 195
- type\_description
  - qpp::exception::CustomException, 135
  - qpp::exception::DimsInvalid, 137
  - qpp::exception::DimsMismatchCvector, 139
  - qpp::exception::DimsMismatchMatrix, 140
  - qpp::exception::DimsMismatchRvector, 142
  - qpp::exception::DimsMismatchVector, 144

- qpp::exception::DimsNotEqual, [146](#)
- qpp::exception::Exception, [163](#)
- qpp::exception::MatrixMismatchSubsys, [197](#)
- qpp::exception::MatrixNotCvector, [199](#)
- qpp::exception::MatrixNotRvector, [200](#)
- qpp::exception::MatrixNotSquare, [202](#)
- qpp::exception::MatrixNotSquareNorCvector, [204](#)
- qpp::exception::MatrixNotSquareNorRvector, [206](#)
- qpp::exception::MatrixNotSquareNorVector, [208](#)
- qpp::exception::MatrixNotVector, [210](#)
- qpp::exception::NoCodeword, [212](#)
- qpp::exception::NotBipartite, [214](#)
- qpp::exception::NotQubitCvector, [216](#)
- qpp::exception::NotQubitMatrix, [218](#)
- qpp::exception::NotQubitRvector, [220](#)
- qpp::exception::NotQubitSubsys, [222](#)
- qpp::exception::NotQubitVector, [224](#)
- qpp::exception::OutOfRange, [226](#)
- qpp::exception::PermInvalid, [228](#)
- qpp::exception::PermMismatchDims, [230](#)
- qpp::exception::SizeMismatch, [238](#)
- qpp::exception::SubsysMismatchDims, [250](#)
- qpp::exception::TypeMismatch, [257](#)
- qpp::exception::UndefinedType, [259](#)
- qpp::exception::Unknown, [260](#)
- qpp::exception::ZeroSize, [262](#)
- types.h, [295](#)
- uniform
  - qpp, [111](#)
- v\_
  - qpp::Dynamic\_bitset, [160](#)
- value\_type
  - qpp::Dynamic\_bitset, [150](#)
- var
  - qpp, [111](#)
- variadic\_vector\_emplace
  - qpp::internal, [120](#)
- W
  - qpp::States, [247](#)
- what
  - qpp::exception::Exception, [163](#)
- what\_
  - qpp::exception::CustomException, [135](#)
- where\_
  - qpp::exception::Exception, [164](#)
- X
  - qpp::Bit\_circuit, [129](#)
  - qpp::Bit\_circuit::Gate\_count, [165](#)
  - qpp::Gates, [175](#)
- x0
  - qpp::States, [247](#)
- x1
  - qpp::States, [247](#)
- x2contfrac
  - qpp, [112](#)
- Xd
  - qpp::Gates, [172](#)
- Y
  - qpp::Gates, [175](#)
- y0
  - qpp::States, [247](#)
- y1
  - qpp::States, [248](#)
- Z
  - qpp::Gates, [175](#)
- z0
  - qpp::States, [248](#)
- z1
  - qpp::States, [248](#)
- Zd
  - qpp::Gates, [172](#)
- zero
  - qpp::States, [243](#)