

Quantum++  
v1.0.0-devel

Generated by Doxygen 1.8.10

Fri Nov 4 2016 00:16:49



# Contents

<b>1</b>	<b>Quantum++</b>	<b>1</b>
<b>2</b>	<b>Namespace Index</b>	<b>5</b>
2.1	Namespace List . . . . .	5
<b>3</b>	<b>Hierarchical Index</b>	<b>7</b>
3.1	Class Hierarchy . . . . .	7
<b>4</b>	<b>Class Index</b>	<b>9</b>
4.1	Class List . . . . .	9
<b>5</b>	<b>File Index</b>	<b>13</b>
5.1	File List . . . . .	13
<b>6</b>	<b>Namespace Documentation</b>	<b>15</b>
6.1	qpp Namespace Reference . . . . .	15
6.1.1	Detailed Description . . . . .	27
6.1.2	Typedef Documentation . . . . .	27
6.1.2.1	bigint . . . . .	27
6.1.2.2	bra . . . . .	27
6.1.2.3	cmat . . . . .	27
6.1.2.4	cplx . . . . .	27
6.1.2.5	dmat . . . . .	27
6.1.2.6	dyn_col_vect . . . . .	28
6.1.2.7	dyn_mat . . . . .	28
6.1.2.8	dyn_row_vect . . . . .	28
6.1.2.9	idx . . . . .	28
6.1.2.10	ket . . . . .	28
6.1.2.11	to_void . . . . .	28
6.1.3	Function Documentation . . . . .	28
6.1.3.1	absm(const Eigen::MatrixBase< Derived > &A) . . . . .	28
6.1.3.2	abssq(InputIterator first, InputIterator last) . . . . .	29

6.1.3.3	<code>abssq(const Container &amp;c, typename std::enable_if&lt; is_iterable&lt; Container &gt;::value &gt;::type !=nullptr)</code> . . . . .	29
6.1.3.4	<code>abssq(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	29
6.1.3.5	<code>adjoint(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	29
6.1.3.6	<code>anticomm(const Eigen::MatrixBase&lt; Derived1 &gt; &amp;A, const Eigen::MatrixBase&lt; Derived2 &gt; &amp;B)</code> . . . . .	30
6.1.3.7	<code>apply(const Eigen::MatrixBase&lt; Derived1 &gt; &amp;state, const Eigen::MatrixBase&lt; Derived2 &gt; &amp;A, const std::vector&lt; idx &gt; &amp;subsys, const std::vector&lt; idx &gt; &amp;dims)</code> . . . . .	30
6.1.3.8	<code>apply(const Eigen::MatrixBase&lt; Derived1 &gt; &amp;state, const Eigen::MatrixBase&lt; Derived2 &gt; &amp;A, const std::vector&lt; idx &gt; &amp;subsys, idx d=2)</code> . . . . .	30
6.1.3.9	<code>apply(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; cmat &gt; &amp;Ks)</code> . . . . .	31
6.1.3.10	<code>apply(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; cmat &gt; &amp;Ks, const std::vector&lt; idx &gt; &amp;subsys, const std::vector&lt; idx &gt; &amp;dims)</code> . . . . .	31
6.1.3.11	<code>apply(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; cmat &gt; &amp;Ks, const std::vector&lt; idx &gt; &amp;subsys, idx d=2)</code> . . . . .	31
6.1.3.12	<code>applyCTRL(const Eigen::MatrixBase&lt; Derived1 &gt; &amp;state, const Eigen::MatrixBase&lt; Derived2 &gt; &amp;A, const std::vector&lt; idx &gt; &amp;ctrl, const std::vector&lt; idx &gt; &amp;subsys, const std::vector&lt; idx &gt; &amp;dims)</code> . . . . .	32
6.1.3.13	<code>applyCTRL(const Eigen::MatrixBase&lt; Derived1 &gt; &amp;state, const Eigen::MatrixBase&lt; Derived2 &gt; &amp;A, const std::vector&lt; idx &gt; &amp;ctrl, const std::vector&lt; idx &gt; &amp;subsys, idx d=2)</code> . . . . .	32
6.1.3.14	<code>avg(const std::vector&lt; double &gt; &amp;prob, const Container &amp;X, typename std::enable_if&lt; is_iterable&lt; Container &gt;::value &gt;::type !=nullptr)</code> . . . . .	33
6.1.3.15	<code>bloch2rho(const std::vector&lt; double &gt; &amp;r)</code> . . . . .	33
6.1.3.16	<code>choi2kraus(const cmat &amp;A)</code> . . . . .	33
6.1.3.17	<code>choi2super(const cmat &amp;A)</code> . . . . .	34
6.1.3.18	<code>comm(const Eigen::MatrixBase&lt; Derived1 &gt; &amp;A, const Eigen::MatrixBase&lt; Derived2 &gt; &amp;B)</code> . . . . .	34
6.1.3.19	<code>complement(std::vector&lt; T &gt; subsys, idx N)</code> . . . . .	34
6.1.3.20	<code>compperm(const std::vector&lt; idx &gt; &amp;perm, const std::vector&lt; idx &gt; &amp;sigma)</code> . . . . .	35
6.1.3.21	<code>concurrence(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	36
6.1.3.22	<code>conjugate(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	36
6.1.3.23	<code>contrac2x(const std::vector&lt; int &gt; &amp;cf, idx N=idx(-1))</code> . . . . .	36
6.1.3.24	<code>cor(const dmat &amp;probXY, const Container &amp;X, const Container &amp;Y, typename std::enable_if&lt; is_iterable&lt; Container &gt;::value &gt;::type !=nullptr)</code> . . . . .	36
6.1.3.25	<code>cosm(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	37
6.1.3.26	<code>cov(const dmat &amp;probXY, const Container &amp;X, const Container &amp;Y, typename std::enable_if&lt; is_iterable&lt; Container &gt;::value &gt;::type !=nullptr)</code> . . . . .	37
6.1.3.27	<code>cwise(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, OutputScalar(*f)( const typename Derived::Scalar &amp;))</code> . . . . .	37
6.1.3.28	<code>det(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	37
6.1.3.29	<code>dirsum(const T &amp;head)</code> . . . . .	38
6.1.3.30	<code>dirsum(const T &amp;head, const Args &amp;...tail)</code> . . . . .	38
6.1.3.31	<code>dirsum(const std::vector&lt; Derived &gt; &amp;As)</code> . . . . .	38

6.1.3.32	<code>dirsum(const std::initializer_list&lt; Derived &gt; &amp;As)</code>	39
6.1.3.33	<code>dirsumpow(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, idx n)</code>	39
6.1.3.34	<code>disp(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, double chop=qpp::chop)</code>	39
6.1.3.35	<code>disp(cplx z, double chop=qpp::chop)</code>	40
6.1.3.36	<code>disp(InputIterator first, InputIterator last, const std::string &amp;separator, const std::string &amp;start="["", const std::string &amp;end=""]")</code>	40
6.1.3.37	<code>disp(const Container &amp;c, const std::string &amp;separator, const std::string &amp;start="["", const std::string &amp;end=""]", typename std::enable_if&lt; is_iterable&lt; Container &gt;::value &gt;::type != nullptr)</code>	40
6.1.3.38	<code>disp(const PointerType *p, idx N, const std::string &amp;separator, const std::string &amp;start="["", const std::string &amp;end=""]")</code>	41
6.1.3.39	<code>egcd(bigint a, bigint b)</code>	42
6.1.3.40	<code>eig(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	42
6.1.3.41	<code>entanglement(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;dims)</code>	42
6.1.3.42	<code>entanglement(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, idx d=2)</code>	43
6.1.3.43	<code>entropy(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	43
6.1.3.44	<code>entropy(const std::vector&lt; double &gt; &amp;prob)</code>	43
6.1.3.45	<code>evals(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	43
6.1.3.46	<code>evecs(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	44
6.1.3.47	<code>expm(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	44
6.1.3.48	<code>factors(bigint a)</code>	44
6.1.3.49	<code>funm(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, cplx(*f)(const cplx &amp;))</code>	44
6.1.3.50	<code>gcd(bigint a, bigint b)</code>	45
6.1.3.51	<code>gcd(const std::vector&lt; bigint &gt; &amp;as)</code>	45
6.1.3.52	<code>gconcurrence(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	45
6.1.3.53	<code>grams(const std::vector&lt; Derived &gt; &amp;As)</code>	46
6.1.3.54	<code>grams(const std::initializer_list&lt; Derived &gt; &amp;As)</code>	46
6.1.3.55	<code>grams(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	46
6.1.3.56	<code>heig(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	46
6.1.3.57	<code>hevals(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	47
6.1.3.58	<code>hevecs(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	47
6.1.3.59	<code>inverse(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	47
6.1.3.60	<code>invperm(const std::vector&lt; idx &gt; &amp;perm)</code>	47
6.1.3.61	<code>ip(const Eigen::MatrixBase&lt; Derived &gt; &amp;phi, const Eigen::MatrixBase&lt; Derived &gt; &amp;psi, const std::vector&lt; idx &gt; &amp;subsys, const std::vector&lt; idx &gt; &amp;dims)</code>	48
6.1.3.62	<code>ip(const Eigen::MatrixBase&lt; Derived &gt; &amp;phi, const Eigen::MatrixBase&lt; Derived &gt; &amp;psi, const std::vector&lt; idx &gt; &amp;subsys, idx d=2)</code>	48
6.1.3.63	<code>isprime(bigint p, idx k=80)</code>	48
6.1.3.64	<code>kraus2choi(const std::vector&lt; cmat &gt; &amp;Ks)</code>	49
6.1.3.65	<code>kraus2super(const std::vector&lt; cmat &gt; &amp;Ks)</code>	49
6.1.3.66	<code>kron(const T &amp;head)</code>	49

6.1.3.67	<code>kron(const T &amp;head, const Args &amp;...tail)</code>	50
6.1.3.68	<code>kron(const std::vector&lt; Derived &gt; &amp;As)</code>	50
6.1.3.69	<code>kron(const std::initializer_list&lt; Derived &gt; &amp;As)</code>	50
6.1.3.70	<code>kronpow(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, idx n)</code>	51
6.1.3.71	<code>lcm(bigint a, bigint b)</code>	51
6.1.3.72	<code>lcm(const std::vector&lt; bigint &gt; &amp;as)</code>	51
6.1.3.73	<code>load(const std::string &amp;fname)</code>	51
6.1.3.74	<code>loadMATLAB(const std::string &amp;mat_file, const std::string &amp;var_name)</code>	52
6.1.3.75	<code>loadMATLAB(const std::string &amp;mat_file, const std::string &amp;var_name)</code>	52
6.1.3.76	<code>logdet(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	53
6.1.3.77	<code>logm(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	53
6.1.3.78	<code>lognegativity(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;dims)</code>	53
6.1.3.79	<code>lognegativity(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, idx d=2)</code>	53
6.1.3.80	<code>marginalX(const dmat &amp;probXY)</code>	54
6.1.3.81	<code>marginalY(const dmat &amp;probXY)</code>	54
6.1.3.82	<code>measure(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; cmat &gt; &amp;Ks)</code>	54
6.1.3.83	<code>measure(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::initializer_list&lt; cmat &gt; &amp;Ks)</code>	54
6.1.3.84	<code>measure(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const cmat &amp;U)</code>	55
6.1.3.85	<code>measure(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; cmat &gt; &amp;Ks, const std::vector&lt; idx &gt; &amp;subsys, const std::vector&lt; idx &gt; &amp;dims)</code>	55
6.1.3.86	<code>measure(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::initializer_list&lt; cmat &gt; &amp;Ks, const std::vector&lt; idx &gt; &amp;subsys, const std::vector&lt; idx &gt; &amp;dims)</code>	55
6.1.3.87	<code>measure(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; cmat &gt; &amp;Ks, const std::vector&lt; idx &gt; &amp;subsys, idx d=2)</code>	56
6.1.3.88	<code>measure(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::initializer_list&lt; cmat &gt; &amp;Ks, const std::vector&lt; idx &gt; &amp;subsys, idx d=2)</code>	56
6.1.3.89	<code>measure(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const cmat &amp;V, const std::vector&lt; idx &gt; &amp;subsys, const std::vector&lt; idx &gt; &amp;dims)</code>	57
6.1.3.90	<code>measure(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const cmat &amp;V, const std::vector&lt; idx &gt; &amp;subsys, idx d=2)</code>	57
6.1.3.91	<code>measure_seq(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, std::vector&lt; idx &gt; &amp;subsys, std::vector&lt; idx &gt; &amp;dims)</code>	58
6.1.3.92	<code>measure_seq(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, std::vector&lt; idx &gt; &amp;subsys, idx d=2)</code>	58
6.1.3.93	<code>mket(const std::vector&lt; idx &gt; &amp;mask, const std::vector&lt; idx &gt; &amp;dims)</code>	59
6.1.3.94	<code>mket(const std::vector&lt; idx &gt; &amp;mask, idx d=2)</code>	59
6.1.3.95	<code>modinv(bigint a, bigint p)</code>	59
6.1.3.96	<code>modmul(bigint a, bigint b, bigint p)</code>	60
6.1.3.97	<code>modpow(bigint a, bigint n, bigint p)</code>	60
6.1.3.98	<code>mprj(const std::vector&lt; idx &gt; &amp;mask, const std::vector&lt; idx &gt; &amp;dims)</code>	60

6.1.3.99	<code>mprj(const std::vector&lt; idx &gt; &amp;mask, idx d=2)</code>	61
6.1.3.100	<code>multiidx2n(const std::vector&lt; idx &gt; &amp;midx, const std::vector&lt; idx &gt; &amp;dims)</code>	61
6.1.3.101	<code>n2multiidx(idx n, const std::vector&lt; idx &gt; &amp;dims)</code>	61
6.1.3.102	<code>negativity(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;dims)</code>	62
6.1.3.103	<code>negativity(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, idx d=2)</code>	62
6.1.3.104	<code>norm(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	62
6.1.3.105	<code>omega(idx D)</code>	62
6.1.3.106	<code>operator""_i(unsigned long long int x) noexcept</code>	63
6.1.3.107	<code>operator""_i(long double x) noexcept</code>	63
6.1.3.108	<code>powm(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, idx n)</code>	63
6.1.3.109	<code>prj(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	63
6.1.3.110	<code>prod(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	64
6.1.3.111	<code>prod(InputIterator first, InputIterator last)</code>	64
6.1.3.112	<code>prod(const Container &amp;c, typename std::enable_if&lt; is_iterable&lt; Container &gt;::value &gt;::type * = nullptr)</code>	64
6.1.3.113	<code>ptrace(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;subsys, const std::vector&lt; idx &gt; &amp;dims)</code>	64
6.1.3.114	<code>ptrace(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;subsys, idx d=2)</code>	65
6.1.3.115	<code>ptrace1(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;dims)</code>	65
6.1.3.116	<code>ptrace1(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, idx d=2)</code>	66
6.1.3.117	<code>ptrace2(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;dims)</code>	66
6.1.3.118	<code>ptrace2(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, idx d=2)</code>	66
6.1.3.119	<code>ptranspose(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;subsys, const std::vector&lt; idx &gt; &amp;dims)</code>	67
6.1.3.120	<code>ptranspose(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;subsys, idx d=2)</code>	67
6.1.3.121	<code>qmutualinfo(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;subsysA, const std::vector&lt; idx &gt; &amp;subsysB, const std::vector&lt; idx &gt; &amp;dims)</code>	67
6.1.3.122	<code>qmutualinfo(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;subsysA, const std::vector&lt; idx &gt; &amp;subsysB, idx d=2)</code>	68
6.1.3.123	<code>rand(double a, double b)</code>	68
6.1.3.124	<code>rand(bigint a, bigint b)</code>	68
6.1.3.125	<code>rand(idx rows, idx cols, double a=0, double b=1)</code>	69
6.1.3.126	<code>rand(idx rows, idx cols, double a, double b)</code>	69
6.1.3.127	<code>rand(idx rows, idx cols, double a, double b)</code>	69
6.1.3.128	<code>randH(idx D=2)</code>	69
6.1.3.129	<code>randidx(idx a=std::numeric_limits&lt; idx &gt;::min(), idx b=std::numeric_limits&lt; idx &gt;::max())</code>	70
6.1.3.130	<code>randket(idx D=2)</code>	70
6.1.3.131	<code>randkraus(idx N, idx D=2)</code>	70
6.1.3.132	<code>randn(idx rows, idx cols, double mean=0, double sigma=1)</code>	70

6.1.3.133 randn(idx rows, idx cols, double mean, double sigma) . . . . .	71
6.1.3.134 randn(idx rows, idx cols, double mean, double sigma) . . . . .	71
6.1.3.135 randn(double mean=0, double sigma=1) . . . . .	71
6.1.3.136 randperm(idx N) . . . . .	72
6.1.3.137 randprime(bigint a, bigint b, idx N=1000) . . . . .	72
6.1.3.138 randprob(idx N) . . . . .	72
6.1.3.139 randrho(idx D=2) . . . . .	72
6.1.3.140 randU(idx D=2) . . . . .	73
6.1.3.141 randV(idx Din, idx Dout) . . . . .	73
6.1.3.142 renyi(const Eigen::MatrixBase< Derived > &A, double alpha) . . . . .	73
6.1.3.143 renyi(const std::vector< double > &prob, double alpha) . . . . .	73
6.1.3.144 reshape(const Eigen::MatrixBase< Derived > &A, idx rows, idx cols) . . . . .	74
6.1.3.145 rho2bloch(const Eigen::MatrixBase< Derived > &A) . . . . .	74
6.1.3.146 rho2pure(const Eigen::MatrixBase< Derived > &A) . . . . .	74
6.1.3.147 save(const Eigen::MatrixBase< Derived > &A, const std::string &fname) . . . . .	75
6.1.3.148 saveMATLAB(const Eigen::MatrixBase< Derived > &A, const std::string &mat↵ _file, const std::string &var_name, const std::string &mode) . . . . .	75
6.1.3.149 saveMATLAB(const Eigen::MatrixBase< Derived > &A, const std::string &mat↵ _file, const std::string &var_name, const std::string &mode) . . . . .	75
6.1.3.150 schatten(const Eigen::MatrixBase< Derived > &A, double p) . . . . .	76
6.1.3.151 schmidtA(const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims) . . . . .	76
6.1.3.152 schmidtA(const Eigen::MatrixBase< Derived > &A, idx d=2) . . . . .	76
6.1.3.153 schmidtB(const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims) . . . . .	76
6.1.3.154 schmidtB(const Eigen::MatrixBase< Derived > &A, idx d=2) . . . . .	77
6.1.3.155 schmidtcoeffs(const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims) . . . . .	77
6.1.3.156 schmidtcoeffs(const Eigen::MatrixBase< Derived > &A, idx d=2) . . . . .	77
6.1.3.157 schmidtprobs(const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims) . . . . .	78
6.1.3.158 schmidtprobs(const Eigen::MatrixBase< Derived > &A, idx d=2) . . . . .	78
6.1.3.159 sigma(const std::vector< double > &prob, const Container &X, typename std↵ ::enable_if< is_iterable< Container >::value >::type !=nullptr) . . . . .	78
6.1.3.160 sinm(const Eigen::MatrixBase< Derived > &A) . . . . .	79
6.1.3.161 spectralpowm(const Eigen::MatrixBase< Derived > &A, const cplx z) . . . . .	79
6.1.3.162 sqrtm(const Eigen::MatrixBase< Derived > &A) . . . . .	79
6.1.3.163 sum(const Eigen::MatrixBase< Derived > &A) . . . . .	79
6.1.3.164 sum(InputIterator first, InputIterator last) . . . . .	80
6.1.3.165 sum(const Container &c, typename std::enable_if< is_iterable< Container >↵ ::value >::type !=nullptr) . . . . .	80
6.1.3.166 super2choi(const cmat &A) . . . . .	80



6.1.3.167	svals(const Eigen::MatrixBase< Derived > &A)	80
6.1.3.168	svd(const Eigen::MatrixBase< Derived > &A)	81
6.1.3.169	svdU(const Eigen::MatrixBase< Derived > &A)	81
6.1.3.170	svdV(const Eigen::MatrixBase< Derived > &A)	81
6.1.3.171	syspermute(const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &perm, const std::vector< idx > &dims)	81
6.1.3.172	syspermute(const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &perm, idx d=2)	82
6.1.3.173	trace(const Eigen::MatrixBase< Derived > &A)	82
6.1.3.174	transpose(const Eigen::MatrixBase< Derived > &A)	82
6.1.3.175	tsallis(const Eigen::MatrixBase< Derived > &A, double q)	82
6.1.3.176	tsallis(const std::vector< double > &prob, double q)	83
6.1.3.177	uniform(idx N)	83
6.1.3.178	var(const std::vector< double > &prob, const Container &X, typename std::enable_if< is_iterable< Container >::value >::type != nullptr)	83
6.1.3.179	x2contfrac(double x, idx N, idx cut=1e5)	83
6.1.4	Variable Documentation	84
6.1.4.1	chop	84
6.1.4.2	ee	84
6.1.4.3	eps	84
6.1.4.4	infty	84
6.1.4.5	maxn	84
6.1.4.6	pi	84
6.2	qpp::exception Namespace Reference	84
6.2.1	Detailed Description	86
6.3	qpp::experimental Namespace Reference	86
6.3.1	Detailed Description	86
6.4	qpp::internal Namespace Reference	86
6.4.1	Detailed Description	87
6.4.2	Function Documentation	87
6.4.2.1	check_cvector(const Eigen::MatrixBase< Derived > &A)	87
6.4.2.2	check_dims(const std::vector< idx > &dims)	87
6.4.2.3	check_dims_match_cvect(const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &A)	87
6.4.2.4	check_dims_match_mat(const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &A)	87
6.4.2.5	check_dims_match_rvect(const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &A)	87
6.4.2.6	check_eq_dims(const std::vector< idx > &dims, idx dim) noexcept	88
6.4.2.7	check_matching_sizes(const T1 &lhs, const T2 &rhs) noexcept	88
6.4.2.8	check_nonzero_size(const T &x) noexcept	88
6.4.2.9	check_perm(const std::vector< idx > &perm)	88

6.4.2.10	<a href="#">check_qubit_cvector(const Eigen::MatrixBase&lt; Derived &gt; &amp;A) noexcept</a>	88
6.4.2.11	<a href="#">check_qubit_matrix(const Eigen::MatrixBase&lt; Derived &gt; &amp;A) noexcept</a>	88
6.4.2.12	<a href="#">check_qubit_rvector(const Eigen::MatrixBase&lt; Derived &gt; &amp;A) noexcept</a>	88
6.4.2.13	<a href="#">check_qubit_vector(const Eigen::MatrixBase&lt; Derived &gt; &amp;A) noexcept</a>	88
6.4.2.14	<a href="#">check_rvector(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</a>	88
6.4.2.15	<a href="#">check_square_mat(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</a>	88
6.4.2.16	<a href="#">check_subsys_match_dims(const std::vector&lt; idx &gt; &amp;subsys, const std::vector&lt; idx &gt; &amp;dims)</a>	88
6.4.2.17	<a href="#">check_vector(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</a>	88
6.4.2.18	<a href="#">dirsum2(const Eigen::MatrixBase&lt; Derived1 &gt; &amp;A, const Eigen::MatrixBase&lt; Derived2 &gt; &amp;B)</a>	88
6.4.2.19	<a href="#">get_dim_subsys(idx sz, idx N)</a>	88
6.4.2.20	<a href="#">get_num_subsys(idx sz, idx d)</a>	88
6.4.2.21	<a href="#">kron2(const Eigen::MatrixBase&lt; Derived1 &gt; &amp;A, const Eigen::MatrixBase&lt; Derived2 &gt; &amp;B)</a>	88
6.4.2.22	<a href="#">multiidx2n(const idx *const midx, idx numdims, const idx *const dims) noexcept</a>	88
6.4.2.23	<a href="#">n2multiidx(idx n, idx numdims, const idx *const dims, idx *result) noexcept</a>	88
6.4.2.24	<a href="#">variadic_vector_emplace(std::vector&lt; T &gt; &amp;)</a>	88
6.4.2.25	<a href="#">variadic_vector_emplace(std::vector&lt; T &gt; &amp;v, First &amp;&amp;first, Args &amp;&amp;...args)</a>	88
<b>7</b>	<b>Class Documentation</b>	<b>89</b>
7.1	<a href="#">qpp::Codes Class Reference</a>	89
7.1.1	<a href="#">Detailed Description</a>	90
7.1.2	<a href="#">Member Enumeration Documentation</a>	90
7.1.2.1	<a href="#">Type</a>	90
7.1.3	<a href="#">Constructor &amp; Destructor Documentation</a>	90
7.1.3.1	<a href="#">Codes()</a>	90
7.1.3.2	<a href="#">~Codes()=default</a>	90
7.1.4	<a href="#">Member Function Documentation</a>	91
7.1.4.1	<a href="#">codeword(Type type, idx i) const</a>	91
7.1.5	<a href="#">Friends And Related Function Documentation</a>	91
7.1.5.1	<a href="#">internal::Singleton&lt; const Codes &gt;</a>	91
7.2	<a href="#">qpp::exception::CustomException Class Reference</a>	91
7.2.1	<a href="#">Detailed Description</a>	92
7.2.2	<a href="#">Constructor &amp; Destructor Documentation</a>	92
7.2.2.1	<a href="#">CustomException(const std::string &amp;where, const std::string &amp;what)</a>	92
7.2.3	<a href="#">Member Function Documentation</a>	92
7.2.3.1	<a href="#">type_description() const override</a>	93
7.2.4	<a href="#">Member Data Documentation</a>	93
7.2.4.1	<a href="#">what_</a>	93
7.3	<a href="#">qpp::exception::DimsInvalid Class Reference</a>	93

7.3.1	Detailed Description	94
7.3.2	Member Function Documentation	94
7.3.2.1	type_description() const override	94
7.4	qpp::exception::DimsMismatchCvector Class Reference	95
7.4.1	Detailed Description	96
7.4.2	Member Function Documentation	96
7.4.2.1	type_description() const override	96
7.5	qpp::exception::DimsMismatchMatrix Class Reference	96
7.5.1	Detailed Description	97
7.5.2	Member Function Documentation	97
7.5.2.1	type_description() const override	97
7.6	qpp::exception::DimsMismatchRvector Class Reference	98
7.6.1	Detailed Description	99
7.6.2	Member Function Documentation	99
7.6.2.1	type_description() const override	99
7.7	qpp::exception::DimsMismatchVector Class Reference	99
7.7.1	Detailed Description	100
7.7.2	Member Function Documentation	100
7.7.2.1	type_description() const override	100
7.8	qpp::exception::DimsNotEqual Class Reference	101
7.8.1	Detailed Description	102
7.8.2	Member Function Documentation	102
7.8.2.1	type_description() const override	102
7.9	qpp::internal::Display_Impl_Struct Reference	102
7.9.1	Member Function Documentation	103
7.9.1.1	display_impl_(const T &A, std::ostream &os, double chop=qpp::chop) const	103
7.10	qpp::exception::Exception Class Reference	103
7.10.1	Detailed Description	105
7.10.2	Constructor & Destructor Documentation	106
7.10.2.1	Exception(const std::string &where)	106
7.10.3	Member Function Documentation	106
7.10.3.1	type_description() const =0	106
7.10.3.2	what() const noexcept override	106
7.10.4	Member Data Documentation	106
7.10.4.1	where_	106
7.11	qpp::Gates Class Reference	106
7.11.1	Detailed Description	109
7.11.2	Constructor & Destructor Documentation	109
7.11.2.1	Gates()	109
7.11.2.2	~Gates()=default	109

7.11.3	Member Function Documentation	109
7.11.3.1	CTRL(const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &ctrl, const std::vector< idx > &subsys, idx N, idx d=2) const	109
7.11.3.2	expandout(const Eigen::MatrixBase< Derived > &A, idx pos, const std::vector< idx > &dims) const	109
7.11.3.3	expandout(const Eigen::MatrixBase< Derived > &A, idx pos, const std::vector< idx > &initializer_list< idx > &dims) const	110
7.11.3.4	expandout(const Eigen::MatrixBase< Derived > &A, idx pos, idx N, idx d=2) const	110
7.11.3.5	Fd(idx D=2) const	111
7.11.3.6	Id(idx D=2) const	111
7.11.3.7	Rn(double theta, const std::vector< double > &n) const	111
7.11.3.8	Xd(idx D=2) const	112
7.11.3.9	Zd(idx D=2) const	112
7.11.4	Friends And Related Function Documentation	112
7.11.4.1	internal::Singleton< const Gates >	112
7.11.5	Member Data Documentation	112
7.11.5.1	CNOT	112
7.11.5.2	CNOTba	112
7.11.5.3	CZ	112
7.11.5.4	FRED	112
7.11.5.5	H	113
7.11.5.6	Id2	113
7.11.5.7	S	113
7.11.5.8	SWAP	113
7.11.5.9	T	113
7.11.5.10	TOF	113
7.11.5.11	X	113
7.11.5.12	Y	113
7.11.5.13	Z	113
7.12	qpp::IDisplay Class Reference	113
7.12.1	Detailed Description	115
7.12.2	Constructor & Destructor Documentation	115
7.12.2.1	IDisplay()=default	115
7.12.2.2	IDisplay(const IDisplay &)=default	115
7.12.2.3	IDisplay(IDisplay &&)=default	115
7.12.2.4	~IDisplay()=default	115
7.12.3	Member Function Documentation	115
7.12.3.1	display(std::ostream &os) const =0	115
7.12.3.2	operator=(const IDisplay &)=default	115
7.12.3.3	operator=(IDisplay &&)=default	115

7.12.4 Friends And Related Function Documentation . . . . .	115
7.12.4.1 operator<< . . . . .	115
7.13 qpp::Init Class Reference . . . . .	116
7.13.1 Detailed Description . . . . .	117
7.13.2 Constructor & Destructor Documentation . . . . .	117
7.13.2.1 Init() . . . . .	117
7.13.2.2 ~Init() . . . . .	117
7.13.3 Friends And Related Function Documentation . . . . .	117
7.13.3.1 internal::Singleton< const Init > . . . . .	117
7.14 qpp::internal::IOManipEigen Class Reference . . . . .	117
7.14.1 Constructor & Destructor Documentation . . . . .	118
7.14.1.1 IOManipEigen(const Eigen::MatrixBase< Derived > &A, double chop=qpp::chop) . . . . .	118
7.14.1.2 IOManipEigen(const cplx z, double chop=qpp::chop) . . . . .	118
7.14.2 Member Function Documentation . . . . .	118
7.14.2.1 display(std::ostream &os) const override . . . . .	118
7.14.3 Member Data Documentation . . . . .	119
7.14.3.1 A_ . . . . .	119
7.14.3.2 chop_ . . . . .	119
7.15 qpp::internal::IOManipPointer< PointerType > Class Template Reference . . . . .	119
7.15.1 Constructor & Destructor Documentation . . . . .	120
7.15.1.1 IOManipPointer(const PointerType *p, idx N, const std::string &separator, const std::string &start="","", const std::string &end=""]") . . . . .	120
7.15.1.2 IOManipPointer(const IOManipPointer &)=default . . . . .	120
7.15.2 Member Function Documentation . . . . .	120
7.15.2.1 display(std::ostream &os) const override . . . . .	120
7.15.2.2 operator=(const IOManipPointer &)=default . . . . .	120
7.15.3 Member Data Documentation . . . . .	120
7.15.3.1 end_ . . . . .	120
7.15.3.2 N_ . . . . .	120
7.15.3.3 p_ . . . . .	120
7.15.3.4 separator_ . . . . .	121
7.15.3.5 start_ . . . . .	121
7.16 qpp::internal::IOManipRange< InputIterator > Class Template Reference . . . . .	121
7.16.1 Constructor & Destructor Documentation . . . . .	122
7.16.1.1 IOManipRange(InputIterator first, InputIterator last, const std::string &separator, const std::string &start="","", const std::string &end=""]") . . . . .	122
7.16.1.2 IOManipRange(const IOManipRange &)=default . . . . .	122
7.16.2 Member Function Documentation . . . . .	122
7.16.2.1 display(std::ostream &os) const override . . . . .	122
7.16.2.2 operator=(const IOManipRange &)=default . . . . .	122

7.16.3	Member Data Documentation . . . . .	122
7.16.3.1	end_ . . . . .	122
7.16.3.2	first_ . . . . .	122
7.16.3.3	last_ . . . . .	122
7.16.3.4	separator_ . . . . .	123
7.16.3.5	start_ . . . . .	123
7.17	qpp::is_complex< T > Struct Template Reference . . . . .	123
7.17.1	Detailed Description . . . . .	123
7.18	qpp::is_complex< std::complex< T > > Struct Template Reference . . . . .	124
7.18.1	Detailed Description . . . . .	124
7.19	qpp::is_iterable< T, typename > Struct Template Reference . . . . .	125
7.19.1	Detailed Description . . . . .	125
7.20	qpp::is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), typename T::value_type > > Struct Template Reference . . . . .	126
7.20.1	Detailed Description . . . . .	127
7.21	qpp::is_matrix_expression< Derived > Struct Template Reference . . . . .	127
7.21.1	Detailed Description . . . . .	128
7.22	qpp::exception::MatrixMismatchSubsys Class Reference . . . . .	128
7.22.1	Detailed Description . . . . .	130
7.22.2	Member Function Documentation . . . . .	130
7.22.2.1	type_description() const override . . . . .	130
7.23	qpp::exception::MatrixNotCvector Class Reference . . . . .	130
7.23.1	Detailed Description . . . . .	131
7.23.2	Member Function Documentation . . . . .	131
7.23.2.1	type_description() const override . . . . .	131
7.24	qpp::exception::MatrixNotRvector Class Reference . . . . .	132
7.24.1	Detailed Description . . . . .	133
7.24.2	Member Function Documentation . . . . .	133
7.24.2.1	type_description() const override . . . . .	133
7.25	qpp::exception::MatrixNotSquare Class Reference . . . . .	133
7.25.1	Detailed Description . . . . .	134
7.25.2	Member Function Documentation . . . . .	134
7.25.2.1	type_description() const override . . . . .	134
7.26	qpp::exception::MatrixNotSquareNorCvector Class Reference . . . . .	135
7.26.1	Detailed Description . . . . .	136
7.26.2	Member Function Documentation . . . . .	136
7.26.2.1	type_description() const override . . . . .	136
7.27	qpp::exception::MatrixNotSquareNorRvector Class Reference . . . . .	136
7.27.1	Detailed Description . . . . .	137
7.27.2	Member Function Documentation . . . . .	137

7.27.2.1	<a href="#">type_description() const override</a>	137
7.28	<a href="#">qpp::exception::MatrixNotSquareNorVector Class Reference</a>	138
7.28.1	<a href="#">Detailed Description</a>	139
7.28.2	<a href="#">Member Function Documentation</a>	139
7.28.2.1	<a href="#">type_description() const override</a>	139
7.29	<a href="#">qpp::exception::MatrixNotVector Class Reference</a>	139
7.29.1	<a href="#">Detailed Description</a>	140
7.29.2	<a href="#">Member Function Documentation</a>	140
7.29.2.1	<a href="#">type_description() const override</a>	140
7.30	<a href="#">qpp::exception::NoCodeword Class Reference</a>	141
7.30.1	<a href="#">Detailed Description</a>	142
7.30.2	<a href="#">Member Function Documentation</a>	142
7.30.2.1	<a href="#">type_description() const override</a>	142
7.31	<a href="#">qpp::exception::NotBipartite Class Reference</a>	142
7.31.1	<a href="#">Detailed Description</a>	143
7.31.2	<a href="#">Member Function Documentation</a>	143
7.31.2.1	<a href="#">type_description() const override</a>	143
7.32	<a href="#">qpp::exception::NotQubitCvector Class Reference</a>	144
7.32.1	<a href="#">Detailed Description</a>	145
7.32.2	<a href="#">Member Function Documentation</a>	145
7.32.2.1	<a href="#">type_description() const override</a>	145
7.33	<a href="#">qpp::exception::NotQubitMatrix Class Reference</a>	145
7.33.1	<a href="#">Detailed Description</a>	146
7.33.2	<a href="#">Member Function Documentation</a>	146
7.33.2.1	<a href="#">type_description() const override</a>	146
7.34	<a href="#">qpp::exception::NotQubitRvector Class Reference</a>	147
7.34.1	<a href="#">Detailed Description</a>	148
7.34.2	<a href="#">Member Function Documentation</a>	148
7.34.2.1	<a href="#">type_description() const override</a>	148
7.35	<a href="#">qpp::exception::NotQubitSubsys Class Reference</a>	148
7.35.1	<a href="#">Detailed Description</a>	149
7.35.2	<a href="#">Member Function Documentation</a>	149
7.35.2.1	<a href="#">type_description() const override</a>	149
7.36	<a href="#">qpp::exception::NotQubitVector Class Reference</a>	150
7.36.1	<a href="#">Detailed Description</a>	151
7.36.2	<a href="#">Member Function Documentation</a>	151
7.36.2.1	<a href="#">type_description() const override</a>	151
7.37	<a href="#">qpp::exception::OutOfRange Class Reference</a>	151
7.37.1	<a href="#">Detailed Description</a>	152
7.37.2	<a href="#">Member Function Documentation</a>	152

7.37.2.1	<code>type_description()</code> const override	152
7.38	<code>qpp::exception::PermInvalid</code> Class Reference	153
7.38.1	Detailed Description	154
7.38.2	Member Function Documentation	154
7.38.2.1	<code>type_description()</code> const override	154
7.39	<code>qpp::exception::PermMismatchDims</code> Class Reference	154
7.39.1	Detailed Description	155
7.39.2	Member Function Documentation	155
7.39.2.1	<code>type_description()</code> const override	155
7.40	<code>qpp::RandomDevices</code> Class Reference	156
7.40.1	Detailed Description	157
7.40.2	Constructor & Destructor Documentation	157
7.40.2.1	<code>RandomDevices()</code>	157
7.40.2.2	<code>~RandomDevices()</code> =default	157
7.40.3	Friends And Related Function Documentation	157
7.40.3.1	<code>internal::Singleton&lt; RandomDevices &gt;</code>	157
7.40.4	Member Data Documentation	157
7.40.4.1	<code>rd_</code>	157
7.40.4.2	<code>rng_</code>	157
7.41	<code>qpp::internal::Singleton&lt; T &gt;</code> Class Template Reference	158
7.41.1	Detailed Description	158
7.41.2	Constructor & Destructor Documentation	159
7.41.2.1	<code>Singleton()</code> noexcept=default	159
7.41.2.2	<code>Singleton(const Singleton &amp;)=delete</code>	159
7.41.2.3	<code>~Singleton()</code> =default	159
7.41.3	Member Function Documentation	159
7.41.3.1	<code>get_instance()</code> noexcept( <code>std::is_nothrow_constructible&lt; T &gt;::value</code> )	159
7.41.3.2	<code>get_thread_local_instance()</code> noexcept( <code>std::is_nothrow_constructible&lt; T &gt;::value</code> )	159
7.41.3.3	<code>operator=(const Singleton &amp;)=delete</code>	159
7.42	<code>qpp::exception::SizeMismatch</code> Class Reference	159
7.42.1	Detailed Description	160
7.42.2	Member Function Documentation	160
7.42.2.1	<code>type_description()</code> const override	160
7.43	<code>qpp::States</code> Class Reference	161
7.43.1	Detailed Description	163
7.43.2	Constructor & Destructor Documentation	163
7.43.2.1	<code>States()</code>	163
7.43.2.2	<code>~States()</code> =default	163
7.43.3	Member Function Documentation	163
7.43.3.1	<code>jn(idx j, idx n, idx d=2) const</code>	163



7.43.3.2	<a href="#">mes(idx d=2) const</a>	163
7.43.3.3	<a href="#">minus(idx n) const</a>	164
7.43.3.4	<a href="#">one(idx n, idx d=2) const</a>	165
7.43.3.5	<a href="#">plus(idx n) const</a>	165
7.43.3.6	<a href="#">zero(idx n, idx d=2) const</a>	165
7.43.4	<a href="#">Friends And Related Function Documentation</a>	165
7.43.4.1	<a href="#">internal::Singleton&lt; const States &gt;</a>	165
7.43.5	<a href="#">Member Data Documentation</a>	165
7.43.5.1	<a href="#">b00</a>	165
7.43.5.2	<a href="#">b01</a>	166
7.43.5.3	<a href="#">b10</a>	166
7.43.5.4	<a href="#">b11</a>	166
7.43.5.5	<a href="#">GHZ</a>	166
7.43.5.6	<a href="#">pb00</a>	166
7.43.5.7	<a href="#">pb01</a>	166
7.43.5.8	<a href="#">pb10</a>	166
7.43.5.9	<a href="#">pb11</a>	166
7.43.5.10	<a href="#">pGHZ</a>	166
7.43.5.11	<a href="#">pW</a>	166
7.43.5.12	<a href="#">px0</a>	166
7.43.5.13	<a href="#">px1</a>	166
7.43.5.14	<a href="#">py0</a>	167
7.43.5.15	<a href="#">py1</a>	167
7.43.5.16	<a href="#">pz0</a>	167
7.43.5.17	<a href="#">pz1</a>	167
7.43.5.18	<a href="#">W</a>	167
7.43.5.19	<a href="#">x0</a>	167
7.43.5.20	<a href="#">x1</a>	167
7.43.5.21	<a href="#">y0</a>	167
7.43.5.22	<a href="#">y1</a>	167
7.43.5.23	<a href="#">z0</a>	167
7.43.5.24	<a href="#">z1</a>	167
7.44	<a href="#">qpp::exception::SubsysMismatchDims Class Reference</a>	168
7.44.1	<a href="#">Detailed Description</a>	169
7.44.2	<a href="#">Member Function Documentation</a>	169
7.44.2.1	<a href="#">type_description() const override</a>	169
7.45	<a href="#">qpp::Timer&lt; T, CLOCK_T &gt; Class Template Reference</a>	169
7.45.1	<a href="#">Detailed Description</a>	171
7.45.2	<a href="#">Constructor &amp; Destructor Documentation</a>	172
7.45.2.1	<a href="#">Timer() noexcept</a>	172

7.45.2.2	Timer(const Timer &)=default	172
7.45.2.3	Timer(Timer &&)=default	172
7.45.2.4	~Timer()=default	172
7.45.3	Member Function Documentation	172
7.45.3.1	display(std::ostream &os) const override	172
7.45.3.2	get_duration() const noexcept	172
7.45.3.3	operator=(const Timer &)=default	173
7.45.3.4	operator=(Timer &&)=default	173
7.45.3.5	tic() noexcept	173
7.45.3.6	tics() const noexcept	173
7.45.3.7	toc() noexcept	173
7.45.4	Member Data Documentation	173
7.45.4.1	end_	173
7.45.4.2	start_	173
7.46	qpp::exception::TypeMismatch Class Reference	174
7.46.1	Detailed Description	175
7.46.2	Member Function Documentation	175
7.46.2.1	type_description() const override	175
7.47	qpp::exception::UndefinedType Class Reference	175
7.47.1	Detailed Description	176
7.47.2	Member Function Documentation	176
7.47.2.1	type_description() const override	176
7.48	qpp::exception::Unknown Class Reference	177
7.48.1	Detailed Description	178
7.48.2	Member Function Documentation	178
7.48.2.1	type_description() const override	178
7.49	qpp::exception::ZeroSize Class Reference	178
7.49.1	Detailed Description	179
7.49.2	Member Function Documentation	179
7.49.2.1	type_description() const override	179
<b>8</b>	<b>File Documentation</b>	<b>181</b>
8.1	classes/codes.h File Reference	181
8.1.1	Detailed Description	181
8.2	classes/exception.h File Reference	181
8.2.1	Detailed Description	183
8.3	classes/gates.h File Reference	184
8.3.1	Detailed Description	184
8.4	classes/ideplay.h File Reference	184
8.4.1	Detailed Description	185

8.5	classes/init.h File Reference	185
8.5.1	Detailed Description	185
8.6	classes/random_devices.h File Reference	186
8.6.1	Detailed Description	186
8.7	classes/states.h File Reference	186
8.7.1	Detailed Description	187
8.8	classes/timer.h File Reference	187
8.8.1	Detailed Description	187
8.9	constants.h File Reference	188
8.9.1	Detailed Description	188
8.10	entanglement.h File Reference	189
8.10.1	Detailed Description	190
8.11	entropies.h File Reference	190
8.11.1	Detailed Description	191
8.12	experimental/experimental.h File Reference	191
8.12.1	Detailed Description	191
8.13	functions.h File Reference	192
8.13.1	Detailed Description	196
8.14	input_output.h File Reference	196
8.14.1	Detailed Description	197
8.15	instruments.h File Reference	197
8.15.1	Detailed Description	199
8.16	internal/classes/iomanip.h File Reference	199
8.16.1	Detailed Description	199
8.17	internal/classes/singleton.h File Reference	199
8.17.1	Detailed Description	200
8.18	internal/util.h File Reference	200
8.18.1	Detailed Description	202
8.19	macros.h File Reference	202
8.19.1	Detailed Description	202
8.19.2	Macro Definition Documentation	202
8.19.2.1	ERROR	202
8.19.2.2	ERRORLN	202
8.19.2.3	PRINT	203
8.19.2.4	PRINTLN	203
8.20	MATLAB/matlab.h File Reference	203
8.20.1	Detailed Description	203
8.21	number_theory.h File Reference	203
8.21.1	Detailed Description	205
8.22	operations.h File Reference	205

---

8.22.1 Detailed Description . . . . .	207
8.23 qpp.h File Reference . . . . .	207
8.23.1 Detailed Description . . . . .	209
8.23.2 Macro Definition Documentation . . . . .	209
8.23.2.1 QPP_UNUSED_ . . . . .	209
8.24 random.h File Reference . . . . .	209
8.24.1 Detailed Description . . . . .	210
8.25 statistics.h File Reference . . . . .	210
8.25.1 Detailed Description . . . . .	211
8.26 traits.h File Reference . . . . .	212
8.26.1 Detailed Description . . . . .	212
8.27 types.h File Reference . . . . .	213
8.27.1 Detailed Description . . . . .	214
<b>Index</b>	<b>215</b>

# Chapter 1

## Quantum++

### Version 1.0.0-devel - development

**Build status:** Master [! \[Build Status\] \(https://api.travis-ci.org/vsoftco/qpp.svg?branch=master\)](https://api.travis-ci.org/vsoftco/qpp.svg?branch=master) Devel [! \[Build Status\] \(https://api.travis-ci.org/vsoftco/qpp.svg?branch=v1.0.0-devel\)](https://api.travis-ci.org/vsoftco/qpp.svg?branch=v1.0.0-devel)

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, please contact me. 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 - 2017 Vlad Gheorghiu, vgheorgh AT gmail DOT com.

Quantum++ is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Quantum++ is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with Quantum++. If not, see <http://www.gnu.org/licenses/>.

### Building instructions

#### Configuration

- Compiler: [g++](#) version 4.8.2 or later (for good C++11 support)
- [Eigen 3](#) library located in \$HOME/eigen

- Quantum++ library located in `$HOME/qpp`

### Optional

- **MATLAB** compiler include header files: `/Applications/MATLAB_R2016a.app/extern/include`
- **MATLAB** compiler shared library files: `/Applications/MATLAB_R2016a.app/bin/maci64`

### Building using CMake (version 3.0.0 or later)

The current version of the repository has a `./CMakeLists.txt` configuration file for building examples using **CMake**. To build an example using **CMake**, I recommend an out-of-source build, i.e., from the root of the project (where `./include` is located), type

```
mkdir ./build
cd ./build
cmake ..
make
```

The commands above build the release version (default) executable `qpp`, from the source file `./examples/minimal.cpp`, without **MATLAB** support (default), inside the directory `./build`. To build a different configuration, e.g. debug version with **MATLAB** support, type from the root of the project

```
cd ./build
rm -rf *
cmake -DCMAKE_BUILD_TYPE=Debug -DWITH_MATLAB=ON ..
make
```

Or, to disable **OpenMP** support (enabled by default), type

```
cd ./build
rm -rf *
cmake -DWITH_OPENMP=OFF ..
make
```

To change the name of the example file, the location of the **Eigen 3** library or the location of **MATLAB** installation, edit the `./CMakeLists.txt` file. See also `./CMakeLists.txt` for additional options. Do not forget to clean the `./build` directory before a fresh build!

### Building without an automatic build system

- Example file: `$HOME/qpp/examples/minimal.cpp`
- Output executable: `$HOME/qpp/examples/minimal`
- You must run the commands below from inside the directory `$HOME/qpp/examples`

### Release version (without **MATLAB** support)

```
g++ -pedantic -std=c++11 -Wall -Wextra -Weffc++ -fopenmp \
    -O3 -DNDEBUG -DEIGEN_NO_DEBUG \
    -isystem $HOME/eigen -I $HOME/qpp/include \
    minimal.cpp -o minimal
```

### Debug version (without **MATLAB** support)

```
g++ -pedantic -std=c++11 -Wall -Wextra -Weffc++ -fopenmp \
    -g3 -DDEBUG \
    -isystem $HOME/eigen -I $HOME/qpp/include \
    minimal.cpp -o minimal
```

### Release version (with MATLAB support)

```
g++ -pedantic -std=c++11 -Wall -Wextra -Weffc++ -fopenmp \
-O3 -DNDEBUG -DEIGEN_NO_DEBUG \
-isystem $HOME/eigen -I $HOME/qpp/include \
-I/Applications/MATLAB_R2016a.app/extern/include \
-L/Applications/MATLAB_R2016a.app/bin/maci64 \
-lmx -lmat minimal.cpp -o minimal
```

### Debug version (with MATLAB support)

```
g++ -pedantic -std=c++11 -Wall -Wextra -Weffc++ -fopenmp \
-g3 -DDEBUG \
-isystem $HOME/eigen -I $HOME/qpp/include \
-I /Applications/MATLAB_R2016a.app/extern/include \
-L /Applications/MATLAB_R2016a.app/bin/maci64 \
-lmx -lmat minimal.cpp -o minimal
```

### Unit testing

Quantum++ was extensively tested via a suite of unit tests constructed with [Google Test 1.8.0](#) (included with the project in `./unit_tests/lib/gtest-1.8.0`). The source code of the unit tests is provided under `./unit_tests/tests`. To build and run the unit tests, I strongly recommend to use [CMake](#) version 3.0.0 or later. Assuming you do use [CMake](#), switch to the `./unit_tests` directory, create a build directory inside it, then from the newly created `./unit_tests/build` type

```
cmake ..
make
```

The commands above build `./unit_tests/build/tests/qpp_testing`, which you then may run. Note that `qpp::Timer` tests or tests related to random functions such as `qpp::rand()` may sometime (very rarely) fail, due to timing imprecision or statistical errors. Such a behaviour is perfectly normal.

### Note

The [CMake](#) configuration file `./unit_tests/CMakeLists.txt` defines the same building options and default choices as the main `./CMakeLists.txt` of Quantum++. Therefore you can use the same flags as the ones mentioned at the beginning of this document when customizing the build. You should modify `./unit_tests/CMakeLists.txt` accordingly in case your [Eigen 3](#) library or [MATLAB](#) include/library files are in a different location than the one assumed in this document.

### Additional remarks

- The C++ compiler must be fully standard-C++11 compliant.
- If using [Windows](#), I recommend compiling under [cygwin](#) via [CMake](#) and `g++`. See also <http://stackoverflow.com/questions/28997206/cygwin-support-for-c11-in-g4-9-2> for a bug related to lack of support for some C++11 math functions, and how to fix it. Quick fix: patch the standard library header file `<cmath>` using the provided patch `./cmath_cygwin.patch`.
- In case you use [OS X/macOS](#) and want to install [clang++](#) version 3.7 or later, I highly recommend to install it via [macports](#).
- If you use [clang++](#) version 3.7 or later and want to use [OpenMP](#) (enabled by default), make sure to modify `CLANG_LIBOMP` and `CLANG_LIBOMP_INCLUDE` in `CMakeLists.txt` so they point to the correct location of the [OpenMP](#) library, as otherwise [clang++](#) will not find `<omp.h>` and the `libomp` shared library.
- If you run the program on [OS X/macOS](#) with [MATLAB](#) support, make sure that the environment variable `DYLD_LIBRARY_PATH` is set to point to the [MATLAB](#) compiler library location, see the `run_mac_MATLAB` script. Otherwise, you get a runtime error similar to

```
> dyld: Library not loaded: @rpath/libmat.dylib.
```

- I recommend running via a script, as otherwise setting the `DYLD_LIBRARY_PATH` globally may interfere with `macports`' `CMake` installation (in case you use `CMake` from `macports`). If you use a script, then the environment variable is local to the script and does not interfere with the rest of the system.
- Example of script, assumed to be located in the root directory of Quantum++

```
#!/bin/sh

MATLAB=/Applications/MATLAB_R2016a.app
export DYLD_LIBRARY_PATH=$DYLD_LIBRARY_PATH:$MATLAB/bin/maci64

./build/qpp
```

- If you build a debug version with `g++` under `OS X/macOS` and use `gdb` to step inside template functions you may want to add `-fno-weak` compiler flag. See <http://stackoverflow.com/questions/23330641/gnu-gdb-can-not-step-into-template-functions-os-x-mavericks> for more details about this problem.



## 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 . . . . .	15
<a href="#">qpp::exception</a>	Quantum++ exception hierarchy namespace . . . . .	84
<a href="#">qpp::experimental</a>	Experimental/test functions/classes, do not use or modify . . . . .	86
<a href="#">qpp::internal</a>	Internal utility functions, do not use them directly or modify them . . . . .	86



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

qpp::internal::Display_Impl_ . . . . .	102
qpp::internal::IOManipEigen . . . . .	117
std::exception	
qpp::exception::Exception . . . . .	103
qpp::exception::CustomException . . . . .	91
qpp::exception::DimsInvalid . . . . .	93
qpp::exception::DimsMismatchCvector . . . . .	95
qpp::exception::DimsMismatchMatrix . . . . .	96
qpp::exception::DimsMismatchRvector . . . . .	98
qpp::exception::DimsMismatchVector . . . . .	99
qpp::exception::DimsNotEqual . . . . .	101
qpp::exception::MatrixMismatchSubsys . . . . .	128
qpp::exception::MatrixNotCvector . . . . .	130
qpp::exception::MatrixNotRvector . . . . .	132
qpp::exception::MatrixNotSquare . . . . .	133
qpp::exception::MatrixNotSquareNorCvector . . . . .	135
qpp::exception::MatrixNotSquareNorRvector . . . . .	136
qpp::exception::MatrixNotSquareNorVector . . . . .	138
qpp::exception::MatrixNotVector . . . . .	139
qpp::exception::NoCodeword . . . . .	141
qpp::exception::NotBipartite . . . . .	142
qpp::exception::NotQubitCvector . . . . .	144
qpp::exception::NotQubitMatrix . . . . .	145
qpp::exception::NotQubitRvector . . . . .	147
qpp::exception::NotQubitSubsys . . . . .	148
qpp::exception::NotQubitVector . . . . .	150
qpp::exception::OutOfRange . . . . .	151
qpp::exception::PermInvalid . . . . .	153
qpp::exception::PermMismatchDims . . . . .	154
qpp::exception::SizeMismatch . . . . .	159
qpp::exception::SubsysMismatchDims . . . . .	168
qpp::exception::TypeMismatch . . . . .	174
qpp::exception::UndefinedType . . . . .	175
qpp::exception::Unknown . . . . .	177
qpp::exception::ZeroSize . . . . .	178
false_type	
qpp::is_complex< T > . . . . .	123

qpp::is_iterable< T, typename > . . . . .	125
qpp::IDisplay . . . . .	113
qpp::internal::IOManipEigen . . . . .	117
qpp::internal::IOManipPointer< PointerType > . . . . .	119
qpp::internal::IOManipRange< InputIterator > . . . . .	121
qpp::Timer< T, CLOCK_T > . . . . .	169
is_base_of	
qpp::is_matrix_expression< Derived > . . . . .	127
qpp::internal::Singleton< T > . . . . .	158
qpp::internal::Singleton< const Codes > . . . . .	158
qpp::Codes . . . . .	89
qpp::internal::Singleton< const Gates > . . . . .	158
qpp::Gates . . . . .	106
qpp::internal::Singleton< const Init > . . . . .	158
qpp::Init . . . . .	116
qpp::internal::Singleton< const States > . . . . .	158
qpp::States . . . . .	161
qpp::internal::Singleton< RandomDevices > . . . . .	158
qpp::RandomDevices . . . . .	156
true_type	
qpp::is_complex< std::complex< T > > . . . . .	124
qpp::is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), typename T::value_type > > . . . . .	126

## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">qpp::Codes</a>	Const Singleton class that defines quantum error correcting codes . . . . .	89
<a href="#">qpp::exception::CustomException</a>	Custom exception . . . . .	91
<a href="#">qpp::exception::DimsInvalid</a>	Invalid dimension(s) exception . . . . .	93
<a href="#">qpp::exception::DimsMismatchCvector</a>	Dimension(s) mismatch column vector size exception . . . . .	95
<a href="#">qpp::exception::DimsMismatchMatrix</a>	Dimension(s) mismatch matrix size exception . . . . .	96
<a href="#">qpp::exception::DimsMismatchRvector</a>	Dimension(s) mismatch row vector size exception . . . . .	98
<a href="#">qpp::exception::DimsMismatchVector</a>	Dimension(s) mismatch vector size exception . . . . .	99
<a href="#">qpp::exception::DimsNotEqual</a>	Dimensions not equal exception . . . . .	101
<a href="#">qpp::internal::Display_Impl_</a>	. . . . .	102
<a href="#">qpp::exception::Exception</a>	Base class for generating Quantum++ custom exceptions . . . . .	103
<a href="#">qpp::Gates</a>	Const Singleton class that implements most commonly used gates . . . . .	106
<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> . . . . .	113
<a href="#">qpp::Init</a>	Const Singleton class that performs additional initializations/cleanups . . . . .	116
<a href="#">qpp::internal::IOManipEigen</a>	. . . . .	117
<a href="#">qpp::internal::IOManipPointer&lt; PointerType &gt;</a>	. . . . .	119
<a href="#">qpp::internal::IOManipRange&lt; InputIterator &gt;</a>	. . . . .	121
<a href="#">qpp::is_complex&lt; T &gt;</a>	Checks whether the type is a complex type . . . . .	123
<a href="#">qpp::is_complex&lt; std::complex&lt; T &gt; &gt;</a>	Checks whether the type is a complex number type, specialization for complex types . . . . .	124
<a href="#">qpp::is_iterable&lt; T, typename &gt;</a>	Checks whether <i>T</i> is compatible with an STL-like iterable container . . . . .	125

<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>	
Checks whether <i>T</i> is compatible with an STL-like iterable container, specialization for STL-like iterable containers	126
<a href="#">qpp::is_matrix_expression&lt; Derived &gt;</a>	
Checks whether the type is an Eigen matrix expression	127
<a href="#">qpp::exception::MatrixMismatchSubsys</a>	
Matrix mismatch subsystems exception	128
<a href="#">qpp::exception::MatrixNotCvector</a>	
Matrix is not a column vector exception	130
<a href="#">qpp::exception::MatrixNotRvector</a>	
Matrix is not a row vector exception	132
<a href="#">qpp::exception::MatrixNotSquare</a>	
Matrix is not square exception	133
<a href="#">qpp::exception::MatrixNotSquareNorCvector</a>	
Matrix is not square nor column vector exception	135
<a href="#">qpp::exception::MatrixNotSquareNorRvector</a>	
Matrix is not square nor row vector exception	136
<a href="#">qpp::exception::MatrixNotSquareNorVector</a>	
Matrix is not square nor vector exception	138
<a href="#">qpp::exception::MatrixNotVector</a>	
Matrix is not a vector exception	139
<a href="#">qpp::exception::NoCodeword</a>	
Codeword does not exist exception	141
<a href="#">qpp::exception::NotBipartite</a>	
Not bi-partite exception	142
<a href="#">qpp::exception::NotQubitCvector</a>	
Column vector is not 2 x 1 exception	144
<a href="#">qpp::exception::NotQubitMatrix</a>	
Matrix is not 2 x 2 exception	145
<a href="#">qpp::exception::NotQubitRvector</a>	
Row vector is not 1 x 2 exception	147
<a href="#">qpp::exception::NotQubitSubsys</a>	
Subsystems are not qubits exception	148
<a href="#">qpp::exception::NotQubitVector</a>	
Vector is not 2 x 1 nor 1 x 2 exception	150
<a href="#">qpp::exception::OutOfRange</a>	
Parameter out of range exception	151
<a href="#">qpp::exception::PermInvalid</a>	
Invalid permutation exception	153
<a href="#">qpp::exception::PermMismatchDims</a>	
Permutation mismatch dimensions exception	154
<a href="#">qpp::RandomDevices</a>	
Singleton class that manages the source of randomness in the library	156
<a href="#">qpp::internal::Singleton&lt; T &gt;</a>	
Singleton policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern)	158
<a href="#">qpp::exception::SizeMismatch</a>	
Size mismatch exception	159
<a href="#">qpp::States</a>	
Const Singleton class that implements most commonly used states	161
<a href="#">qpp::exception::SubsysMismatchDims</a>	
Subsystems mismatch dimensions exception	168
<a href="#">qpp::Timer&lt; T, CLOCK_T &gt;</a>	
Chronometer	169
<a href="#">qpp::exception::TypeMismatch</a>	
Type mismatch exception	174

<a href="#">qpp::exception::UndefinedType</a>	
Not defined for this type exception . . . . .	175
<a href="#">qpp::exception::Unknown</a>	
Unknown exception . . . . .	177
<a href="#">qpp::exception::ZeroSize</a>	
Object has zero size exception . . . . .	178





## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">constants.h</a>	
Constants . . . . .	188
<a href="#">entanglement.h</a>	
Entanglement functions . . . . .	189
<a href="#">entropies.h</a>	
Entropy functions . . . . .	190
<a href="#">functions.h</a>	
Generic quantum computing functions . . . . .	192
<a href="#">input_output.h</a>	
Input/output functions . . . . .	196
<a href="#">instruments.h</a>	
Measurement functions . . . . .	197
<a href="#">macros.h</a>	
Preprocessor macros . . . . .	202
<a href="#">number_theory.h</a>	
Number theory functions . . . . .	203
<a href="#">operations.h</a>	
Quantum operation functions . . . . .	205
<a href="#">qpp.h</a>	
Quantum++ main header file, includes all other necessary headers . . . . .	207
<a href="#">random.h</a>	
Randomness-related functions . . . . .	209
<a href="#">statistics.h</a>	
Statistics functions . . . . .	210
<a href="#">traits.h</a>	
Type traits . . . . .	212
<a href="#">types.h</a>	
Type aliases . . . . .	213
classes/ <a href="#">codes.h</a>	
Quantum error correcting codes . . . . .	181
classes/ <a href="#">exception.h</a>	
Exceptions . . . . .	181
classes/ <a href="#">gates.h</a>	
Quantum gates . . . . .	184
classes/ <a href="#">display.h</a>	
Display interface via the non-virtual interface (NVI) . . . . .	184
classes/ <a href="#">init.h</a>	
Initialization . . . . .	185

classes/ <a href="#">random_devices.h</a>	
Random devices . . . . .	186
classes/ <a href="#">states.h</a>	
Quantum states . . . . .	186
classes/ <a href="#">timer.h</a>	
Timing . . . . .	187
experimental/ <a href="#">experimental.h</a>	
Experimental/test functions/classes . . . . .	191
internal/ <a href="#">util.h</a>	
Internal utility functions . . . . .	200
internal/classes/ <a href="#">iomanip.h</a>	
Input/output manipulators . . . . .	199
internal/classes/ <a href="#">singleton.h</a>	
Singleton pattern via CRTP . . . . .	199
MATLAB/ <a href="#">matlab.h</a>	
Input/output interfacing with MATLAB . . . . .	203

## 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.*

#### Classes

- class [Codes](#)  
*const Singleton class that defines quantum error correcting codes*
- 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.*
- 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... >  
using [to\\_void](#) = void  
*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](#) (unsigned long long int x) noexcept  
*User-defined literal for complex  $i = \sqrt{-1}$  (integer overload)*
- 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 hevects` (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 > x2contfrac` (double x, idx N, idx cut=1e5)  
*Simple continued fraction expansion.*
- `double contfrac2x` (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>::infinity()`  
*Used to denote infinity in double precision.*

### 6.1.1 Detailed Description

Quantum++ main namespace.

### 6.1.2 Typedef Documentation

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

Big integer.

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

Complex (double precision) dynamic Eigen row vector.

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

Complex (double precision) dynamic Eigen matrix.

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

Complex number in double precision.

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

Real (double precision) dynamic Eigen matrix.

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

Non-negative integer index.

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

Complex (double precision) dynamic Eigen column vector.

#### 6.1.2.11 `template<typename... > using qpp::to_void = typedef void`

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 `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 `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 `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 `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 `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 `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 `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
$A$	Eigen expression
<i>subsys</i>	Subsystem indexes where the gate $A$ 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 `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** `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** `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** `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 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 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 `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 `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**

<i>r</i>	3-dimensional real vector
----------	---------------------------

**Returns**

Qubit density matrix

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

Converts Choi matrix to superoperator matrix.

## See also

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

## Parameters

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

## Returns

Superoperator matrix

#### 6.1.3.18 `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

$A$	Eigen expression
$B$	Eigen expression

## Returns

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

#### 6.1.3.19 `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
$N$	Total number of systems

## Returns

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

6.1.3.20 `std::vector<idx> qpp::compperm ( 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 `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 `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 `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 `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 $X$ and $Y$ in lexicographical order ( $X$ labels the rows, $Y$ labels the columns)
$X$	Real random variable values represented by an STL-like container
$Y$	Real random variable values represented by an STL-like container

## Returns

Correlation of  $X$  and  $Y$

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

Matrix cos.

## Parameters

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

## Returns

Matrix cosine of  $A$

6.1.3.26 `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 $X$ and $Y$ in lexicographical order ( $X$ labels the rows, $Y$ labels the columns)
$X$	Real random variable values represented by an STL-like container
$Y$	Real random variable values represented by an STL-like container

## Returns

Covariance of  $X$  and  $Y$

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

Functor.

## Parameters

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

## Returns

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

6.1.3.28 `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 `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 `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 `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 `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 `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 `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::LOManipEigen](#)

**6.1.3.35** `internal::LOManipEigen 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::LOManipEigen](#)

**6.1.3.36** `template<typename InputIterator > internal::LOManipRange<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::LOManipRange](#)

**6.1.3.37** `template<typename Container > internal::LOManipRange<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::LOManipRange](#)

6.1.3.38 `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::LOManipPointer](#)

**6.1.3.39** `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** `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** `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

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

## Returns

Entanglement, with the logarithm in base 2

**6.1.3.42** `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

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

## Returns

Entanglement, with the logarithm in base 2

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

Prime factor decomposition.

## Note

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

## Parameters

$a$	Integer different from 0, 1 or -1
-----	-----------------------------------

## Returns

Integer vector containing the factors

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

Functional calculus  $f(A)$



## Parameters

$A$	Eigen expression
$f$	Pointer-to-function from complex to complex

## Returns

$$f(A)$$

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

Greatest common divisor of two integers.

See also

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

## Parameters

$a$	Integer
$b$	Integer

## Returns

Greatest common divisor of  $a$  and  $b$

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

Greatest common divisor of a list of integers.

See also

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

## Parameters

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

## Returns

Greatest common divisor of all numbers in  $as$

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

G-concurrence 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

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

## Returns

G-concurrence

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

Gram-Schmidt orthogonalization.

## Parameters

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

Gram-Schmidt orthogonalization.

## Parameters

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

Gram-Schmidt orthogonalization.

## Parameters

$A$	Eigen expression, the input vectors are the columns of $A$
-----	--

## 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** `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

$A$	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 `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 `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 `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 `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 `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 `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 `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 `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

$Ks$	Set of Kraus operators
------	------------------------

Returns

Choi matrix

6.1.3.65 `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  $Ks$  in the standard operator basis  $\{|i\rangle\langle j|\}$  ordered in lexicographical order, i.e.  $|0\rangle\langle 0|$ ,  $|0\rangle\langle 1|$  etc.

Parameters

$Ks$	Set of Kraus operators
------	------------------------

Returns

Superoperator matrix

6.1.3.66 `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

$head$	Eigen expression
--------	------------------

Returns

Its argument  $head$

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

Kronecker product.

See also

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

Parameters

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

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 `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 `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 `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 `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** `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** `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 `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 `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 `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 `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 `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 `std::vector<double> qpp::marginalY ( 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  $Y$

6.1.3.82 `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 `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

<i>A</i>	Eigen expression
<i>Ks</i>	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 `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

<i>A</i>	Eigen expression
<i>U</i>	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 `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 `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 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 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** `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** `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

$A$	Eigen expression
$V$	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
$d$	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 `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

$A$	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 `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

$A$	Eigen expression
$subsys$	Subsystem indexes that are measured
$d$	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** `ket qpp::mket ( const std::vector< idx > & mask, const std::vector< idx > & dims ) [inline]`

Multi-partite qudit 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

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

## Returns

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

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

Multi-partite qudit 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

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

## Returns

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

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

Projector onto multi-partite qudit ket.

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

Projector onto multi-partite qudit ket.

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** `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** `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

$n$	Non-negative integer index
$dims$	Dimensions of the multi-partite system

## Returns

Multi-index of the same size as  $dims$

6.1.3.102 `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

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

## Returns

Negativity

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

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

## Parameters

$A$	Eigen expression
$d$	Subsystem dimensions

## Returns

Negativity

6.1.3.104 `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 `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** `constexpr cplx qpp::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>
```

**6.1.3.107** `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>
```

**6.1.3.108** `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  $A$

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

Projector.

Normalized projector onto state vector

## Parameters

<i>A</i>	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*

6.1.3.110 `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*

6.1.3.111 `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

6.1.3.112 `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

6.1.3.113 `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

$A$	Eigen expression
$subsys$	Subsystem indexes
$dims$	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  $A$

6.1.3.114 `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  $A$

6.1.3.115 `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  $A$

6.1.3.116 `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  $A$

6.1.3.117 `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  $A$

6.1.3.118 `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  $A$

6.1.3.119 `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.

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  $A$

6.1.3.120 `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.

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  $A$

6.1.3.121 `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

<i>subsysB</i>	Indexes of the second subsystem
<i>dims</i>	Dimensions of the multi-partite system

**Returns**

Mutual information between the 2 subsystems

**6.1.3.122** `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

**6.1.3.123** `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)

**6.1.3.124** `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]



**6.1.3.125** `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`

**6.1.3.126** `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

**6.1.3.127** `template<> cmatrix 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

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

Generates a random Hermitian matrix.

## Parameters

$D$	Dimension of the Hilbert space
-----	--------------------------------

## Returns

Random Hermitian matrix

**6.1.3.129** `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

$a$	Beginning of the interval, belongs to it
$b$	End of the interval, belongs to it

## Returns

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

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

Generates a random normalized ket (pure state vector)

## Parameters

$D$	Dimension of the Hilbert space
-----	--------------------------------

## Returns

Random normalized ket

**6.1.3.131** `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

$N$	Number of Kraus operators
$D$	Dimension of the Hilbert space

## Returns

Set of  $N$  Kraus operators satisfying the closure condition

**6.1.3.132** `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  $N(\text{mean}, \text{sigma})$

If complex, then both real and imaginary parts are normally distributed in  $N(\text{mean}, \text{sigma})$

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

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

Generates a random real matrix with entries normally distributed in  $N(\text{mean}, \text{sigma})$ , 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

**6.1.3.134** `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

**6.1.3.135** `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  $N(\text{mean}, \text{sigma})$

**6.1.3.136** `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**

$N$	Size of the permutation
-----	-------------------------

**Returns**

Random permutation of size  $N$

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

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

**Parameters**

$a$	Beginning of the interval, belongs to it
$b$	End of the interval, belongs to it
$N$	Maximum number of candidates

**Returns**

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

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

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

**Parameters**

$N$	Size of the probability vector
-----	--------------------------------

**Returns**

Random probability vector

**6.1.3.139** `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

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

Generates a random unitary matrix.

**Parameters**

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

**Returns**

Random unitary

**6.1.3.141** `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

**6.1.3.142** `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

**6.1.3.143** `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 <a href="#">qpp::infy</a> for $\alpha = \infty$

## Returns

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

6.1.3.144 `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*

6.1.3.145 `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

6.1.3.146 `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*

6.1.3.147 `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

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

6.1.3.148 `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

<i>A</i>	Eigen expression over the complex field
<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 `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

6.1.3.150 `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  $A$

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

Schmidt basis on Alice side.

## Parameters

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

## Returns

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

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

Schmidt basis on Alice side.

## Parameters

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

## Returns

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

6.1.3.153 `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.

6.1.3.154 `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.

6.1.3.155 `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

6.1.3.156 `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

$A$	Eigen expression
$d$	Subsystem dimensions

## Returns

Schmidt coefficients of  $A$ , ordered in decreasing order, as a real dynamic column vector

6.1.3.157 `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

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

## Returns

Real vector consisting of the Schmidt probabilities of  $A$ , ordered in decreasing order

6.1.3.158 `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

$A$	Eigen expression
$d$	Subsystem dimensions

## Returns

Real vector consisting of the Schmidt probabilities of  $A$ , ordered in decreasing order

6.1.3.159 `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  $X$

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

Matrix sin.

## Parameters

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

## Returns

Matrix sine of  $A$

6.1.3.161 `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  $A^z$

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

Matrix square root.

## Parameters

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

## Returns

Matrix square root of  $A$

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

Element-wise sum of  $A$ .

## Parameters

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

## Returns

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

**6.1.3.164** `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

**6.1.3.165** `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

**6.1.3.166** `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

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

Singular values.

## Parameters

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

## Returns

Singular values of  $A$ , ordered in decreasing order, as a real dynamic column vector

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

Full singular value decomposition.

## Parameters

$A$	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

6.1.3.169 `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  $A$

6.1.3.170 `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  $A$

6.1.3.171 `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
$perm$	Permutation
$dims$	Dimensions of the multi-partite system

## Returns

Permuted system, as a dynamic matrix over the same scalar field as  $A$

6.1.3.172 `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  $A$

6.1.3.173 `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  $A$

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

Transpose.

## Parameters

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

## Returns

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

6.1.3.175 `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

$A$	Eigen expression
$q$	Non-negative real number

## Returns

Tsallis-  $q$  entropy

6.1.3.176 `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

$prob$	Real probability vector
$q$	Non-negative real number

## Returns

Tsallis-  $q$  entropy

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

Uniform probability distribution vector.

## Parameters

$N$	Size of the alphabet
-----	----------------------

## Returns

Real vector consisting of a uniform distribution of size  $N$

6.1.3.178 `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

$prob$	Real probability vector representing the probability distribution of $X$
$X$	Real random variable values represented by an STL-like container

## Returns

Variance of  $X$

6.1.3.179 `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 `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 `constexpr double qpp::ee = 2.718281828459045235360287471352662497`

Base of natural logarithm,  $e$ .

6.1.4.3 `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 `constexpr double qpp::infy = std::numeric_limits<double>::infinity()`

Used to denote infinity in double precision.

6.1.4.5 `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 `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 x 1 exception.*
- class [NotQubitMatrix](#)  
*Matrix is not 2 x 2 exception.*
- class [NotQubitRvector](#)  
*Row vector is not 1 x 2 exception.*
- class [NotQubitSubsys](#)  
*Subsystems are not qubits exception.*
- class [NotQubitVector](#)  
*Vector is not 2 x 1 nor 1 x 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 `template<typename Derived > bool qpp::internal::check_cvector ( const Eigen::MatrixBase< Derived > & A )`

6.4.2.2 `bool qpp::internal::check_dims ( const std::vector< idx > & dims ) [inline]`

6.4.2.3 `template<typename Derived > bool qpp::internal::check_dims_match_cvect ( const std::vector< idx > & dims, const Eigen::MatrixBase< Derived > & A )`

6.4.2.4 `template<typename Derived > bool qpp::internal::check_dims_match_mat ( const std::vector< idx > & dims, const Eigen::MatrixBase< Derived > & A )`

6.4.2.5 `template<typename Derived > bool qpp::internal::check_dims_match_rvect ( const std::vector< idx > & dims, const Eigen::MatrixBase< Derived > & A )`

- 6.4.2.6 `bool qpp::internal::check_eq_dims ( const std::vector< idx > & dims, idx dim ) [inline], [noexcept]`
- 6.4.2.7 `template<typename T1 , typename T2 > bool qpp::internal::check_matching_sizes ( const T1 & lhs, const T2 & rhs ) [noexcept]`
- 6.4.2.8 `template<typename T > bool qpp::internal::check_nonzero_size ( const T & x ) [noexcept]`
- 6.4.2.9 `bool qpp::internal::check_perm ( const std::vector< idx > & perm ) [inline]`
- 6.4.2.10 `template<typename Derived > bool qpp::internal::check_qubit_cvector ( const Eigen::MatrixBase< Derived > & A ) [noexcept]`
- 6.4.2.11 `template<typename Derived > bool qpp::internal::check_qubit_matrix ( const Eigen::MatrixBase< Derived > & A ) [noexcept]`
- 6.4.2.12 `template<typename Derived > bool qpp::internal::check_qubit_rvector ( const Eigen::MatrixBase< Derived > & A ) [noexcept]`
- 6.4.2.13 `template<typename Derived > bool qpp::internal::check_qubit_vector ( const Eigen::MatrixBase< Derived > & A ) [noexcept]`
- 6.4.2.14 `template<typename Derived > bool qpp::internal::check_rvector ( const Eigen::MatrixBase< Derived > & A )`
- 6.4.2.15 `template<typename Derived > bool qpp::internal::check_square_mat ( const Eigen::MatrixBase< Derived > & A )`
- 6.4.2.16 `bool qpp::internal::check_subsys_match_dims ( const std::vector< idx > & subsys, const std::vector< idx > & dims ) [inline]`
- 6.4.2.17 `template<typename Derived > bool qpp::internal::check_vector ( const Eigen::MatrixBase< Derived > & A )`
- 6.4.2.18 `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 `idx qpp::internal::get_dim_subsys ( idx sz, idx N ) [inline]`
- 6.4.2.20 `idx qpp::internal::get_num_subsys ( idx sz, idx d ) [inline]`
- 6.4.2.21 `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 `idx qpp::internal::multiidx2n ( const idx *const midx, idx numdims, const idx *const dims ) [inline], [noexcept]`
- 6.4.2.23 `void qpp::internal::n2multiidx ( idx n, idx numdims, const idx *const dims, idx * result ) [inline], [noexcept]`
- 6.4.2.24 `template<typename T > void qpp::internal::variadic_vector_emplace ( std::vector< T > & )`
- 6.4.2.25 `template<typename T , typename First , typename... Args> void qpp::internal::variadic_vector_emplace ( std::vector< T > & v, First && first, Args &&... args )`

## Chapter 7

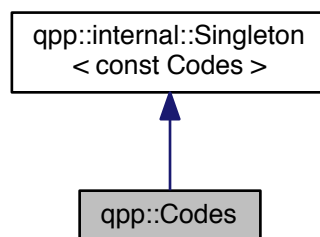
# Class Documentation

### 7.1 qpp::Codes Class Reference

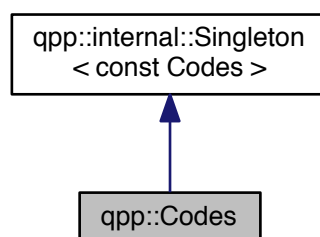
const Singleton class that defines quantum error correcting codes

```
#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

### 7.1.1 Detailed Description

const Singleton class that defines quantum error correcting codes

### 7.1.2 Member Enumeration Documentation

#### 7.1.2.1 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

### 7.1.3 Constructor & Destructor Documentation

#### 7.1.3.1 `qpp::Codes::Codes ( )` [inline],[private]

Default constructor.

#### 7.1.3.2 `qpp::Codes::~~Codes ( )` [private],[default]

Default destructor.

### 7.1.4 Member Function Documentation

#### 7.1.4.1 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*

### 7.1.5 Friends And Related Function Documentation

#### 7.1.5.1 friend class internal::Singleton< const Codes > [friend]

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

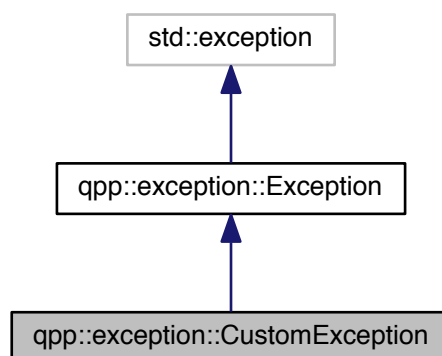
- [classes/codes.h](#)

## 7.2 qpp::exception::CustomException Class Reference

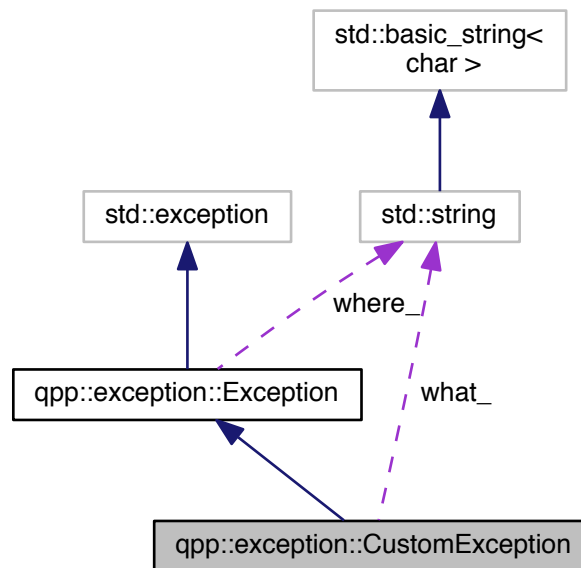
Custom exception.

```
#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\\_](#) {}

### 7.2.1 Detailed Description

Custom exception.

Custom exception, the user must provide a custom message

### 7.2.2 Constructor & Destructor Documentation

- 7.2.2.1 `qpp::exception::CustomException::CustomException ( const std::string & where, const std::string & what )`  
[inline]

### 7.2.3 Member Function Documentation



7.2.3.1 `std::string qpp::exception::CustomException::type_description ( ) const` `[inline]`, `[override]`, `[private]`, `[virtual]`

[Exception](#) type description.

#### Returns

[Exception](#) type description

Implements [qpp::exception::Exception](#).

## 7.2.4 Member Data Documentation

7.2.4.1 `std::string qpp::exception::CustomException::what_ {}` `[private]`

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

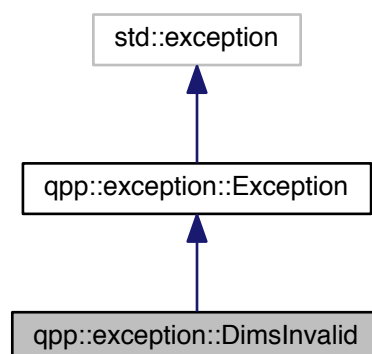
- [classes/exception.h](#)

## 7.3 qpp::exception::DimsInvalid Class Reference

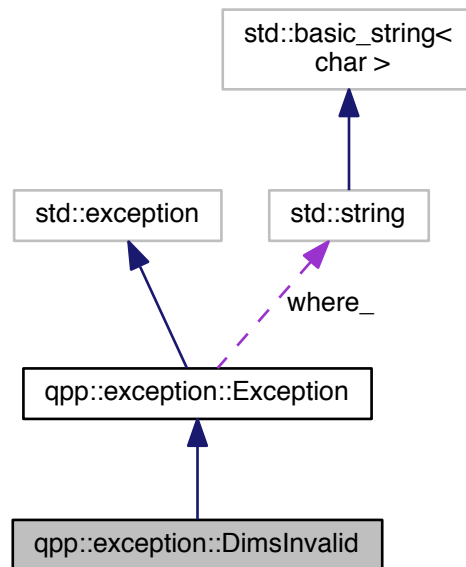
Invalid dimension(s) exception.

```
#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.*

### 7.3.1 Detailed Description

Invalid dimension(s) exception.

`std::vector<idx>` of dimensions has zero size or contains zeros

### 7.3.2 Member Function Documentation

**7.3.2.1** `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:

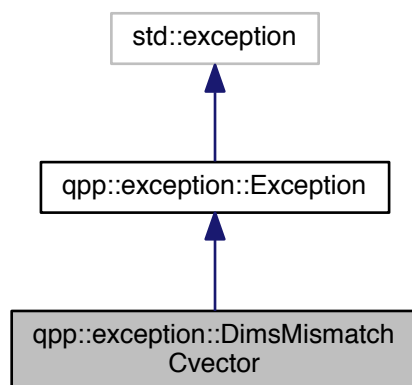
- `classes/exception.h`

## 7.4 qpp::exception::DimsMismatchCvector Class Reference

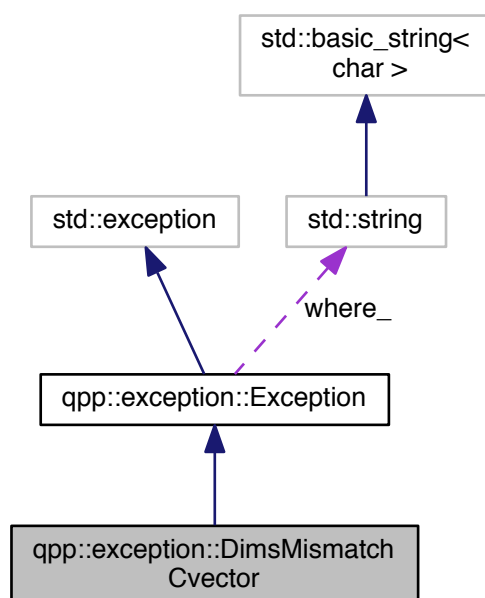
Dimension(s) mismatch column vector size exception.

```
#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.*

### 7.4.1 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)

### 7.4.2 Member Function Documentation

**7.4.2.1** `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:

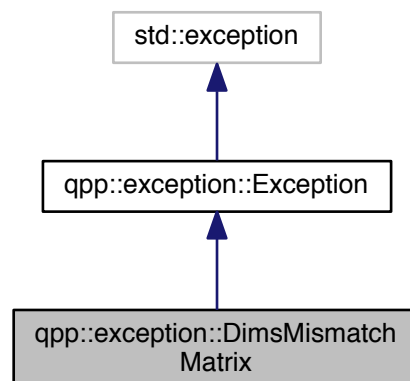
- `classes/exception.h`

## 7.5 qpp::exception::DimsMismatchMatrix Class Reference

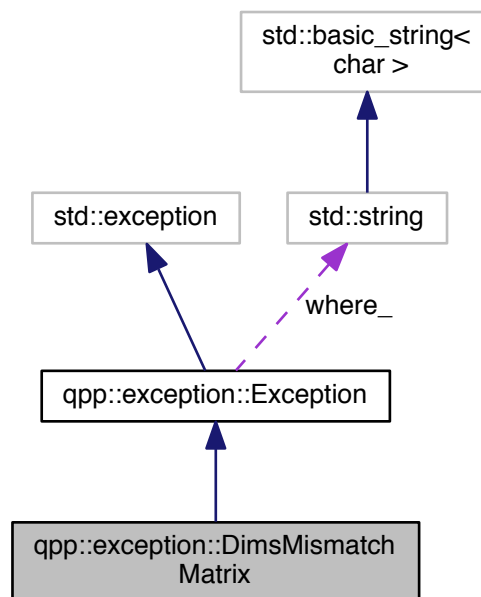
Dimension(s) mismatch matrix size exception.

```
#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.*

### 7.5.1 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)

### 7.5.2 Member Function Documentation

**7.5.2.1** `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:

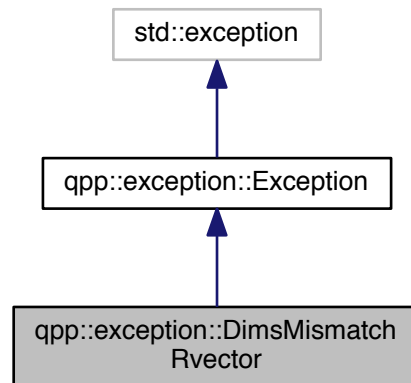
- `classes/exception.h`

## 7.6 qpp::exception::DimsMismatchRvector Class Reference

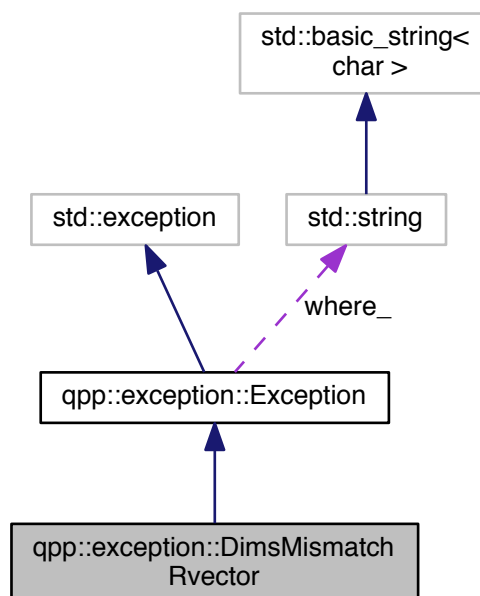
Dimension(s) mismatch row vector size exception.

```
#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.*

### 7.6.1 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)

### 7.6.2 Member Function Documentation

**7.6.2.1** `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:

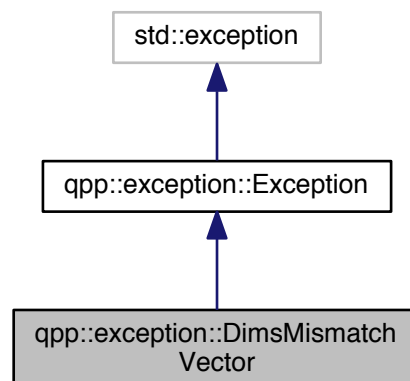
- `classes/exception.h`

## 7.7 qpp::exception::DimsMismatchVector Class Reference

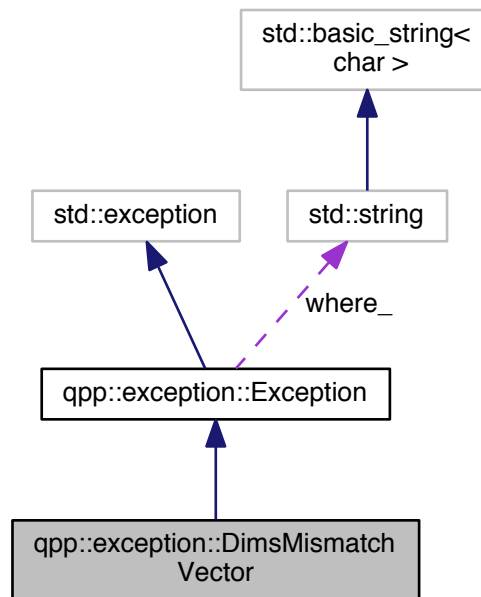
Dimension(s) mismatch vector size exception.

```
#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.*

### 7.7.1 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)

### 7.7.2 Member Function Documentation

**7.7.2.1** `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`

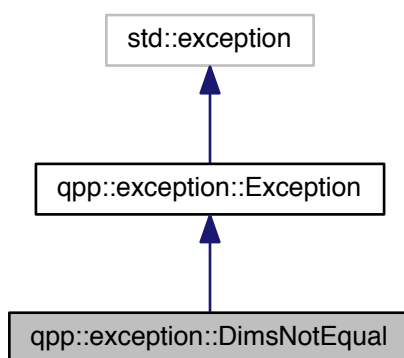


## 7.8 qpp::exception::DimsNotEqual Class Reference

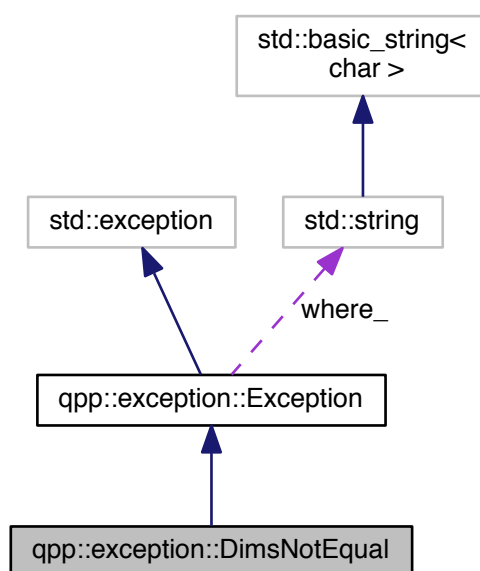
Dimensions not equal exception.

```
#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.*

### 7.8.1 Detailed Description

Dimensions not equal exception.

Local/global dimensions are not equal

### 7.8.2 Member Function Documentation

**7.8.2.1** `std::string qpp::exception::DimsNotEqual::type_description ( ) const` `[inline]`, `[override]`, `[virtual]`

*Exception type description.*

#### Returns

*Exception type description*

Implements `qpp::exception::Exception`.

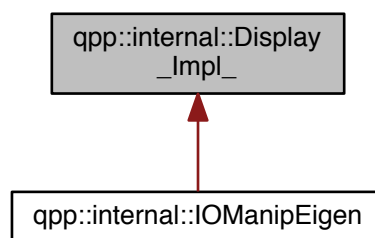
The documentation for this class was generated from the following file:

- `classes/exception.h`

## 7.9 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`

### 7.9.1 Member Function Documentation

7.9.1.1 `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:

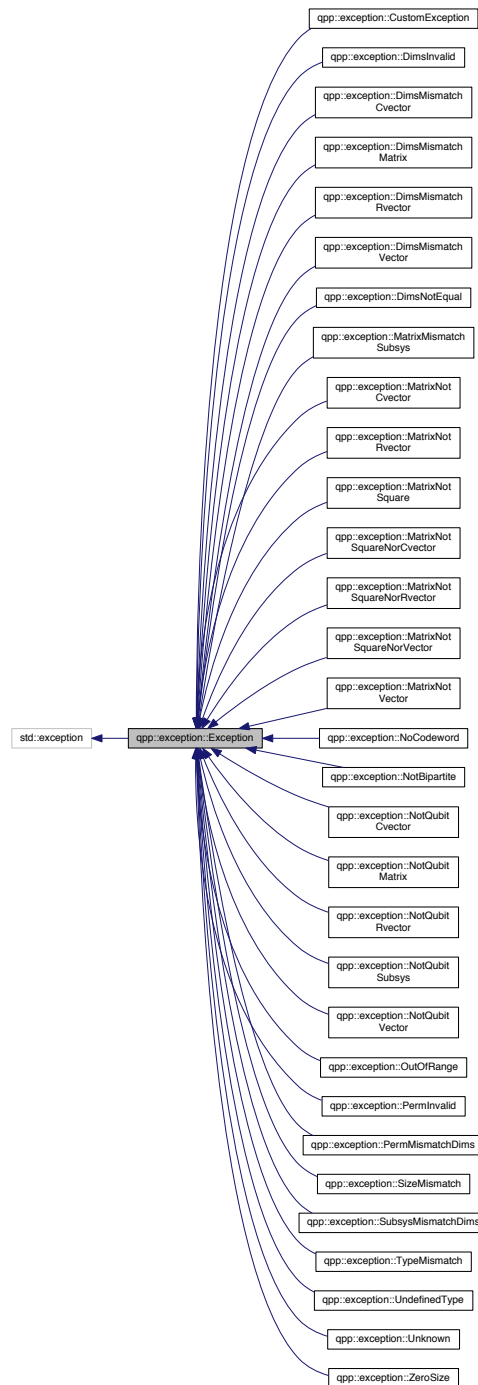
- [internal/util.h](#)

## 7.10 qpp::exception::Exception Class Reference

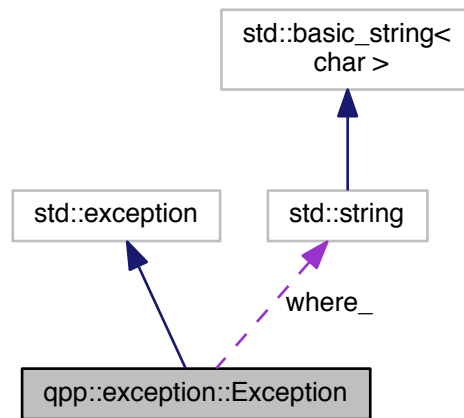
Base class for generating Quantum++ custom exceptions.

```
#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\\_](#)

### 7.10.1 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 qpp::exception::Exception constructor
        using Exception::Exception;
    }
}
  
```

```
};
} // namespace exception
} // namespace qpp
```

## 7.10.2 Constructor & Destructor Documentation

### 7.10.2.1 `qpp::exception::Exception::Exception ( const std::string & where ) [inline]`

Constructs an exception.

Parameters

<i>where</i>	Text representing where the exception occurred
--------------	--

## 7.10.3 Member Function Documentation

### 7.10.3.1 `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](#).

### 7.10.3.2 `virtual const char* qpp::exception::Exception::what ( ) const [inline],[override],[virtual],[noexcept]`

Overrides `std::exception::what()`

Returns

[Exception](#) description

## 7.10.4 Member Data Documentation

### 7.10.4.1 `std::string qpp::exception::Exception::where_ [private]`

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

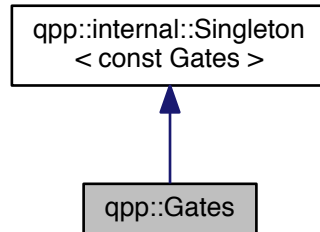
- [classes/exception.h](#)

## 7.11 `qpp::Gates` Class Reference

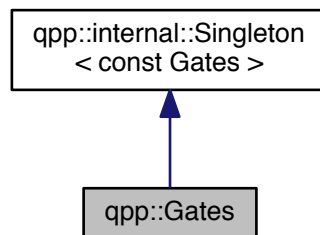
const Singleton class that implements most commonly used gates

```
#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

### 7.11.1 Detailed Description

const Singleton class that implements most commonly used gates

### 7.11.2 Constructor & Destructor Documentation

#### 7.11.2.1 qpp::Gates::Gates ( ) [inline], [private]

Initializes the gates.

#### 7.11.2.2 qpp::Gates::~~Gates ( ) [private], [default]

Default destructor.

### 7.11.3 Member Function Documentation

#### 7.11.3.1 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

$A$	Eigen expression
<i>ctrl</i>	Control subsystem indexes
<i>subsys</i>	Subsystem indexes where the gate $A$ is applied
$N$	Total number of subsystems
$d$	Subsystem dimensions

Returns

CTRL- $A$  gate, as a matrix over the same scalar field as  $A$

#### 7.11.3.2 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](#)( $I, I, \dots, I, A, I, \dots, I$ ).

## 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  $A$

7.11.3.3 `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  $A$

7.11.3.4 `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
-----	------------------

<i>pos</i>	Position
<i>N</i>	Number of subsystems
<i>d</i>	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  $A$

### 7.11.3.5 `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**

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

**Returns**

Fourier transform gate for qudits

### 7.11.3.6 `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**

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

**Returns**

Identity gate on a Hilbert space of dimension  $D$

### 7.11.3.7 `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><math>\theta</math></i>	Rotation angle
<i><math>n</math></i>	3-dimensional real (unit) vector

**Returns**

Rotation gate

#### 7.11.3.8 `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

$D$	Dimension of the Hilbert space
-----	--------------------------------

##### Returns

Generalized X gate for qudits

#### 7.11.3.9 `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

$D$	Dimension of the Hilbert space
-----	--------------------------------

##### Returns

Generalized Z gate for qudits

### 7.11.4 Friends And Related Function Documentation

#### 7.11.4.1 `friend class internal::Singleton< const Gates > [friend]`

### 7.11.5 Member Data Documentation

#### 7.11.5.1 `cmat qpp::Gates::CNOT {cmat::Identity(4, 4)}`

Controlled-NOT control target gate.

#### 7.11.5.2 `cmat qpp::Gates::CNOTba {cmat::Zero(4, 4)}`

Controlled-NOT target control gate.

#### 7.11.5.3 `cmat qpp::Gates::CZ {cmat::Identity(4, 4)}`

Controlled-Phase gate.

#### 7.11.5.4 `cmat qpp::Gates::FRED {cmat::Identity(8, 8)}`

Fredkin gate.

7.11.5.5 `cmat qpp::Gates::H {cmat::Zero(2, 2)}`

Hadamard gate.

7.11.5.6 `cmat qpp::Gates::Id2 {cmat::Identity(2, 2)}`

Identity gate.

7.11.5.7 `cmat qpp::Gates::S {cmat::Zero(2, 2)}`

S gate.

7.11.5.8 `cmat qpp::Gates::SWAP {cmat::Identity(4, 4)}`

SWAP gate.

7.11.5.9 `cmat qpp::Gates::T {cmat::Zero(2, 2)}`

T gate.

7.11.5.10 `cmat qpp::Gates::TOF {cmat::Identity(8, 8)}`

Toffoli gate.

7.11.5.11 `cmat qpp::Gates::X {cmat::Zero(2, 2)}`

Pauli Sigma-X gate.

7.11.5.12 `cmat qpp::Gates::Y {cmat::Zero(2, 2)}`

Pauli Sigma-Y gate.

7.11.5.13 `cmat qpp::Gates::Z {cmat::Zero(2, 2)}`

Pauli Sigma-Z gate.

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

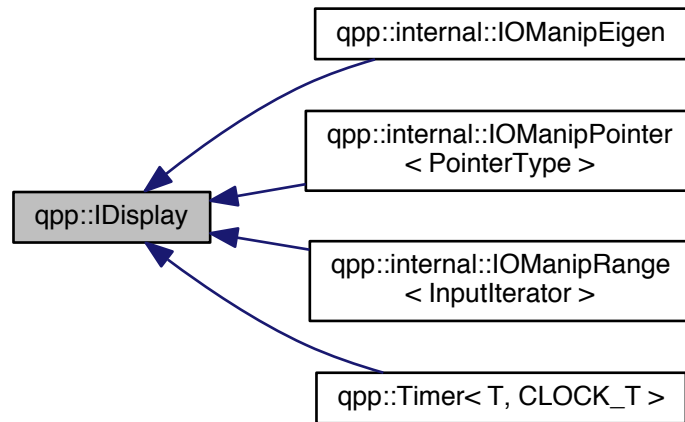
- [classes/gates.h](#)

## 7.12 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.*

### 7.12.1 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.

### 7.12.2 Constructor & Destructor Documentation

#### 7.12.2.1 `qpp::IDisplay::IDisplay ( )` [default]

Default constructor.

#### 7.12.2.2 `qpp::IDisplay::IDisplay ( const IDisplay & )` [default]

Default copy constructor.

#### 7.12.2.3 `qpp::IDisplay::IDisplay ( IDisplay && )` [default]

Default move constructor.

#### 7.12.2.4 `virtual qpp::IDisplay::~IDisplay ( )` [virtual], [default]

Default virtual destructor.

### 7.12.3 Member Function Documentation

#### 7.12.3.1 `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::internal::IOManipEigen`, `qpp::Timer< T, CLOCK_T >`, `qpp::internal::IOManipPointer< PointerType >`, and `qpp::internal::IOManipRange< InputIterator >`.

#### 7.12.3.2 `IDisplay& qpp::IDisplay::operator= ( const IDisplay & )` [default]

Default copy assignment operator.

#### 7.12.3.3 `IDisplay& qpp::IDisplay::operator= ( IDisplay && )` [default]

Default move assignment operator.

### 7.12.4 Friends And Related Function Documentation

#### 7.12.4.1 `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:

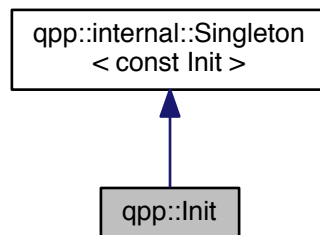
- [classes/ideplay.h](#)

## 7.13 qpp::Init Class Reference

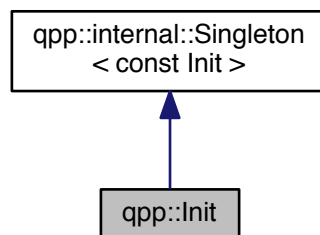
const Singleton class that performs additional initializations/cleanups

```
#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

### 7.13.1 Detailed Description

const Singleton class that performs additional initializations/cleanups

### 7.13.2 Constructor & Destructor Documentation

#### 7.13.2.1 `qpp::Init::Init ( )` `[inline]`, `[private]`

Additional initializations.

#### 7.13.2.2 `qpp::Init::~~Init ( )` `[inline]`, `[private]`

Cleanups.

### 7.13.3 Friends And Related Function Documentation

#### 7.13.3.1 `friend class internal::Singleton< const Init >` `[friend]`

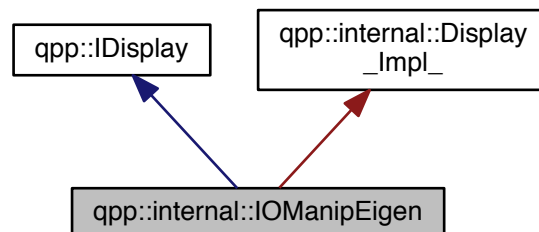
The documentation for this class was generated from the following file:

- [classes/init.h](#)

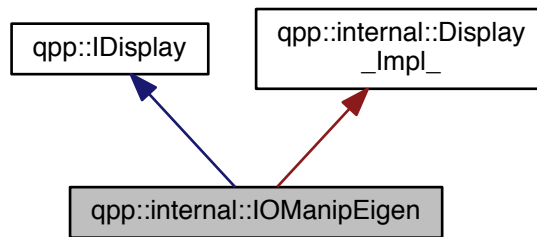
## 7.14 qpp::internal::IOManipEigen Class Reference

```
#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_`

## 7.14.1 Constructor & Destructor Documentation

7.14.1.1 `template<typename Derived > qpp::internal::IOManipEigen::IOManipEigen ( const Eigen::MatrixBase< Derived > &A, double chop = qpp::chop ) [inline],[explicit]`

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

## 7.14.2 Member Function Documentation

7.14.2.1 `std::ostream& qpp::internal::IOManipEigen::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`.

### 7.14.3 Member Data Documentation

7.14.3.1 `cmat qpp::internal::IOManipEigen::A_` [private]

7.14.3.2 `double qpp::internal::IOManipEigen::chop_` [private]

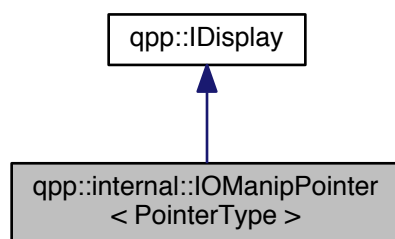
The documentation for this class was generated from the following file:

- [internal/classes/iomanip.h](#)

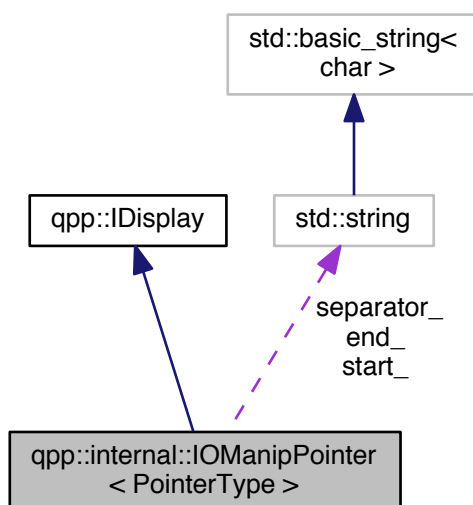
## 7.15 qpp::internal::IOManipPointer< PointerType > Class Template Reference

```
#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\\_](#)

## 7.15.1 Constructor & Destructor Documentation

- 7.15.1.1 `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]`
- 7.15.1.2 `template<typename PointerType> qpp::internal::IOManipPointer< PointerType >::IOManipPointer ( const IOManipPointer< PointerType > & ) [default]`

## 7.15.2 Member Function Documentation

- 7.15.2.1 `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](#).

- 7.15.2.2 `template<typename PointerType> IOManipPointer& qpp::internal::IOManipPointer< PointerType >::operator= ( const IOManipPointer< PointerType > & ) [default]`

## 7.15.3 Member Data Documentation

- 7.15.3.1 `template<typename PointerType> std::string qpp::internal::IOManipPointer< PointerType >::end_ [private]`
- 7.15.3.2 `template<typename PointerType> idx qpp::internal::IOManipPointer< PointerType >::N_ [private]`
- 7.15.3.3 `template<typename PointerType> const PointerType* qpp::internal::IOManipPointer< PointerType >::p_ [private]`

7.15.3.4 `template<typename PointerType> std::string qpp::internal::IOManipPointer< PointerType >::separator_  
[private]`

7.15.3.5 `template<typename PointerType> std::string qpp::internal::IOManipPointer< PointerType >::start_  
[private]`

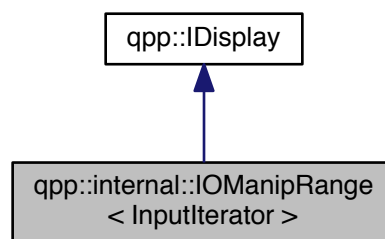
The documentation for this class was generated from the following file:

- [internal/classes/iomanip.h](#)

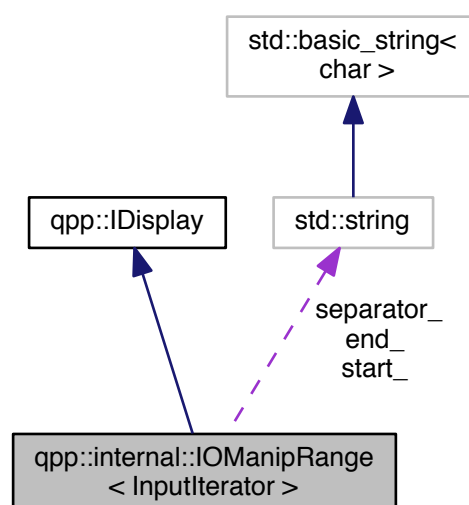
## 7.16 qpp::internal::IOManipRange< InputIterator > Class Template Reference

```
#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\\_](#)

### 7.16.1 Constructor & Destructor Documentation

- 7.16.1.1 `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]`
- 7.16.1.2 `template<typename InputIterator> qpp::internal::IOManipRange< InputIterator >::IOManipRange ( const IOManipRange< InputIterator > & ) [default]`

### 7.16.2 Member Function Documentation

- 7.16.2.1 `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](#).

- 7.16.2.2 `template<typename InputIterator> IOManipRange& qpp::internal::IOManipRange< InputIterator >::operator= ( const IOManipRange< InputIterator > & ) [default]`

### 7.16.3 Member Data Documentation

- 7.16.3.1 `template<typename InputIterator> std::string qpp::internal::IOManipRange< InputIterator >::end_ [private]`
- 7.16.3.2 `template<typename InputIterator> InputIterator qpp::internal::IOManipRange< InputIterator >::first_ [private]`
- 7.16.3.3 `template<typename InputIterator> InputIterator qpp::internal::IOManipRange< InputIterator >::last_ [private]`

7.16.3.4 `template<typename InputIterator> std::string qpp::internal::IOManipRange< InputIterator >::separator_`  
`[private]`

7.16.3.5 `template<typename InputIterator> std::string qpp::internal::IOManipRange< InputIterator >::start_`  
`[private]`

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

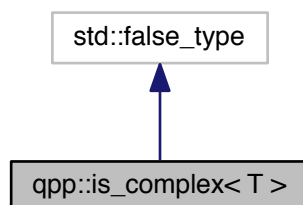
- [internal/classes/iomanip.h](#)

## 7.17 qpp::is\_complex< T > Struct Template Reference

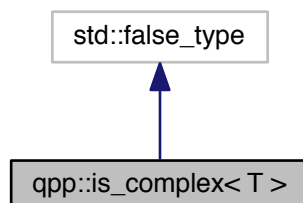
Checks whether the type is a complex type.

```
#include <traits.h>
```

Inheritance diagram for `qpp::is_complex< T >`:



Collaboration diagram for `qpp::is_complex< T >`:



### 7.17.1 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:

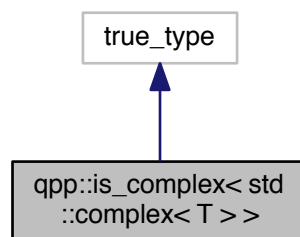
- [traits.h](#)

## 7.18 qpp::is\_complex< std::complex< T > > Struct Template Reference

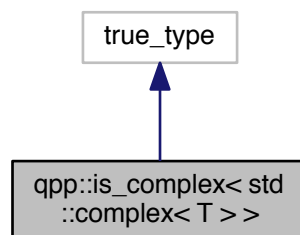
Checks whether the type is a complex number type, specialization for complex types.

```
#include <traits.h>
```

Inheritance diagram for qpp::is\_complex< std::complex< T > >:



Collaboration diagram for qpp::is\_complex< std::complex< T > >:



### 7.18.1 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](#)

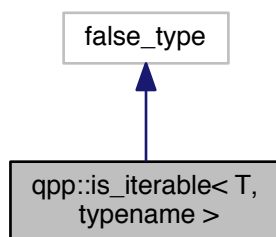


## 7.19 qpp::is\_iterable< T, typename > Struct Template Reference

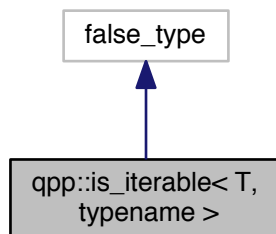
Checks whether *T* is compatible with an STL-like iterable container.

```
#include <traits.h>
```

Inheritance diagram for qpp::is\_iterable< T, typename >:



Collaboration diagram for qpp::is\_iterable< T, typename >:



### 7.19.1 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:

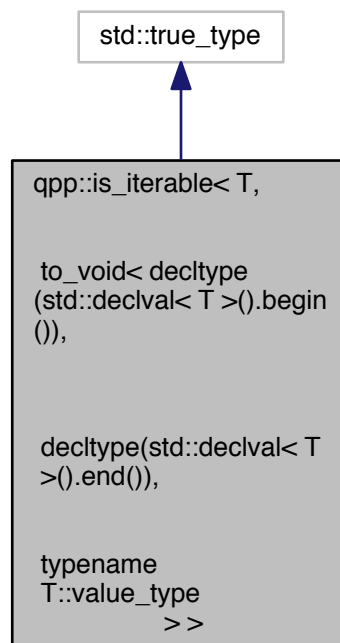
- [traits.h](#)

## 7.20 `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.

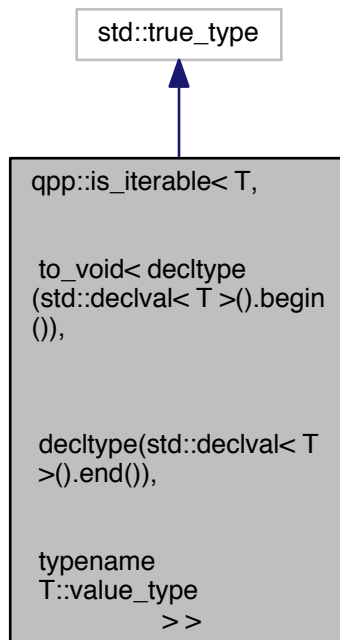
```
#include <traits.h>
```

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 > >`:

```
:declval< T >().end()), typename T::value_type > >:
```



### 7.20.1 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:

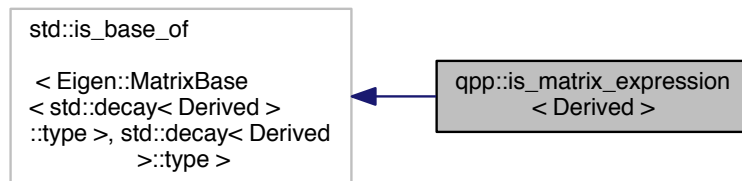
- [traits.h](#)

## 7.21 qpp::is\_matrix\_expression< Derived > Struct Template Reference

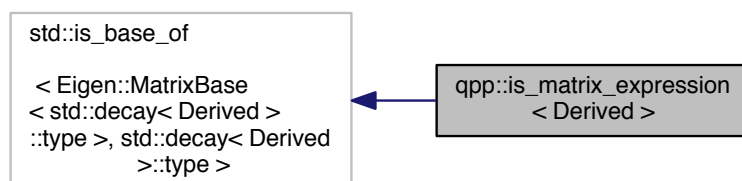
Checks whether the type is an Eigen matrix expression.

```
#include <traits.h>
```

Inheritance diagram for `qpp::is_matrix_expression< Derived >`:



Collaboration diagram for `qpp::is_matrix_expression< Derived >`:



### 7.21.1 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 *Eigen::MatrixBase<Derived>*. Otherwise, *value* is equal to *false*.

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

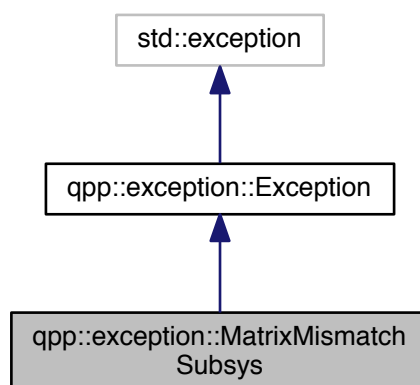
- [traits.h](#)

## 7.22 qpp::exception::MatrixMismatchSubsys Class Reference

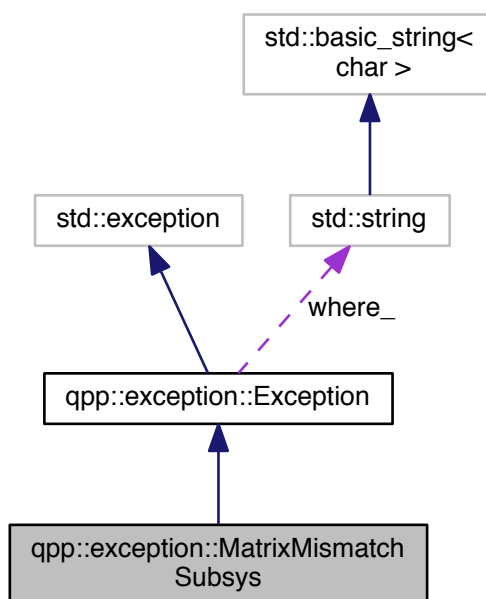
Matrix mismatch subsystems exception.

```
#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.*

### 7.22.1 Detailed Description

Matrix mismatch subsystems exception.

Matrix size mismatch subsystem sizes (e.g. in [qpp::apply\(\)](#))

### 7.22.2 Member Function Documentation

7.22.2.1 `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:

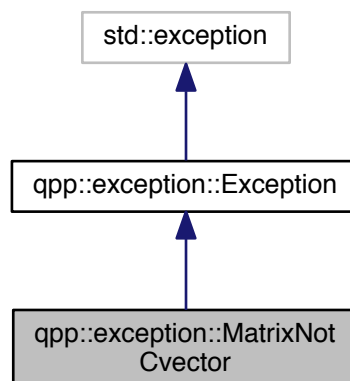
- [classes/exception.h](#)

## 7.23 qpp::exception::MatrixNotCvector Class Reference

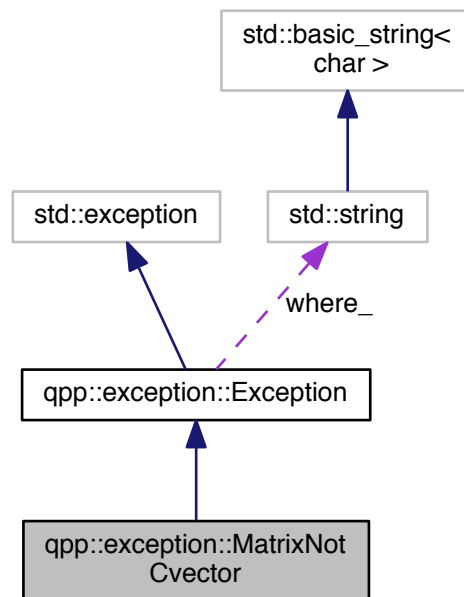
Matrix is not a column vector exception.

```
#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.*

### 7.23.1 Detailed Description

Matrix is not a column vector exception.

Eigen::Matrix is not a column vector

### 7.23.2 Member Function Documentation

**7.23.2.1** `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:

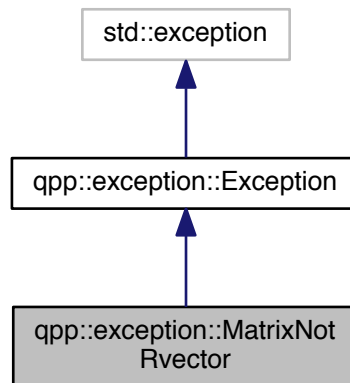
- `classes/exception.h`

## 7.24 qpp::exception::MatrixNotRvector Class Reference

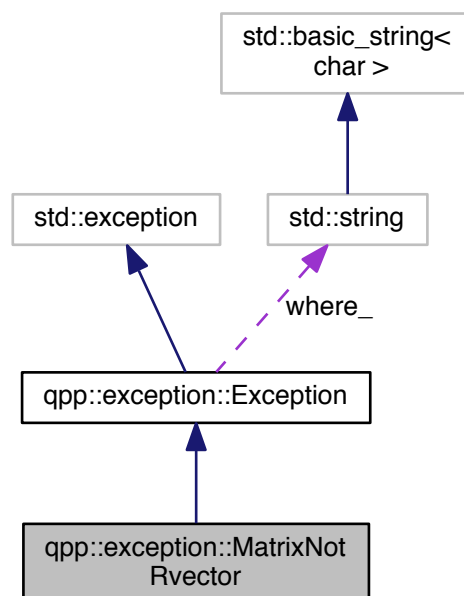
Matrix is not a row vector exception.

```
#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.*

### 7.24.1 Detailed Description

Matrix is not a row vector exception.

Eigen::Matrix is not a row vector

### 7.24.2 Member Function Documentation

**7.24.2.1** `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:

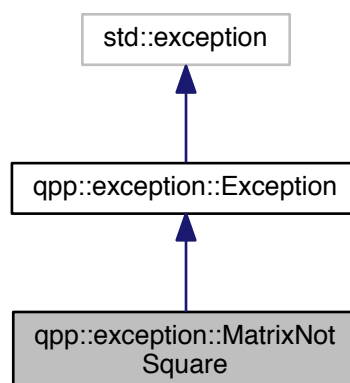
- `classes/exception.h`

## 7.25 qpp::exception::MatrixNotSquare Class Reference

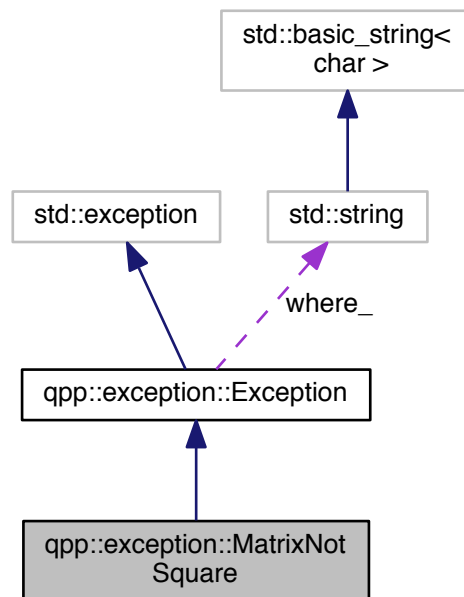
Matrix is not square exception.

```
#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.*

### 7.25.1 Detailed Description

Matrix is not square exception.

Eigen::Matrix is not a square matrix

### 7.25.2 Member Function Documentation

**7.25.2.1** `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:

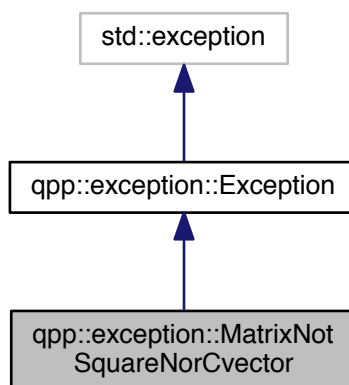
- `classes/exception.h`

## 7.26 qpp::exception::MatrixNotSquareNorCvector Class Reference

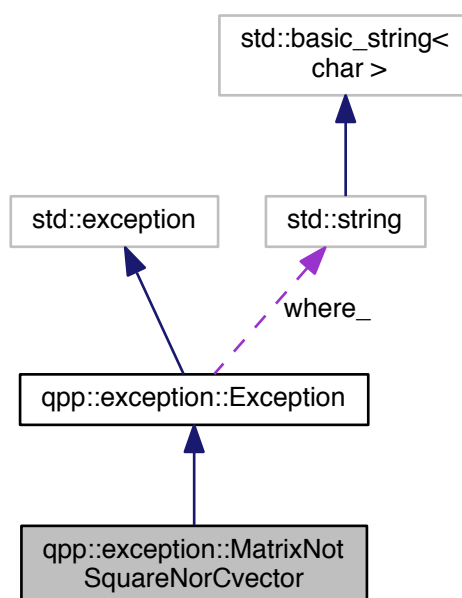
Matrix is not square nor column vector exception.

```
#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.*

### 7.26.1 Detailed Description

Matrix is not square nor column vector exception.

Eigen::Matrix is not a square matrix nor a column vector

### 7.26.2 Member Function Documentation

**7.26.2.1** `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:

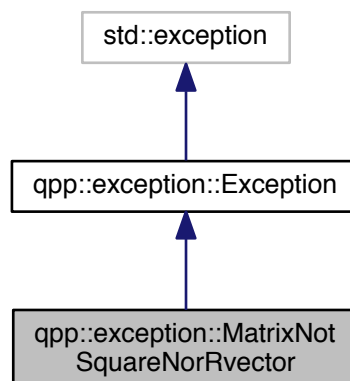
- `classes/exception.h`

## 7.27 qpp::exception::MatrixNotSquareNorRvector Class Reference

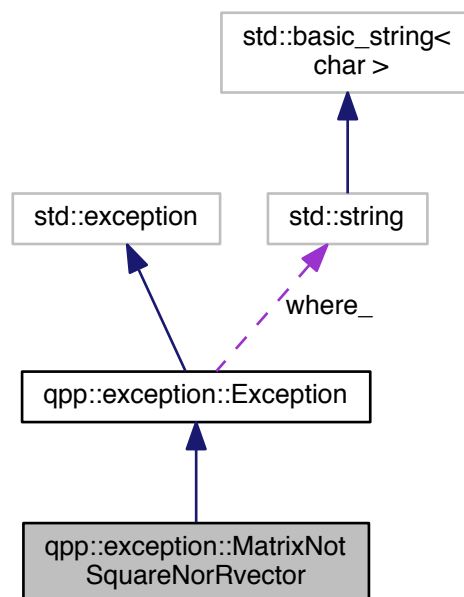
Matrix is not square nor row vector exception.

```
#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.*

### 7.27.1 Detailed Description

Matrix is not square nor row vector exception.

Eigen::Matrix is not a square matrix nor a row vector

### 7.27.2 Member Function Documentation

**7.27.2.1** `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:

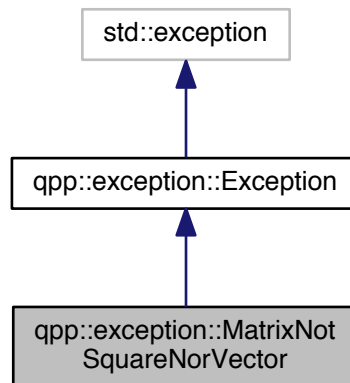
- `classes/exception.h`

## 7.28 qpp::exception::MatrixNotSquareNorVector Class Reference

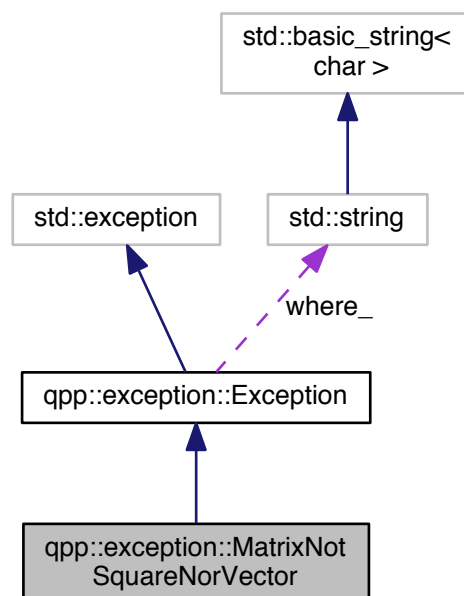
Matrix is not square nor vector exception.

```
#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.*

### 7.28.1 Detailed Description

Matrix is not square nor vector exception.

Eigen::Matrix is not a square matrix nor a row/column vector

### 7.28.2 Member Function Documentation

**7.28.2.1** `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:

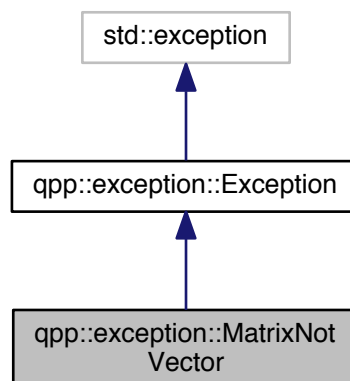
- `classes/exception.h`

## 7.29 qpp::exception::MatrixNotVector Class Reference

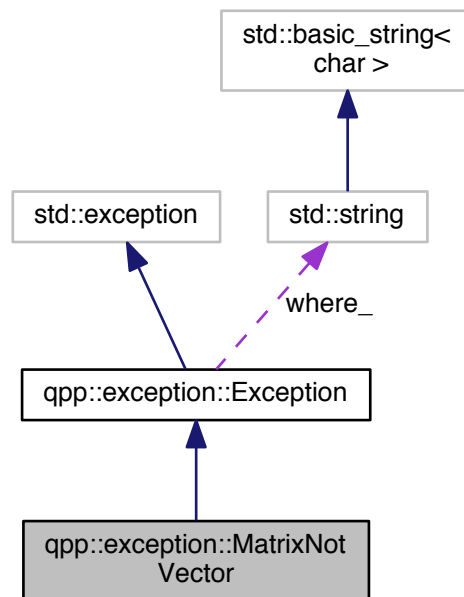
Matrix is not a vector exception.

```
#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.*

### 7.29.1 Detailed Description

Matrix is not a vector exception.

Eigen::Matrix is not a row or column vector

### 7.29.2 Member Function Documentation

**7.29.2.1** `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`

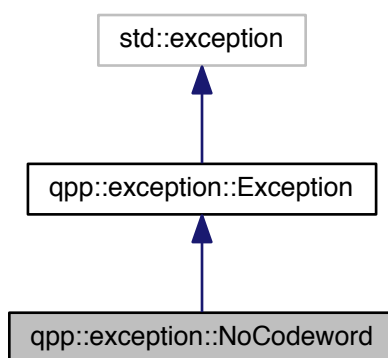


## 7.30 qpp::exception::NoCodeword Class Reference

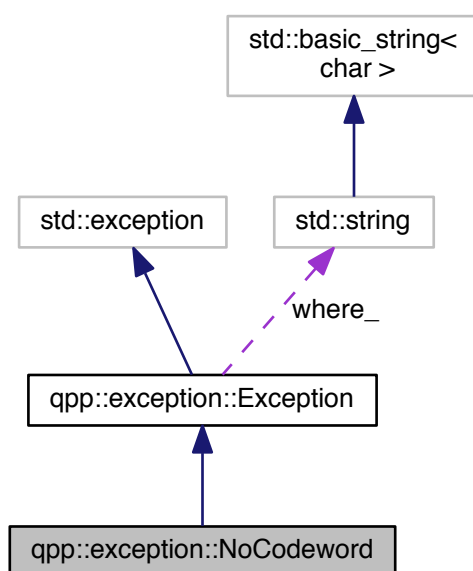
Codeword does not exist exception.

```
#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.*

### 7.30.1 Detailed Description

Codeword does not exist exception.

Codeword does not exist, thrown when calling `qpp::Codes::codeword()` with an invalid index

### 7.30.2 Member Function Documentation

**7.30.2.1** `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:

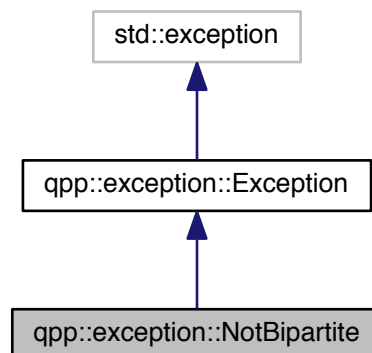
- `classes/exception.h`

## 7.31 qpp::exception::NotBipartite Class Reference

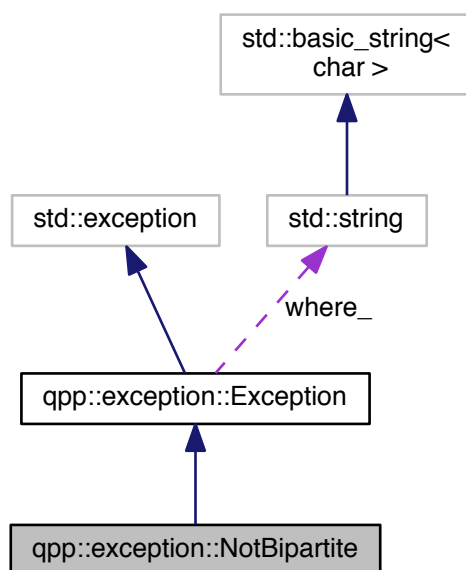
Not bi-partite exception.

```
#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.*

### 7.31.1 Detailed Description

Not bi-partite exception.

`std::vector<idx>` of dimensions has size different from 2

### 7.31.2 Member Function Documentation

7.31.2.1 `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:

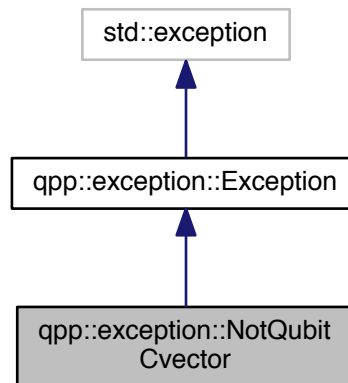
- `classes/exception.h`

## 7.32 qpp::exception::NotQubitCvector Class Reference

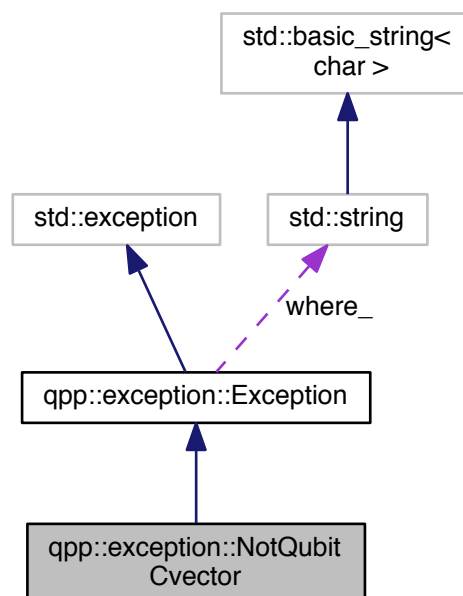
Column vector is not 2 x 1 exception.

```
#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.*

### 7.32.1 Detailed Description

Column vector is not 2 x 1 exception.

Eigen::Matrix is not 2 x 1

### 7.32.2 Member Function Documentation

**7.32.2.1** `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:

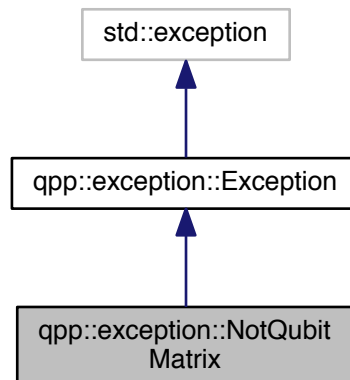
- `classes/exception.h`

## 7.33 qpp::exception::NotQubitMatrix Class Reference

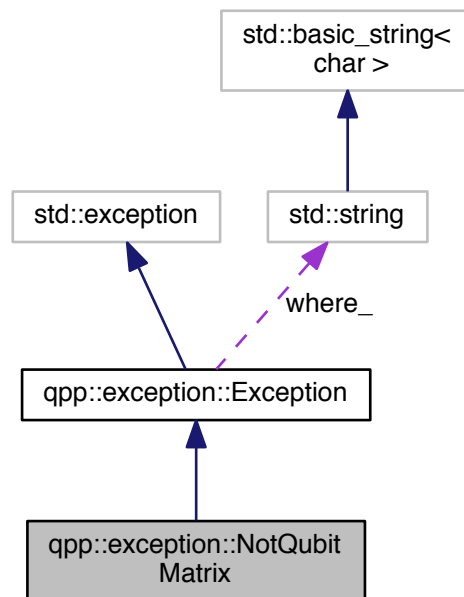
Matrix is not 2 x 2 exception.

```
#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.*

### 7.33.1 Detailed Description

Matrix is not 2 x 2 exception.

Eigen::Matrix is not 2 x 2

### 7.33.2 Member Function Documentation

**7.33.2.1** `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:

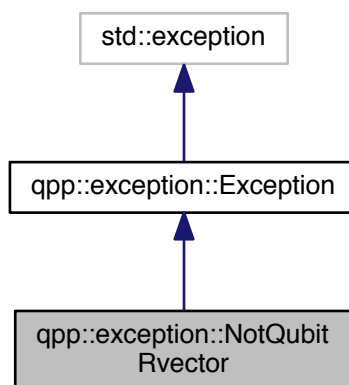
- `classes/exception.h`

## 7.34 qpp::exception::NotQubitRvector Class Reference

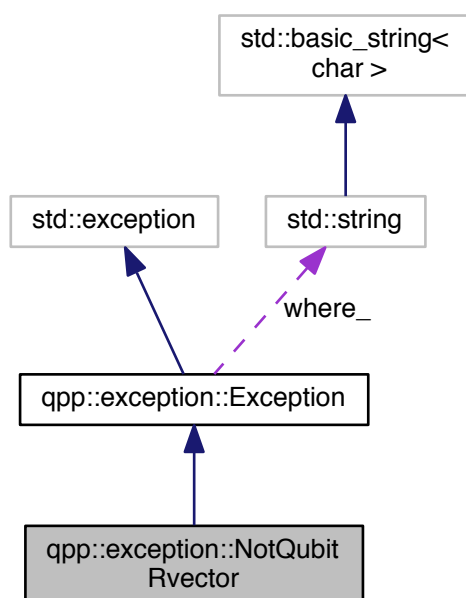
Row vector is not 1 x 2 exception.

```
#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.*

### 7.34.1 Detailed Description

Row vector is not 1 x 2 exception.

Eigen::Matrix is not 1 x 2

### 7.34.2 Member Function Documentation

**7.34.2.1** `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:

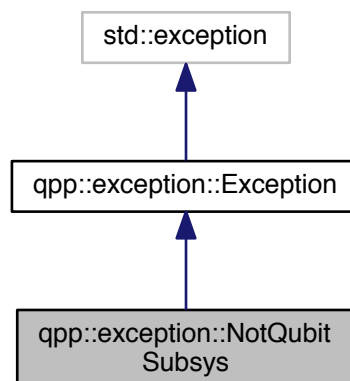
- `classes/exception.h`

## 7.35 qpp::exception::NotQubitSubsys Class Reference

Subsystems are not qubits exception.

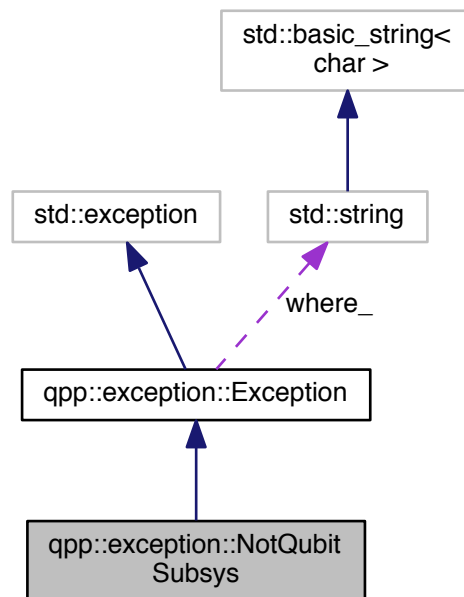
```
#include <classes/exception.h>
```

Inheritance diagram for `qpp::exception::NotQubitSubsys`:





Collaboration diagram for qpp::exception::NotQubitSubsys:



## Public Member Functions

- `std::string type_description () const` override  
*Exception type description.*

### 7.35.1 Detailed Description

Subsystems are not qubits exception.

Subsystems are not 2-dimensional (qubits)

### 7.35.2 Member Function Documentation

**7.35.2.1** `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:

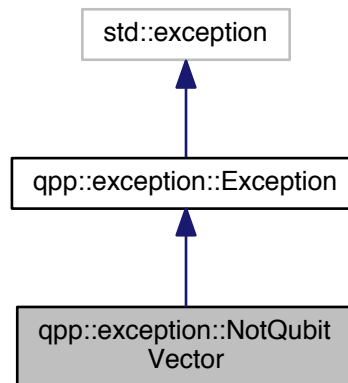
- `classes/exception.h`

## 7.36 qpp::exception::NotQubitVector Class Reference

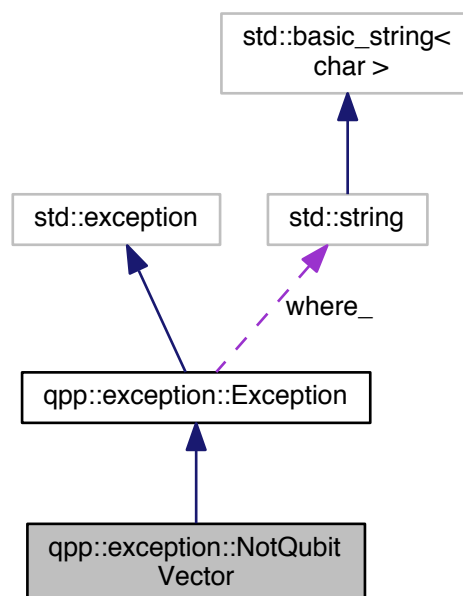
Vector is not 2 x 1 nor 1 x 2 exception.

```
#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.*

### 7.36.1 Detailed Description

Vector is not 2 x 1 nor 1 x 2 exception.

Eigen::Matrix is not 2 x 1 nor 1 x 2

### 7.36.2 Member Function Documentation

**7.36.2.1** `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:

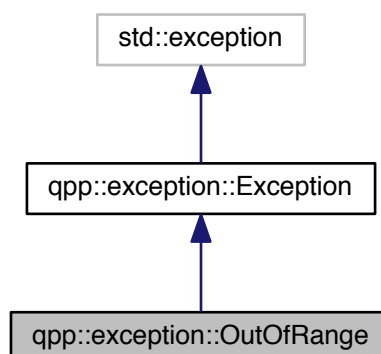
- `classes/exception.h`

## 7.37 qpp::exception::OutOfRange Class Reference

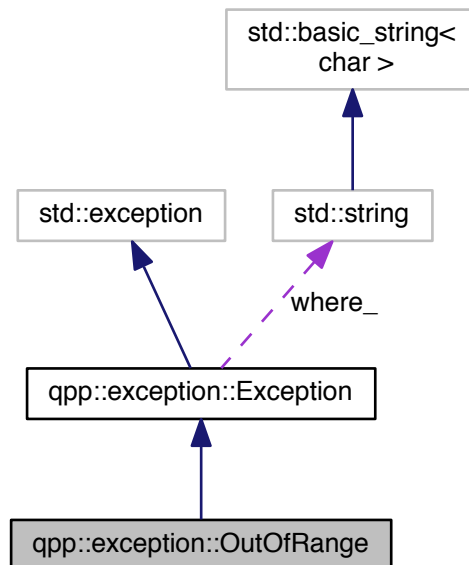
Parameter out of range exception.

```
#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.*

### 7.37.1 Detailed Description

Parameter out of range exception.

Parameter out of range

### 7.37.2 Member Function Documentation

**7.37.2.1** `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:

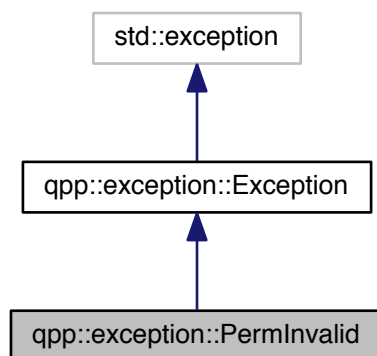
- `classes/exception.h`

## 7.38 qpp::exception::PermInvalid Class Reference

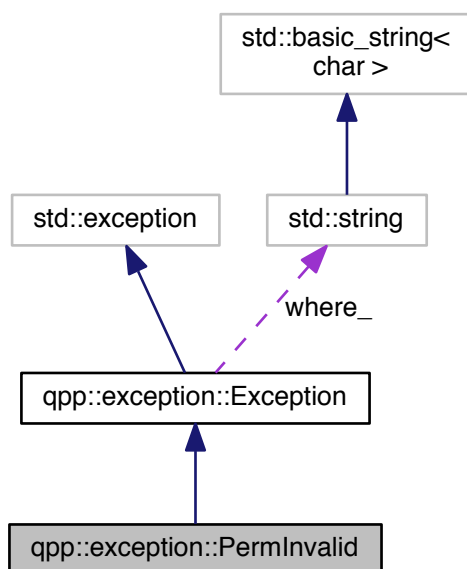
Invalid permutation exception.

```
#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.*

### 7.38.1 Detailed Description

Invalid permutation exception.

`std::vector<idx>` does not represent a valid permutation

### 7.38.2 Member Function Documentation

7.38.2.1 `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:

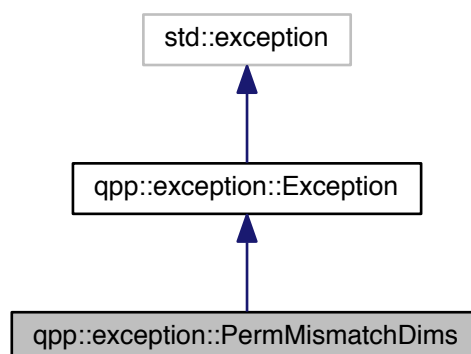
- `classes/exception.h`

## 7.39 qpp::exception::PermMismatchDims Class Reference

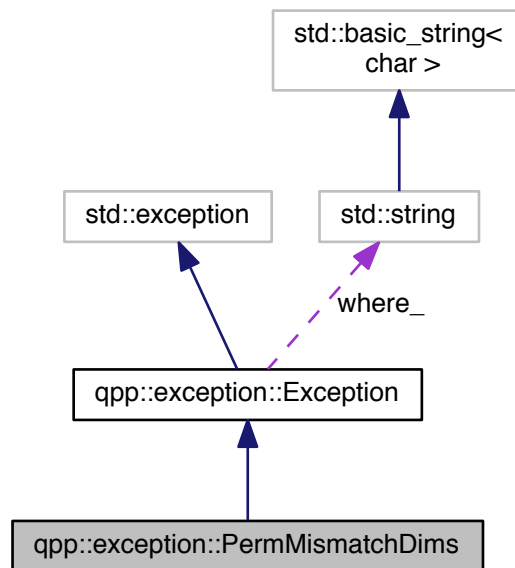
Permutation mismatch dimensions exception.

```
#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.*

### 7.39.1 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

### 7.39.2 Member Function Documentation

**7.39.2.1** `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:

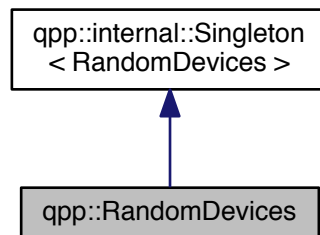
- `classes/exception.h`

## 7.40 qpp::RandomDevices Class Reference

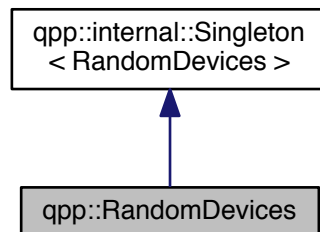
Singleton class that manages the source of randomness in the library.

```
#include <classes/random_devices.h>
```

Inheritance diagram for qpp::RandomDevices:



Collaboration diagram for qpp::RandomDevices:



### Public Attributes

- `std::mt19937 rng_`  
*Mersenne twister random number generator.*

### 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 rng\_*

## Friends

- class `internal::Singleton< RandomDevices >`

## Additional Inherited Members

### 7.40.1 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!

### 7.40.2 Constructor & Destructor Documentation

**7.40.2.1** `qpp::RandomDevices::RandomDevices ( )` `[inline]`, `[private]`

Initializes and seeds the random number generators.

**7.40.2.2** `qpp::RandomDevices::~~RandomDevices ( )` `[private]`, `[default]`

Default destructor.

### 7.40.3 Friends And Related Function Documentation

**7.40.3.1** `friend class internal::Singleton< RandomDevices >` `[friend]`

### 7.40.4 Member Data Documentation

**7.40.4.1** `std::random_device qpp::RandomDevices::rd_` `[private]`

used to seed std::mt19937 rng\_

**7.40.4.2** `std::mt19937 qpp::RandomDevices::rng_`

Mersenne twister random number generator.

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

- `classes/random_devices.h`

## 7.41 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

#### 7.41.1 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.

### 7.41.2 Constructor & Destructor Documentation

7.41.2.1 `template<typename T> qpp::internal::Singleton< T >::Singleton ( )` [protected], [default], [noexcept]

7.41.2.2 `template<typename T> qpp::internal::Singleton< T >::Singleton ( const Singleton< T > & )` [protected], [delete]

7.41.2.3 `template<typename T> virtual qpp::internal::Singleton< T >::~~Singleton ( )` [protected], [virtual], [default]

### 7.41.3 Member Function Documentation

7.41.3.1 `template<typename T> static T& qpp::internal::Singleton< T >::get_instance ( )` [inline], [static], [noexcept]

7.41.3.2 `template<typename T> static T& qpp::internal::Singleton< T >::get_thread_local_instance ( )` [inline], [static], [noexcept]

7.41.3.3 `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:

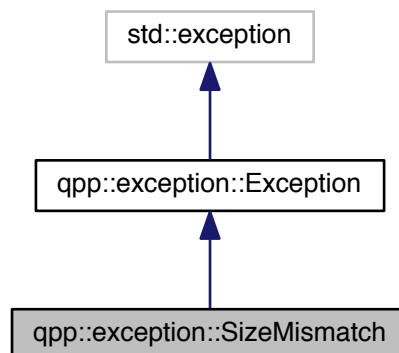
- [internal/classes/singleton.h](#)

## 7.42 qpp::exception::SizeMismatch Class Reference

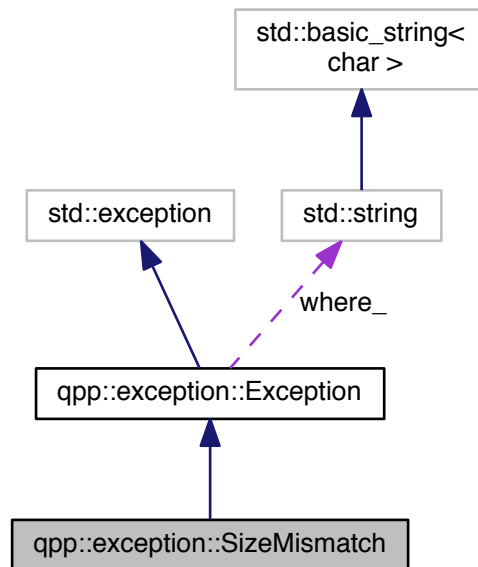
Size mismatch exception.

```
#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.*

### 7.42.1 Detailed Description

Size mismatch exception.

Sizes do not match

### 7.42.2 Member Function Documentation

**7.42.2.1** `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:

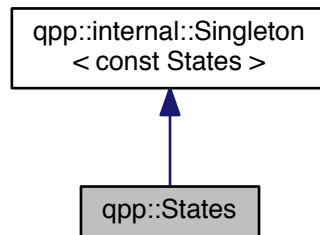
- `classes/exception.h`

## 7.43 qpp::States Class Reference

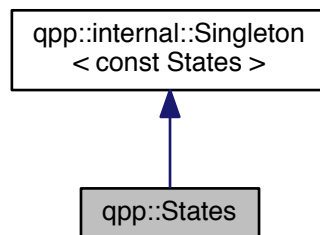
const Singleton class that implements most commonly used states

```
#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

### 7.43.1 Detailed Description

const Singleton class that implements most commonly used states

### 7.43.2 Constructor & Destructor Documentation

#### 7.43.2.1 `qpp::States::States ( ) [inline], [private]`

Initialize the states

#### 7.43.2.2 `qpp::States::~~States ( ) [private], [default]`

Default destructor.

### 7.43.3 Member Function Documentation

#### 7.43.3.1 `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

#### 7.43.3.2 `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

7.43.3.3 `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

7.43.3.4 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

7.43.3.5 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

7.43.3.6 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

## 7.43.4 Friends And Related Function Documentation

## 7.43.4.1 friend class internal::Singleton&lt; const States &gt; [friend]

## 7.43.5 Member Data Documentation

## 7.43.5.1 ket qpp::States::b00 {ket::Zero(4)}

Bell-00 state (following the convention in Nielsen and Chuang)

7.43.5.2 `ket qpp::States::b01 {ket::Zero(4)}`

Bell-01 state (following the convention in Nielsen and Chuang)

7.43.5.3 `ket qpp::States::b10 {ket::Zero(4)}`

Bell-10 state (following the convention in Nielsen and Chuang)

7.43.5.4 `ket qpp::States::b11 {ket::Zero(4)}`

Bell-11 state (following the convention in Nielsen and Chuang)

7.43.5.5 `ket qpp::States::GHZ {ket::Zero(8)}`

GHZ state.

7.43.5.6 `cmat qpp::States::pb00 {cmat::Zero(4, 4)}`

Projector onto the Bell-00 state.

7.43.5.7 `cmat qpp::States::pb01 {cmat::Zero(4, 4)}`

Projector onto the Bell-01 state.

7.43.5.8 `cmat qpp::States::pb10 {cmat::Zero(4, 4)}`

Projector onto the Bell-10 state.

7.43.5.9 `cmat qpp::States::pb11 {cmat::Zero(4, 4)}`

Projector onto the Bell-11 state.

7.43.5.10 `cmat qpp::States::pGHZ {cmat::Zero(8, 8)}`

Projector onto the GHZ state.

7.43.5.11 `cmat qpp::States::pW {cmat::Zero(8, 8)}`

Projector onto the W state.

7.43.5.12 `cmat qpp::States::px0 {cmat::Zero(2, 2)}`

Projector onto the Pauli Sigma-X 0-eigenstate  $|+\rangle\langle+|$ .

7.43.5.13 `cmat qpp::States::px1 {cmat::Zero(2, 2)}`

Projector onto the Pauli Sigma-X 1-eigenstate  $|-\rangle\langle-|$ .

7.43.5.14 `cmat qpp::States::py0 {cmat::Zero(2, 2)}`

Projector onto the Pauli Sigma-Y 0-eigenstate  $|y+\rangle\langle y+|$ .

7.43.5.15 `cmat qpp::States::py1 {cmat::Zero(2, 2)}`

Projector onto the Pauli Sigma-Y 1-eigenstate  $|y-\rangle\langle y-|$ .

7.43.5.16 `cmat qpp::States::pz0 {cmat::Zero(2, 2)}`

Projector onto the Pauli Sigma-Z 0-eigenstate  $|0\rangle\langle 0|$ .

7.43.5.17 `cmat qpp::States::pz1 {cmat::Zero(2, 2)}`

Projector onto the Pauli Sigma-Z 1-eigenstate  $|1\rangle\langle 1|$ .

7.43.5.18 `ket qpp::States::W {ket::Zero(8)}`

W state.

7.43.5.19 `ket qpp::States::x0 {ket::Zero(2)}`

Pauli Sigma-X 0-eigenstate  $|+\rangle$

7.43.5.20 `ket qpp::States::x1 {ket::Zero(2)}`

Pauli Sigma-X 1-eigenstate  $|-\rangle$

7.43.5.21 `ket qpp::States::y0 {ket::Zero(2)}`

Pauli Sigma-Y 0-eigenstate  $|y+\rangle$

7.43.5.22 `ket qpp::States::y1 {ket::Zero(2)}`

Pauli Sigma-Y 1-eigenstate  $|y-\rangle$

7.43.5.23 `ket qpp::States::z0 {ket::Zero(2)}`

Pauli Sigma-Z 0-eigenstate  $|0\rangle$

7.43.5.24 `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:

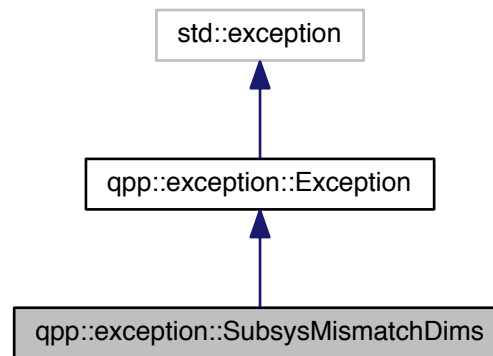
- [classes/states.h](#)

## 7.44 qpp::exception::SubsysMismatchDims Class Reference

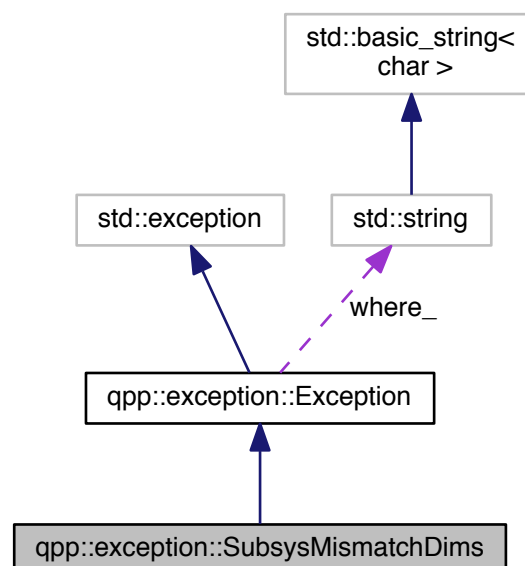
Subsystems mismatch dimensions exception.

```
#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.*

### 7.44.1 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

### 7.44.2 Member Function Documentation

**7.44.2.1** `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:

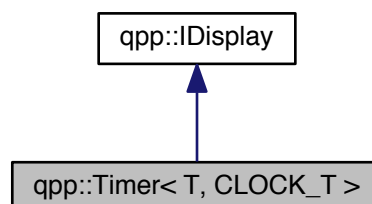
- `classes/exception.h`

## 7.45 qpp::Timer< T, CLOCK\_T > Class Template Reference

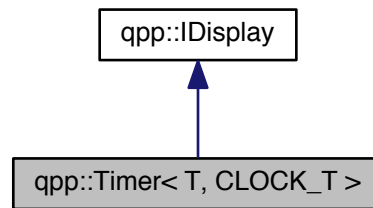
Chronometer.

```
#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*

### 7.45.1 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

## 7.45.2 Constructor &amp; Destructor Documentation

7.45.2.1 `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.

7.45.2.2 `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.

7.45.2.3 `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.

7.45.2.4 `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.

## 7.45.3 Member Function Documentation

7.45.3.1 `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

<i>os</i>	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](#).

7.45.3.2 `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

<i>U</i>	Duration, default is T, which defaults to std::chrono::duration<double, 1>, i.e. seconds in double precision
----------	--

## Returns

Duration that passed between the instantiation/reset and invocation of [qpp::Timer::toc\(\)](#)

**7.45.3.3** `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.

**7.45.3.4** `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.

**7.45.3.5** `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

**7.45.3.6** `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\(\)](#)

**7.45.3.7** `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

Current instance

## 7.45.4 Member Data Documentation

**7.45.4.1** `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]`

**7.45.4.2** `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:

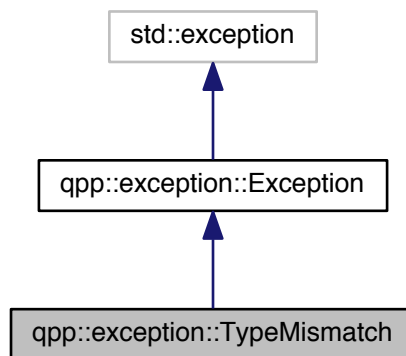
- [classes/timer.h](#)

## 7.46 qpp::exception::TypeMismatch Class Reference

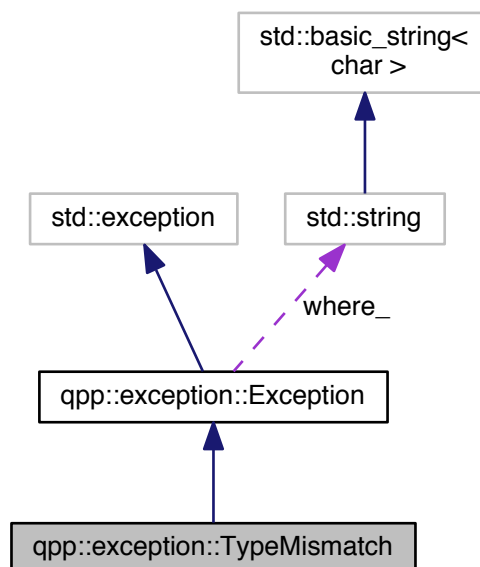
Type mismatch exception.

```
#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.*

### 7.46.1 Detailed Description

Type mismatch exception.

Scalar types do not match

### 7.46.2 Member Function Documentation

**7.46.2.1** `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:

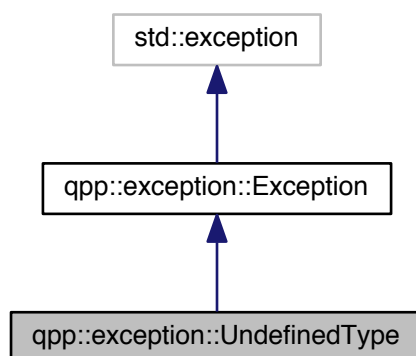
- `classes/exception.h`

## 7.47 qpp::exception::UndefinedType Class Reference

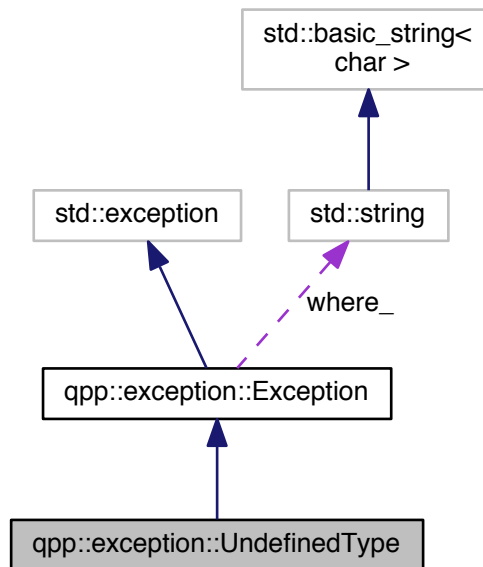
Not defined for this type exception.

```
#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.*

### 7.47.1 Detailed Description

Not defined for this type exception.

Templated specialization is not defined for this type

### 7.47.2 Member Function Documentation

**7.47.2.1** `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:

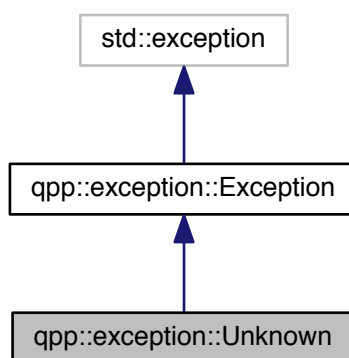
- `classes/exception.h`

## 7.48 qpp::exception::Unknown Class Reference

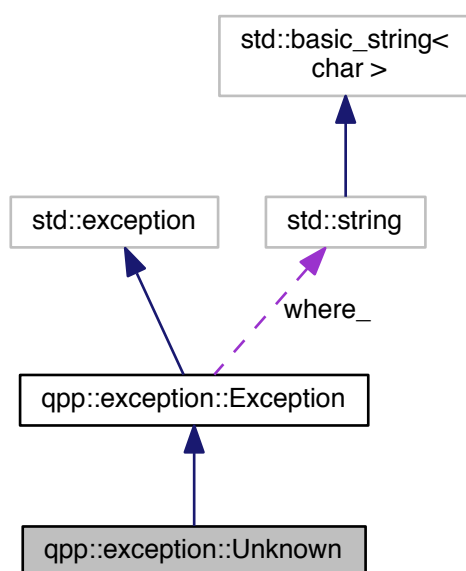
Unknown exception.

```
#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.*

### 7.48.1 Detailed Description

Unknown exception.

Thrown when no other exception is suitable (not recommended, it is better to define another suitable exception type)

### 7.48.2 Member Function Documentation

7.48.2.1 `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:

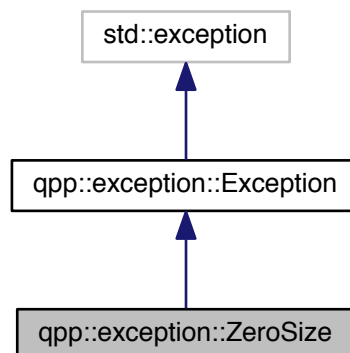
- `classes/exception.h`

## 7.49 qpp::exception::ZeroSize Class Reference

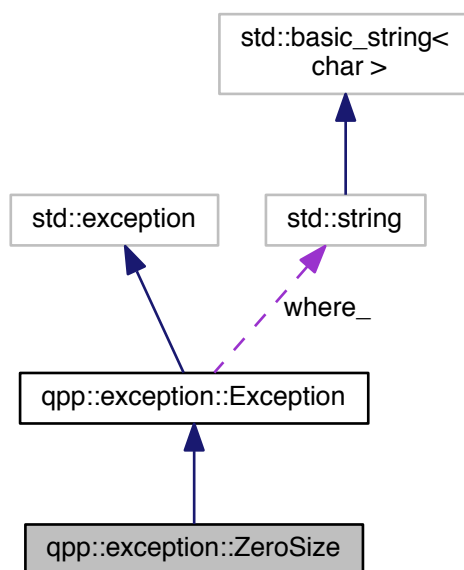
Object has zero size exception.

```
#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.*

### 7.49.1 Detailed Description

Object has zero size exception.

Zero sized object, e.g. empty `Eigen::Matrix` or `std::vector` with no elements

### 7.49.2 Member Function Documentation

**7.49.2.1** `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.*

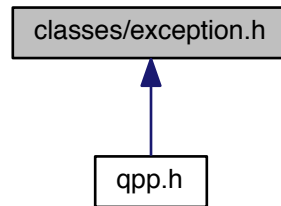
#### 8.1.1 Detailed Description

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::PermInvalid](#)  
*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.*

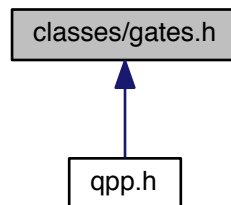
### 8.2.1 Detailed Description

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.*

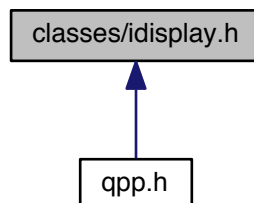
#### 8.3.1 Detailed Description

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.*

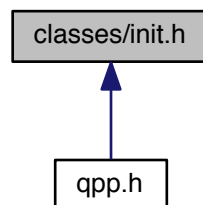
### 8.4.1 Detailed Description

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.*

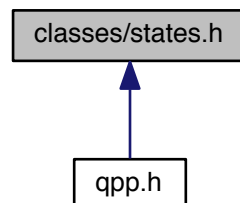
### 8.6.1 Detailed Description

Random devices.

## 8.7 classes/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.*

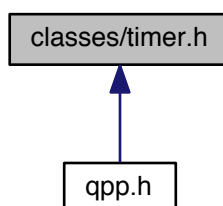
### 8.7.1 Detailed Description

Quantum states.

## 8.8 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.*

### 8.8.1 Detailed Description

Timing.

## 8.9 constants.h File Reference

Constants.

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



### Namespaces

- [qpp](#)  
*Quantum++ main namespace.*

### Functions

- constexpr cplx [qpp::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>::infinity()  
*Used to denote infinity in double precision.*

### 8.9.1 Detailed Description

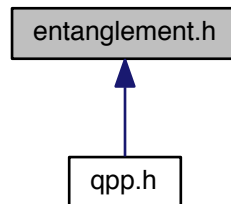
Constants.



## 8.10 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.*

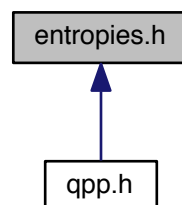
### 8.10.1 Detailed Description

Entanglement functions.

## 8.11 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.*

### 8.11.1 Detailed Description

Entropy functions.

## 8.12 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.*

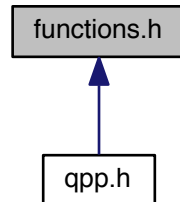
### 8.12.1 Detailed Description

Experimental/test functions/classes.

## 8.13 functions.h File Reference

Generic quantum computing functions.

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



### Namespaces

- [qpp](#)  
*Quantum++ main namespace.*

### 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::evects](#) (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::hevects](#) (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.*

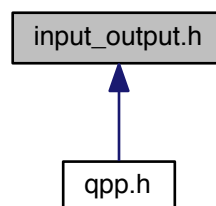
### 8.13.1 Detailed Description

Generic quantum computing functions.

## 8.14 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.*

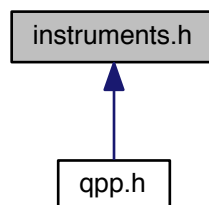
### 8.14.1 Detailed Description

Input/output functions.

## 8.15 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.*

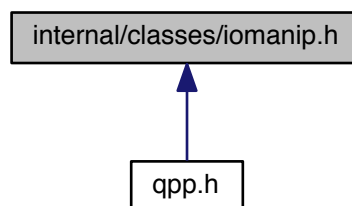
### 8.15.1 Detailed Description

Measurement functions.

## 8.16 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.*

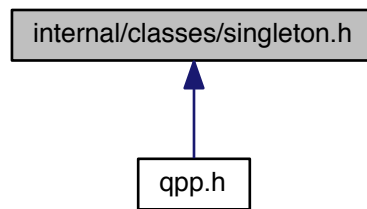
### 8.16.1 Detailed Description

Input/output manipulators.

## 8.17 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.*

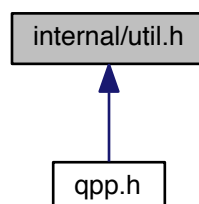
### 8.17.1 Detailed Description

Singleton pattern via CRTP.

## 8.18 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)`

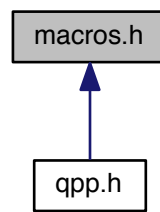
### 8.18.1 Detailed Description

Internal utility functions.

## 8.19 macros.h File Reference

Preprocessor macros.

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



### Macros

- `#define PRINT(x) std::cout << (x)`
- `#define PRINTLN(x) std::cout << (x) << std::endl`
- `#define ERROR(x) std::cerr << (x)`
- `#define ERRORLN(x) std::cerr << (x) << std::endl`

### 8.19.1 Detailed Description

Preprocessor macros.

### 8.19.2 Macro Definition Documentation

#### 8.19.2.1 `#define ERROR( x ) std::cerr << (x)`

Prints an error message to std::cerr

#### 8.19.2.2 `#define ERRORLN( x ) std::cerr << (x) << std::endl`

Prints an error message to std::cerr and adds a new line

**8.19.2.3** `#define PRINT( x ) std::cout << (x)`

Prints a message

**8.19.2.4** `#define PRINTLN( x ) std::cout << (x) << std::endl`

Prints a message and adds a new line

## 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,.*

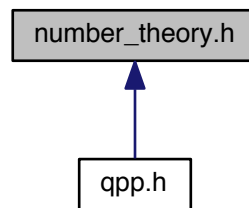
### 8.20.1 Detailed Description

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].*



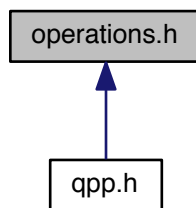
### 8.21.1 Detailed Description

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 <iostream>
#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 "macros.h"
#include "types.h"
#include "classes/exception.h"
#include "constants.h"
#include "traits.h"
#include "classes/ideisplay.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"
```

## Namespaces

- [qpp](#)

*Quantum++ main namespace.*

## Macros

- `#define QPP_UNUSED_`

### 8.23.1 Detailed Description

Quantum++ main header file, includes all other necessary headers.

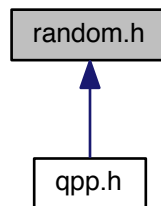
### 8.23.2 Macro Definition Documentation

#### 8.23.2.1 `#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.*

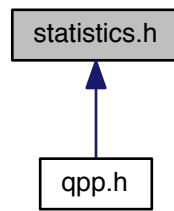
### 8.24.1 Detailed Description

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.*

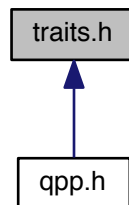
### 8.25.1 Detailed Description

Statistics functions.

## 8.26 traits.h File Reference

Type traits.

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



### Classes

- struct [qpp::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... >  
using [qpp::to\\_void](#) = void  
*Alias template that implements the proposal for void\_t.*

#### 8.26.1 Detailed Description

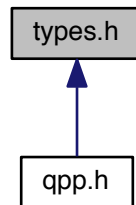
Type traits.



## 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.

# Index

- ~Codes
  - qpp::Codes, 90
- ~Gates
  - qpp::Gates, 109
- ~IDisplay
  - qpp::IDisplay, 115
- ~Init
  - qpp::Init, 117
- ~RandomDevices
  - qpp::RandomDevices, 157
- ~Singleton
  - qpp::internal::Singleton, 159
- ~States
  - qpp::States, 163
- ~Timer
  - qpp::Timer, 172
- A\_
  - qpp::internal::IOManipEigen, 119
- absm
  - qpp, 28
- abssq
  - qpp, 29
- adjoint
  - qpp, 29
- anticomm
  - qpp, 30
- apply
  - qpp, 30, 31
- applyCTRL
  - qpp, 32
- avg
  - qpp, 33
- b00
  - qpp::States, 165
- b01
  - qpp::States, 165
- b10
  - qpp::States, 166
- b11
  - qpp::States, 166
- bigint
  - qpp, 27
- bloch2rho
  - qpp, 33
- bra
  - qpp, 27
- CNOT
  - qpp::Gates, 112
- CNOTba
  - qpp::Gates, 112
- CTRL
  - qpp::Gates, 109
- CZ
  - qpp::Gates, 112
- check\_cvector
  - qpp::internal, 87
- check\_dims
  - qpp::internal, 87
- check\_dims\_match\_cvect
  - qpp::internal, 87
- check\_dims\_match\_mat
  - qpp::internal, 87
- check\_dims\_match\_rvect
  - qpp::internal, 87
- check\_eq\_dims
  - qpp::internal, 87
- check\_matching\_sizes
  - qpp::internal, 88
- check\_nonzero\_size
  - qpp::internal, 88
- check\_perm
  - qpp::internal, 88
- check\_qubit\_cvector
  - qpp::internal, 88
- check\_qubit\_matrix
  - qpp::internal, 88
- check\_qubit\_rvector
  - qpp::internal, 88
- check\_qubit\_vector
  - qpp::internal, 88
- check\_rvector
  - qpp::internal, 88
- check\_square\_mat
  - qpp::internal, 88
- check\_subsys\_match\_dims
  - qpp::internal, 88
- check\_vector
  - qpp::internal, 88
- choi2kraus
  - qpp, 33
- choi2super
  - qpp, 34
- chop
  - qpp, 84
- chop\_
  - qpp::internal::IOManipEigen, 119

classes/codes.h, 181  
 classes/exception.h, 181  
 classes/gates.h, 184  
 classes/ideisplay.h, 184  
 classes/init.h, 185  
 classes/random\_devices.h, 186  
 classes/states.h, 186  
 classes/timer.h, 187  
 cmat  
     qpp, 27  
 Codes  
     qpp::Codes, 90  
 codeword  
     qpp::Codes, 91  
 comm  
     qpp, 34  
 complement  
     qpp, 34  
 compperm  
     qpp, 34  
 concurrence  
     qpp, 36  
 conjugate  
     qpp, 36  
 constants.h, 188  
 contrfrac2x  
     qpp, 36  
 cor  
     qpp, 36  
 cosm  
     qpp, 37  
 cov  
     qpp, 37  
 cplx  
     qpp, 27  
 CustomException  
     qpp::exception::CustomException, 92  
 cwise  
     qpp, 37  
 det  
     qpp, 37  
 dirsum  
     qpp, 38, 39  
 dirsum2  
     qpp::internal, 88  
 dirsumpow  
     qpp, 39  
 disp  
     qpp, 39, 40  
 display  
     qpp::IDisplay, 115  
     qpp::Timer, 172  
     qpp::internal::IOManipEigen, 118  
     qpp::internal::IOManipPointer, 120  
     qpp::internal::IOManipRange, 122  
 display\_impl\_  
     qpp::internal::Display\_Impl\_, 103  
 dmat  
     qpp, 27  
 dyn\_col\_vect  
     qpp, 27  
 dyn\_mat  
     qpp, 28  
 dyn\_row\_vect  
     qpp, 28  
 ERROR  
     macros.h, 202  
 ERRORLN  
     macros.h, 202  
 ee  
     qpp, 84  
 egcd  
     qpp, 42  
 eig  
     qpp, 42  
 end\_  
     qpp::Timer, 173  
     qpp::internal::IOManipPointer, 120  
     qpp::internal::IOManipRange, 122  
 entanglement  
     qpp, 42, 43  
 entanglement.h, 189  
 entropies.h, 190  
 entropy  
     qpp, 43  
 eps  
     qpp, 84  
 evals  
     qpp, 43  
 evects  
     qpp, 44  
 Exception  
     qpp::exception::Exception, 106  
 expandout  
     qpp::Gates, 109, 110  
 experimental/experimental.h, 191  
 expm  
     qpp, 44  
 FIVE\_QUBIT  
     qpp::Codes, 90  
 FRED  
     qpp::Gates, 112  
 factors  
     qpp, 44  
 Fd  
     qpp::Gates, 111  
 first\_  
     qpp::internal::IOManipRange, 122  
 functions.h, 192  
 funm  
     qpp, 44  
 GHZ  
     qpp::States, 166  
 Gates

- qpp::Gates, 109
- gcd
  - qpp, 45
- gconcurrency
  - qpp, 45
- get\_dim\_subsys
  - qpp::internal, 88
- get\_duration
  - qpp::Timer, 172
- get\_instance
  - qpp::internal::Singleton, 159
- get\_num\_subsys
  - qpp::internal, 88
- get\_thread\_local\_instance
  - qpp::internal::Singleton, 159
- grams
  - qpp, 46
- H
  - qpp::Gates, 112
- heig
  - qpp, 46
- hevals
  - qpp, 47
- hevects
  - qpp, 47
- IDisplay
  - qpp::IDisplay, 115
- IOManipEigen
  - qpp::internal::IOManipEigen, 118
- IOManipPointer
  - qpp::internal::IOManipPointer, 120
- IOManipRange
  - qpp::internal::IOManipRange, 122
- Id
  - qpp::Gates, 111
- Id2
  - qpp::Gates, 113
- idx
  - qpp, 28
- infty
  - qpp, 84
- Init
  - qpp::Init, 117
- input\_output.h, 196
- instruments.h, 197
- internal/classes/iomanip.h, 199
- internal/classes/singleton.h, 199
- internal/util.h, 200
- internal::Singleton< const Codes >
  - qpp::Codes, 91
- internal::Singleton< const Gates >
  - qpp::Gates, 112
- internal::Singleton< const Init >
  - qpp::Init, 117
- internal::Singleton< const States >
  - qpp::States, 165
- internal::Singleton< RandomDevices >
  - qpp::RandomDevices, 157
- inverse
  - qpp, 47
- invperm
  - qpp, 47
- ip
  - qpp, 48
- isprime
  - qpp, 48
- jn
  - qpp::States, 163
- ket
  - qpp, 28
- kraus2choi
  - qpp, 48
- kraus2super
  - qpp, 49
- kron
  - qpp, 49, 50
- kron2
  - qpp::internal, 88
- kronpow
  - qpp, 50
- last\_
  - qpp::internal::IOManipRange, 122
- lcm
  - qpp, 51
- load
  - qpp, 51
- loadMATLAB
  - qpp, 52
- logdet
  - qpp, 53
- logm
  - qpp, 53
- lognegativity
  - qpp, 53
- MATLAB/matlab.h, 203
- macros.h, 202
  - ERROR, 202
  - ERRORLN, 202
  - PRINT, 202
  - PRINTLN, 203
- marginalX
  - qpp, 54
- marginalY
  - qpp, 54
- maxn
  - qpp, 84
- measure
  - qpp, 54–57
- measure\_seq
  - qpp, 58
- mes
  - qpp::States, 163

minus  
     qpp::States, 163  
 mket  
     qpp, 59  
 modinv  
     qpp, 59  
 modmul  
     qpp, 60  
 modpow  
     qpp, 60  
 mprj  
     qpp, 60, 61  
 multiidx2n  
     qpp, 61  
     qpp::internal, 88  
  
 n2multiidx  
     qpp, 61  
     qpp::internal, 88  
 N\_  
     qpp::internal::IOManipPointer, 120  
 NINE\_QUBIT\_SHOR  
     qpp::Codes, 90  
 negativity  
     qpp, 62  
 norm  
     qpp, 62  
 number\_theory.h, 203  
  
 omega  
     qpp, 62  
 one  
     qpp::States, 165  
 operations.h, 205  
 operator<<  
     qpp::IDisplay, 115  
 operator=  
     qpp::IDisplay, 115  
     qpp::Timer, 173  
     qpp::internal::IOManipPointer, 120  
     qpp::internal::IOManipRange, 122  
     qpp::internal::Singleton, 159  
 operator""\_i  
     qpp, 63  
  
 p\_  
     qpp::internal::IOManipPointer, 120  
 pGHZ  
     qpp::States, 166  
 PRINT  
     macros.h, 202  
 PRINTLN  
     macros.h, 203  
 pW  
     qpp::States, 166  
 pb00  
     qpp::States, 166  
 pb01  
     qpp::States, 166  
  
 pb10  
     qpp::States, 166  
 pb11  
     qpp::States, 166  
 pi  
     qpp, 84  
 plus  
     qpp::States, 165  
 powm  
     qpp, 63  
 prj  
     qpp, 63  
 prod  
     qpp, 64  
 ptrace  
     qpp, 64, 65  
 ptrace1  
     qpp, 65  
 ptrace2  
     qpp, 66  
 ptranspose  
     qpp, 67  
 px0  
     qpp::States, 166  
 px1  
     qpp::States, 166  
 py0  
     qpp::States, 166  
 py1  
     qpp::States, 167  
 pz0  
     qpp::States, 167  
 pz1  
     qpp::States, 167  
  
 QPP\_UNUSED\_  
     qpp.h, 209  
 qmutualinfo  
     qpp, 67, 68  
 qpp, 15  
     absm, 28  
     abssq, 29  
     adjoint, 29  
     anticomm, 30  
     apply, 30, 31  
     applyCTRL, 32  
     avg, 33  
     bigint, 27  
     bloch2rho, 33  
     bra, 27  
     choi2kraus, 33  
     choi2super, 34  
     chop, 84  
     cmat, 27  
     comm, 34  
     complement, 34  
     compperm, 34  
     concurrence, 36  
     conjugate, 36

contfrac2x, 36  
 cor, 36  
 cosm, 37  
 cov, 37  
 cplx, 27  
 cwise, 37  
 det, 37  
 dirsum, 38, 39  
 dirsumpow, 39  
 disp, 39, 40  
 dmat, 27  
 dyn\_col\_vect, 27  
 dyn\_mat, 28  
 dyn\_row\_vect, 28  
 ee, 84  
 egcd, 42  
 eig, 42  
 entanglement, 42, 43  
 entropy, 43  
 eps, 84  
 evals, 43  
 evects, 44  
 expm, 44  
 factors, 44  
 funm, 44  
 gcd, 45  
 gconcurrency, 45  
 grams, 46  
 heig, 46  
 hevals, 47  
 hevects, 47  
 idx, 28  
 infty, 84  
 inverse, 47  
 invperm, 47  
 ip, 48  
 isprime, 48  
 ket, 28  
 kraus2choi, 48  
 kraus2super, 49  
 kron, 49, 50  
 kronpow, 50  
 lcm, 51  
 load, 51  
 loadMATLAB, 52  
 logdet, 53  
 logm, 53  
 lognegativity, 53  
 marginalX, 54  
 marginalY, 54  
 maxn, 84  
 measure, 54–57  
 measure\_seq, 58  
 mket, 59  
 modinv, 59  
 modmul, 60  
 modpow, 60  
 mprj, 60, 61  
 multiidx2n, 61  
 n2multiidx, 61  
 negativity, 62  
 norm, 62  
 omega, 62  
 operator""\_i, 63  
 pi, 84  
 powm, 63  
 prj, 63  
 prod, 64  
 ptrace, 64, 65  
 ptrace1, 65  
 ptrace2, 66  
 ptranspose, 67  
 qmutualinfo, 67, 68  
 rand, 68, 69  
 randH, 69  
 randU, 73  
 randV, 73  
 randidx, 70  
 randket, 70  
 randkraus, 70  
 randn, 70, 71  
 randperm, 72  
 randprime, 72  
 randprob, 72  
 randrho, 72  
 renyi, 73  
 reshape, 74  
 rho2bloch, 74  
 rho2pure, 74  
 save, 75  
 saveMATLAB, 75  
 schatten, 76  
 schmidtA, 76  
 schmidtB, 76, 77  
 schmidtcoeffs, 77  
 schmidtprobs, 78  
 sigma, 78  
 sinm, 79  
 spectralpowm, 79  
 sqrtm, 79  
 sum, 79, 80  
 super2choi, 80  
 svals, 80  
 svd, 81  
 svdU, 81  
 svdV, 81  
 syspermute, 81, 82  
 to\_void, 28  
 trace, 82  
 transpose, 82  
 tsallis, 82, 83  
 uniform, 83  
 var, 83  
 x2contfrac, 83  
 qpp.h, 207  
 QPP\_UNUSED\_, 209

- qpp::Codes, 89
  - ~Codes, 90
  - Codes, 90
  - codeword, 91
  - FIVE\_QUBIT, 90
  - internal::Singleton< const Codes >, 91
  - NINE\_QUBIT\_SHOR, 90
  - SEVEN\_QUBIT\_STEANE, 90
  - Type, 90
- qpp::Gates, 106
  - ~Gates, 109
  - CNOT, 112
  - CNOTba, 112
  - CTRL, 109
  - CZ, 112
  - expandout, 109, 110
  - FRED, 112
  - Fd, 111
  - Gates, 109
  - H, 112
  - Id, 111
  - Id2, 113
  - internal::Singleton< const Gates >, 112
  - Rn, 111
  - S, 113
  - SWAP, 113
  - T, 113
  - TOF, 113
  - X, 113
  - Xd, 111
  - Y, 113
  - Z, 113
  - Zd, 112
- qpp::IDisplay, 113
  - ~IDisplay, 115
  - display, 115
  - IDisplay, 115
  - operator<<, 115
  - operator=, 115
- qpp::Init, 116
  - ~Init, 117
  - Init, 117
  - internal::Singleton< const Init >, 117
- qpp::RandomDevices, 156
  - ~RandomDevices, 157
  - internal::Singleton< RandomDevices >, 157
  - RandomDevices, 157
  - rd\_, 157
  - rng\_, 157
- qpp::States, 161
  - ~States, 163
  - b00, 165
  - b01, 165
  - b10, 166
  - b11, 166
  - GHZ, 166
  - internal::Singleton< const States >, 165
  - jn, 163
  - mes, 163
  - minus, 163
  - one, 165
  - pGHZ, 166
  - pW, 166
  - pb00, 166
  - pb01, 166
  - pb10, 166
  - pb11, 166
  - plus, 165
  - px0, 166
  - px1, 166
  - py0, 166
  - py1, 167
  - pz0, 167
  - pz1, 167
  - States, 163
  - W, 167
  - x0, 167
  - x1, 167
  - y0, 167
  - y1, 167
  - z0, 167
  - z1, 167
  - zero, 165
- qpp::Timer
  - ~Timer, 172
  - display, 172
  - end\_, 173
  - get\_duration, 172
  - operator=, 173
  - start\_, 173
  - tic, 173
  - tics, 173
  - Timer, 172
  - toc, 173
- qpp::Timer< T, CLOCK\_T >, 169
- qpp::exception, 84
- qpp::exception::CustomException, 91
  - CustomException, 92
  - type\_description, 92
  - what\_, 93
- qpp::exception::DimsInvalid, 93
  - type\_description, 94
- qpp::exception::DimsMismatchCvector, 95
  - type\_description, 96
- qpp::exception::DimsMismatchMatrix, 96
  - type\_description, 97
- qpp::exception::DimsMismatchRvector, 98
  - type\_description, 99
- qpp::exception::DimsMismatchVector, 99
  - type\_description, 100
- qpp::exception::DimsNotEqual, 101
  - type\_description, 102
- qpp::exception::Exception, 103
  - Exception, 106
  - type\_description, 106
  - what, 106



- where\_, 106
- qpp::exception::MatrixMismatchSubsys, 128
  - type\_description, 130
- qpp::exception::MatrixNotCvector, 130
  - type\_description, 131
- qpp::exception::MatrixNotRvector, 132
  - type\_description, 133
- qpp::exception::MatrixNotSquare, 133
  - type\_description, 134
- qpp::exception::MatrixNotSquareNorCvector, 135
  - type\_description, 136
- qpp::exception::MatrixNotSquareNorRvector, 136
  - type\_description, 137
- qpp::exception::MatrixNotSquareNorVector, 138
  - type\_description, 139
- qpp::exception::MatrixNotVector, 139
  - type\_description, 140
- qpp::exception::NoCodeword, 141
  - type\_description, 142
- qpp::exception::NotBipartite, 142
  - type\_description, 143
- qpp::exception::NotQubitCvector, 144
  - type\_description, 145
- qpp::exception::NotQubitMatrix, 145
  - type\_description, 146
- qpp::exception::NotQubitRvector, 147
  - type\_description, 148
- qpp::exception::NotQubitSubsys, 148
  - type\_description, 149
- qpp::exception::NotQubitVector, 150
  - type\_description, 151
- qpp::exception::OutOfRange, 151
  - type\_description, 152
- qpp::exception::PermInvalid, 153
  - type\_description, 154
- qpp::exception::PermMismatchDims, 154
  - type\_description, 155
- qpp::exception::SizeMismatch, 159
  - type\_description, 160
- qpp::exception::SubsysMismatchDims, 168
  - type\_description, 169
- qpp::exception::TypeMismatch, 174
  - type\_description, 175
- qpp::exception::UndefinedType, 175
  - type\_description, 176
- qpp::exception::Unknown, 177
  - type\_description, 178
- qpp::exception::ZeroSize, 178
  - type\_description, 179
- qpp::experimental, 86
- qpp::internal, 86
  - check\_cvector, 87
  - check\_dims, 87
  - check\_dims\_match\_cvect, 87
  - check\_dims\_match\_mat, 87
  - check\_dims\_match\_rvect, 87
  - check\_eq\_dims, 87
  - check\_matching\_sizes, 88
  - check\_nonzero\_size, 88
  - check\_perm, 88
  - check\_qubit\_cvector, 88
  - check\_qubit\_matrix, 88
  - check\_qubit\_rvector, 88
  - check\_qubit\_vector, 88
  - check\_rvector, 88
  - check\_square\_mat, 88
  - check\_subsys\_match\_dims, 88
  - check\_vector, 88
  - dirsum2, 88
  - get\_dim\_subsys, 88
  - get\_num\_subsys, 88
  - kron2, 88
  - multiidx2n, 88
  - n2multiidx, 88
  - variadic\_vector\_emplace, 88
- qpp::internal::Display\_Impl\_, 102
  - display\_impl\_, 103
- qpp::internal::IOManipEigen, 117
  - A\_, 119
  - chop\_, 119
  - display, 118
  - IOManipEigen, 118
- qpp::internal::IOManipPointer
  - display, 120
  - end\_, 120
  - IOManipPointer, 120
  - N\_, 120
  - operator=, 120
  - p\_, 120
  - separator\_, 120
  - start\_, 121
- qpp::internal::IOManipPointer< PointerType >, 119
- qpp::internal::IOManipRange
  - display, 122
  - end\_, 122
  - first\_, 122
  - IOManipRange, 122
  - last\_, 122
  - operator=, 122
  - separator\_, 122
  - start\_, 123
- qpp::internal::IOManipRange< InputIterator >, 121
- qpp::internal::Singleton
  - ~Singleton, 159
  - get\_instance, 159
  - get\_thread\_local\_instance, 159
  - operator=, 159
  - Singleton, 159
- qpp::internal::Singleton< T >, 158
- qpp::is\_complex< std::complex< T > >, 124
- qpp::is\_complex< T >, 123
- qpp::is\_iterable< T, to\_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().begin()) >, typename T::value\_type > >, 126
- qpp::is\_iterable< T, typename >, 125
- qpp::is\_matrix\_expression< Derived >, 127

- rand
  - qpp, [68](#), [69](#)
- randH
  - qpp, [69](#)
- randU
  - qpp, [73](#)
- randV
  - qpp, [73](#)
- randidx
  - qpp, [70](#)
- randket
  - qpp, [70](#)
- randkraus
  - qpp, [70](#)
- randn
  - qpp, [70](#), [71](#)
- random.h, [209](#)
- RandomDevices
  - qpp::RandomDevices, [157](#)
- randperm
  - qpp, [72](#)
- randprime
  - qpp, [72](#)
- randprob
  - qpp, [72](#)
- randrho
  - qpp, [72](#)
- rd\_
  - qpp::RandomDevices, [157](#)
- renyi
  - qpp, [73](#)
- reshape
  - qpp, [74](#)
- rho2bloch
  - qpp, [74](#)
- rho2pure
  - qpp, [74](#)
- Rn
  - qpp::Gates, [111](#)
- rng\_
  - qpp::RandomDevices, [157](#)
- S
  - qpp::Gates, [113](#)
- SEVEN\_QUBIT\_STEANE
  - qpp::Codes, [90](#)
- SWAP
  - qpp::Gates, [113](#)
- save
  - qpp, [75](#)
- saveMATLAB
  - qpp, [75](#)
- schatten
  - qpp, [76](#)
- schmidtA
  - qpp, [76](#)
- schmidtB
  - qpp, [76](#), [77](#)
- schmidtcoeffs
  - qpp, [77](#)
- schmidtprobs
  - qpp, [78](#)
- separator\_
  - qpp::internal::IOManipPointer, [120](#)
  - qpp::internal::IOManipRange, [122](#)
- sigma
  - qpp, [78](#)
- Singleton
  - qpp::internal::Singleton, [159](#)
- sinm
  - qpp, [79](#)
- spectralpowm
  - qpp, [79](#)
- sqrtn
  - qpp, [79](#)
- start\_
  - qpp::Timer, [173](#)
  - qpp::internal::IOManipPointer, [121](#)
  - qpp::internal::IOManipRange, [123](#)
- States
  - qpp::States, [163](#)
- statistics.h, [210](#)
- sum
  - qpp, [79](#), [80](#)
- super2choi
  - qpp, [80](#)
- svals
  - qpp, [80](#)
- svd
  - qpp, [81](#)
- svdU
  - qpp, [81](#)
- svdV
  - qpp, [81](#)
- syspermute
  - qpp, [81](#), [82](#)
- T
  - qpp::Gates, [113](#)
- TOF
  - qpp::Gates, [113](#)
- tic
  - qpp::Timer, [173](#)
- tics
  - qpp::Timer, [173](#)
- Timer
  - qpp::Timer, [172](#)
- to\_void
  - qpp, [28](#)
- toc
  - qpp::Timer, [173](#)
- trace
  - qpp, [82](#)
- traits.h, [212](#)
- transpose
  - qpp, [82](#)
- tsallis
  - qpp, [82](#), [83](#)

Type  
 qpp::Codes, 90  
 type\_description  
 qpp::exception::CustomException, 92  
 qpp::exception::DimsInvalid, 94  
 qpp::exception::DimsMismatchCvector, 96  
 qpp::exception::DimsMismatchMatrix, 97  
 qpp::exception::DimsMismatchRvector, 99  
 qpp::exception::DimsMismatchVector, 100  
 qpp::exception::DimsNotEqual, 102  
 qpp::exception::Exception, 106  
 qpp::exception::MatrixMismatchSubsys, 130  
 qpp::exception::MatrixNotCvector, 131  
 qpp::exception::MatrixNotRvector, 133  
 qpp::exception::MatrixNotSquare, 134  
 qpp::exception::MatrixNotSquareNorCvector, 136  
 qpp::exception::MatrixNotSquareNorRvector, 137  
 qpp::exception::MatrixNotSquareNorVector, 139  
 qpp::exception::MatrixNotVector, 140  
 qpp::exception::NoCodeword, 142  
 qpp::exception::NotBipartite, 143  
 qpp::exception::NotQubitCvector, 145  
 qpp::exception::NotQubitMatrix, 146  
 qpp::exception::NotQubitRvector, 148  
 qpp::exception::NotQubitSubsys, 149  
 qpp::exception::NotQubitVector, 151  
 qpp::exception::OutOfRange, 152  
 qpp::exception::PermInvalid, 154  
 qpp::exception::PermMismatchDims, 155  
 qpp::exception::SizeMismatch, 160  
 qpp::exception::SubsysMismatchDims, 169  
 qpp::exception::TypeMismatch, 175  
 qpp::exception::UndefinedType, 176  
 qpp::exception::Unknown, 178  
 qpp::exception::ZeroSize, 179  
 types.h, 213  
 uniform  
 qpp, 83  
 var  
 qpp, 83  
 variadic\_vector\_emplace  
 qpp::internal, 88  
 W  
 qpp::States, 167  
 what  
 qpp::exception::Exception, 106  
 what\_  
 qpp::exception::CustomException, 93  
 where\_  
 qpp::exception::Exception, 106  
 X  
 qpp::Gates, 113  
 x0  
 qpp::States, 167  
 x1  
 qpp::States, 167  
 x2contfrac  
 qpp, 83  
 Xd  
 qpp::Gates, 111  
 Y  
 qpp::Gates, 113  
 y0  
 qpp::States, 167  
 y1  
 qpp::States, 167  
 Z  
 qpp::Gates, 113  
 z0  
 qpp::States, 167  
 z1  
 qpp::States, 167  
 Zd  
 qpp::Gates, 112  
 zero  
 qpp::States, 165