

Quantum++  
v1.0.0-devel

Generated by Doxygen 1.8.10

Tue Oct 18 2016 19:22:34



# 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>11</b>
5.1	File List . . . . .	11
<b>6</b>	<b>Namespace Documentation</b>	<b>13</b>
6.1	qpp Namespace Reference . . . . .	13
6.1.1	Detailed Description . . . . .	25
6.1.2	Typedef Documentation . . . . .	25
6.1.2.1	bigint . . . . .	25
6.1.2.2	bra . . . . .	25
6.1.2.3	cmat . . . . .	25
6.1.2.4	cplx . . . . .	25
6.1.2.5	dmat . . . . .	25
6.1.2.6	dyn_col_vect . . . . .	26
6.1.2.7	dyn_mat . . . . .	26
6.1.2.8	dyn_row_vect . . . . .	26
6.1.2.9	idx . . . . .	26
6.1.2.10	ket . . . . .	26
6.1.2.11	to_void . . . . .	26
6.1.3	Function Documentation . . . . .	26
6.1.3.1	absm(const Eigen::MatrixBase< Derived > &A) . . . . .	26
6.1.3.2	abssq(InputIterator first, InputIterator last) . . . . .	27

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> . . . . .	27
6.1.3.4	<code>abssq(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	27
6.1.3.5	<code>adjoint(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	27
6.1.3.6	<code>anticomm(const Eigen::MatrixBase&lt; Derived1 &gt; &amp;A, const Eigen::MatrixBase&lt; Derived2 &gt; &amp;B)</code> . . . . .	28
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> . . . . .	28
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> . . . . .	28
6.1.3.9	<code>apply(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; cmat &gt; &amp;Ks)</code> . . . . .	29
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> . . . . .	29
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> . . . . .	29
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> . . . . .	30
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> . . . . .	30
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> . . . . .	31
6.1.3.15	<code>bloch2rho(const std::vector&lt; double &gt; &amp;r)</code> . . . . .	31
6.1.3.16	<code>choi2kraus(const cmat &amp;A)</code> . . . . .	31
6.1.3.17	<code>choi2super(const cmat &amp;A)</code> . . . . .	32
6.1.3.18	<code>comm(const Eigen::MatrixBase&lt; Derived1 &gt; &amp;A, const Eigen::MatrixBase&lt; Derived2 &gt; &amp;B)</code> . . . . .	32
6.1.3.19	<code>complement(std::vector&lt; T &gt; subsys, idx N)</code> . . . . .	32
6.1.3.20	<code>compperm(const std::vector&lt; idx &gt; &amp;perm, const std::vector&lt; idx &gt; &amp;sigma)</code> . . . . .	33
6.1.3.21	<code>concurrence(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	34
6.1.3.22	<code>conjugate(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	34
6.1.3.23	<code>contfrac2x(const std::vector&lt; int &gt; &amp;cf, idx N)</code> . . . . .	34
6.1.3.24	<code>contfrac2x(const std::vector&lt; int &gt; &amp;cf)</code> . . . . .	35
6.1.3.25	<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> . . . . .	35
6.1.3.26	<code>cosm(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	35
6.1.3.27	<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> . . . . .	35
6.1.3.28	<code>cwise(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, OutputScalar(*)&amp;f( const typename Derived::Scalar &amp;))</code> . . . . .	36
6.1.3.29	<code>det(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	36
6.1.3.30	<code>dirsum(const T &amp;head)</code> . . . . .	36
6.1.3.31	<code>dirsum(const T &amp;head, const Args &amp;...tail)</code> . . . . .	37

6.1.3.32	<code>dirsum(const std::vector&lt; Derived &gt; &amp;As)</code>	37
6.1.3.33	<code>dirsum(const std::initializer_list&lt; Derived &gt; &amp;As)</code>	37
6.1.3.34	<code>dirsumpow(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, idx n)</code>	38
6.1.3.35	<code>disp(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, double chop=qpp::chop)</code>	38
6.1.3.36	<code>disp(cplx z, double chop=qpp::chop)</code>	38
6.1.3.37	<code>disp(InputIterator first, InputIterator last, const std::string &amp;separator, const std::string &amp;start="["", const std::string &amp;end=""]")</code>	38
6.1.3.38	<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>	39
6.1.3.39	<code>disp(const PointerType *p, idx N, const std::string &amp;separator, const std::string &amp;start="["", const std::string &amp;end=""]")</code>	39
6.1.3.40	<code>egcd(bigint a, bigint b)</code>	39
6.1.3.41	<code>eig(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	40
6.1.3.42	<code>entanglement(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;dims)</code>	40
6.1.3.43	<code>entanglement(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, idx d=2)</code>	40
6.1.3.44	<code>entropy(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	41
6.1.3.45	<code>entropy(const std::vector&lt; double &gt; &amp;prob)</code>	41
6.1.3.46	<code>evals(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	41
6.1.3.47	<code>evecs(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	41
6.1.3.48	<code>expm(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	42
6.1.3.49	<code>factors(bigint a)</code>	42
6.1.3.50	<code>funm(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, cplx(*f)(const cplx &amp;))</code>	42
6.1.3.51	<code>gcd(bigint a, bigint b)</code>	42
6.1.3.52	<code>gcd(const std::vector&lt; bigint &gt; &amp;as)</code>	43
6.1.3.53	<code>gconcurrency(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	43
6.1.3.54	<code>grams(const std::vector&lt; Derived &gt; &amp;As)</code>	43
6.1.3.55	<code>grams(const std::initializer_list&lt; Derived &gt; &amp;As)</code>	44
6.1.3.56	<code>grams(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	44
6.1.3.57	<code>heig(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	44
6.1.3.58	<code>hevals(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	45
6.1.3.59	<code>hevecs(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	45
6.1.3.60	<code>inverse(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	45
6.1.3.61	<code>invperm(const std::vector&lt; idx &gt; &amp;perm)</code>	45
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, const std::vector&lt; idx &gt; &amp;dims)</code>	46
6.1.3.63	<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>	47
6.1.3.64	<code>isprime(bigint p, idx k=80)</code>	47
6.1.3.65	<code>kraus2choi(const std::vector&lt; cmat &gt; &amp;Ks)</code>	47
6.1.3.66	<code>kraus2super(const std::vector&lt; cmat &gt; &amp;Ks)</code>	48

6.1.3.67	<code>kron(const T &amp;head)</code>	48
6.1.3.68	<code>kron(const T &amp;head, const Args &amp;...tail)</code>	48
6.1.3.69	<code>kron(const std::vector&lt; Derived &gt; &amp;As)</code>	49
6.1.3.70	<code>kron(const std::initializer_list&lt; Derived &gt; &amp;As)</code>	49
6.1.3.71	<code>kronpow(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, idx n)</code>	49
6.1.3.72	<code>lcm(bigint a, bigint b)</code>	50
6.1.3.73	<code>lcm(const std::vector&lt; bigint &gt; &amp;as)</code>	50
6.1.3.74	<code>load(const std::string &amp;fname)</code>	50
6.1.3.75	<code>loadMATLABmatrix(const std::string &amp;, const std::string &amp;)</code>	51
6.1.3.76	<code>loadMATLABmatrix(const std::string &amp;mat_file, const std::string &amp;var_name)</code>	51
6.1.3.77	<code>loadMATLABmatrix(const std::string &amp;mat_file, const std::string &amp;var_name)</code>	51
6.1.3.78	<code>logdet(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	52
6.1.3.79	<code>logm(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	52
6.1.3.80	<code>lognegativity(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;dims)</code>	52
6.1.3.81	<code>lognegativity(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, idx d=2)</code>	52
6.1.3.82	<code>marginalX(const dmat &amp;probXY)</code>	53
6.1.3.83	<code>marginalY(const dmat &amp;probXY)</code>	53
6.1.3.84	<code>measure(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; cmat &gt; &amp;Ks)</code>	53
6.1.3.85	<code>measure(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::initializer_list&lt; cmat &gt; &amp;Ks)</code>	53
6.1.3.86	<code>measure(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const cmat &amp;U)</code>	54
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, const std::vector&lt; idx &gt; &amp;dims)</code>	54
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, const std::vector&lt; idx &gt; &amp;dims)</code>	54
6.1.3.89	<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>	55
6.1.3.90	<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>	55
6.1.3.91	<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>	56
6.1.3.92	<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>	56
6.1.3.93	<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>	57
6.1.3.94	<code>measure_seq(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, std::vector&lt; idx &gt; &amp;subsys, idx d=2)</code>	57
6.1.3.95	<code>mket(const std::vector&lt; idx &gt; &amp;mask, const std::vector&lt; idx &gt; &amp;dims)</code>	58
6.1.3.96	<code>mket(const std::vector&lt; idx &gt; &amp;mask, idx d=2)</code>	58
6.1.3.97	<code>modinv(bigint a, bigint p)</code>	58
6.1.3.98	<code>modmul(bigint a, bigint b, bigint p)</code>	59

6.1.3.99	<code>modpow(bigint a, bigint n, bigint p)</code>	59
6.1.3.100	<code>mprj(const std::vector&lt; idx &gt; &amp;mask, const std::vector&lt; idx &gt; &amp;dims)</code>	59
6.1.3.101	<code>mprj(const std::vector&lt; idx &gt; &amp;mask, idx d=2)</code>	60
6.1.3.102	<code>multiidx2n(const std::vector&lt; idx &gt; &amp;midx, const std::vector&lt; idx &gt; &amp;dims)</code>	60
6.1.3.103	<code>n2multiidx(idx n, const std::vector&lt; idx &gt; &amp;dims)</code>	60
6.1.3.104	<code>negativity(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;dims)</code>	61
6.1.3.105	<code>negativity(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, idx d=2)</code>	61
6.1.3.106	<code>norm(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	61
6.1.3.107	<code>omega(idx D)</code>	61
6.1.3.108	<code>operator""_i(unsigned long long int x) noexcept</code>	62
6.1.3.109	<code>operator""_i(long double x) noexcept</code>	62
6.1.3.110	<code>powm(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, idx n)</code>	62
6.1.3.111	<code>prj(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	62
6.1.3.112	<code>prod(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	63
6.1.3.113	<code>prod(InputIterator first, InputIterator last)</code>	63
6.1.3.114	<code>prod(const Container &amp;c, typename std::enable_if&lt; is_iterable&lt; Container &gt;::value &gt;::type * = nullptr)</code>	63
6.1.3.115	<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>	63
6.1.3.116	<code>ptrace(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;subsys, idx d=2)</code>	64
6.1.3.117	<code>ptrace1(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;dims)</code>	64
6.1.3.118	<code>ptrace1(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, idx d=2)</code>	65
6.1.3.119	<code>ptrace2(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;dims)</code>	65
6.1.3.120	<code>ptrace2(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, idx d=2)</code>	65
6.1.3.121	<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>	66
6.1.3.122	<code>ptranspose(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;subsys, idx d=2)</code>	66
6.1.3.123	<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>	66
6.1.3.124	<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>	67
6.1.3.125	<code>rand(double a, double b)</code>	67
6.1.3.126	<code>rand(bigint a, bigint b)</code>	67
6.1.3.127	<code>rand(idx rows, idx cols, double a=0, double b=1)</code>	67
6.1.3.128	<code>rand(idx rows, idx cols, double a, double b)</code>	68
6.1.3.129	<code>rand(idx rows, idx cols, double a, double b)</code>	68
6.1.3.130	<code>randH(idx D)</code>	68
6.1.3.131	<code>randidx(idx a=std::numeric_limits&lt; idx &gt;::min(), idx b=std::numeric_limits&lt; idx &gt;::max())</code>	69
6.1.3.132	<code>randket(idx D)</code>	70

6.1.3.133 randkraus(idx N, idx D) . . . . .	70
6.1.3.134 randn(idx rows, idx cols, double mean=0, double sigma=1) . . . . .	70
6.1.3.135 randn(idx rows, idx cols, double mean, double sigma) . . . . .	70
6.1.3.136 randn(idx rows, idx cols, double mean, double sigma) . . . . .	71
6.1.3.137 randn(double mean=0, double sigma=1) . . . . .	71
6.1.3.138 randperm(idx N) . . . . .	71
6.1.3.139 randprime(bigint a, bigint b, idx N=1000) . . . . .	72
6.1.3.140 randrho(idx D) . . . . .	72
6.1.3.141 randU(idx D) . . . . .	72
6.1.3.142 randV(idx Din, idx Dout) . . . . .	72
6.1.3.143 renyi(const Eigen::MatrixBase< Derived > &A, double alpha) . . . . .	73
6.1.3.144 renyi(const std::vector< double > &prob, double alpha) . . . . .	73
6.1.3.145 reshape(const Eigen::MatrixBase< Derived > &A, idx rows, idx cols) . . . . .	73
6.1.3.146 rho2bloch(const Eigen::MatrixBase< Derived > &A) . . . . .	74
6.1.3.147 rho2pure(const Eigen::MatrixBase< Derived > &A) . . . . .	74
6.1.3.148 save(const Eigen::MatrixBase< Derived > &A, const std::string &fname) . . . . .	74
6.1.3.149 saveMATLABmatrix(const Eigen::MatrixBase< Derived > &, const std::string &, const std::string &, const std::string &) . . . . .	75
6.1.3.150 saveMATLABmatrix(const Eigen::MatrixBase< dmat > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode) . . . . .	75
6.1.3.151 saveMATLABmatrix(const Eigen::MatrixBase< cmat > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode) . . . . .	75
6.1.3.152 schatten(const Eigen::MatrixBase< Derived > &A, double p) . . . . .	75
6.1.3.153 schmidtA(const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims) . . . . .	76
6.1.3.154 schmidtA(const Eigen::MatrixBase< Derived > &A, idx d=2) . . . . .	76
6.1.3.155 schmidtB(const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims) . . . . .	76
6.1.3.156 schmidtB(const Eigen::MatrixBase< Derived > &A, idx d=2) . . . . .	76
6.1.3.157 schmidtcoeffs(const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims) . . . . .	77
6.1.3.158 schmidtcoeffs(const Eigen::MatrixBase< Derived > &A, idx d=2) . . . . .	77
6.1.3.159 schmidtprobs(const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims) . . . . .	78
6.1.3.160 schmidtprobs(const Eigen::MatrixBase< Derived > &A, idx d=2) . . . . .	78
6.1.3.161 sigma(const std::vector< double > &prob, const Container &X, typename std::← ::enable_if< is_iterable< Container >::value >::type !=nullptr) . . . . .	78
6.1.3.162 sinm(const Eigen::MatrixBase< Derived > &A) . . . . .	78
6.1.3.163 spectralpowm(const Eigen::MatrixBase< Derived > &A, const cplx z) . . . . .	79
6.1.3.164 sqrtm(const Eigen::MatrixBase< Derived > &A) . . . . .	79
6.1.3.165 sum(const Eigen::MatrixBase< Derived > &A) . . . . .	79
6.1.3.166 sum(InputIterator first, InputIterator last) . . . . .	79



6.1.3.167	<code>sum(const Container &amp;c, typename std::enable_if&lt; is_iterable&lt; Container &gt;::value &gt;::type != nullptr)</code> . . . . .	80
6.1.3.168	<code>super2choi(const cmat &amp;A)</code> . . . . .	80
6.1.3.169	<code>svals(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	80
6.1.3.170	<code>svd(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	80
6.1.3.171	<code>svdU(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	81
6.1.3.172	<code>svdV(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	81
6.1.3.173	<code>syspermute(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;perm, const std::vector&lt; idx &gt; &amp;dims)</code> . . . . .	81
6.1.3.174	<code>syspermute(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, const std::vector&lt; idx &gt; &amp;perm, idx d=2)</code> . . . . .	81
6.1.3.175	<code>trace(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	82
6.1.3.176	<code>transpose(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	82
6.1.3.177	<code>tsallis(const Eigen::MatrixBase&lt; Derived &gt; &amp;A, double q)</code> . . . . .	82
6.1.3.178	<code>tsallis(const std::vector&lt; double &gt; &amp;prob, double q)</code> . . . . .	82
6.1.3.179	<code>uniform(idx N)</code> . . . . .	83
6.1.3.180	<code>var(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> . . . . .	83
6.1.3.181	<code>x2contfrac(double x, idx N, idx cut=1e5)</code> . . . . .	83
6.1.4	Variable Documentation . . . . .	83
6.1.4.1	<code>chop</code> . . . . .	83
6.1.4.2	<code>ee</code> . . . . .	84
6.1.4.3	<code>eps</code> . . . . .	84
6.1.4.4	<code>infty</code> . . . . .	84
6.1.4.5	<code>maxn</code> . . . . .	84
6.1.4.6	<code>pi</code> . . . . .	84
6.2	<code>qpp::experimental</code> Namespace Reference . . . . .	84
6.2.1	Detailed Description . . . . .	84
6.3	<code>qpp::internal</code> Namespace Reference . . . . .	84
6.3.1	Detailed Description . . . . .	85
6.3.2	Function Documentation . . . . .	86
6.3.2.1	<code>check_cvector(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	86
6.3.2.2	<code>check_dims(const std::vector&lt; idx &gt; &amp;dims)</code> . . . . .	86
6.3.2.3	<code>check_dims_match_cvect(const std::vector&lt; idx &gt; &amp;dims, const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	86
6.3.2.4	<code>check_dims_match_mat(const std::vector&lt; idx &gt; &amp;dims, const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	86
6.3.2.5	<code>check_dims_match_rvect(const std::vector&lt; idx &gt; &amp;dims, const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code> . . . . .	86
6.3.2.6	<code>check_eq_dims(const std::vector&lt; idx &gt; &amp;dims, idx dim) noexcept</code> . . . . .	86
6.3.2.7	<code>check_matching_sizes(const T1 &amp;lhs, const T2 &amp;rhs) noexcept</code> . . . . .	86
6.3.2.8	<code>check_nonzero_size(const T &amp;x) noexcept</code> . . . . .	86

6.3.2.9	<code>check_perm(const std::vector&lt; idx &gt; &amp;perm)</code>	86
6.3.2.10	<code>check_qubit_cvector(const Eigen::MatrixBase&lt; Derived &gt; &amp;A) noexcept</code>	86
6.3.2.11	<code>check_qubit_matrix(const Eigen::MatrixBase&lt; Derived &gt; &amp;A) noexcept</code>	86
6.3.2.12	<code>check_qubit_rvector(const Eigen::MatrixBase&lt; Derived &gt; &amp;A) noexcept</code>	86
6.3.2.13	<code>check_qubit_vector(const Eigen::MatrixBase&lt; Derived &gt; &amp;A) noexcept</code>	86
6.3.2.14	<code>check_rvector(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	86
6.3.2.15	<code>check_square_mat(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	86
6.3.2.16	<code>check_subsys_match_dims(const std::vector&lt; idx &gt; &amp;subsys, const std::vector&lt; idx &gt; &amp;dims)</code>	86
6.3.2.17	<code>check_vector(const Eigen::MatrixBase&lt; Derived &gt; &amp;A)</code>	86
6.3.2.18	<code>dirsum2(const Eigen::MatrixBase&lt; Derived1 &gt; &amp;A, const Eigen::MatrixBase&lt; Derived2 &gt; &amp;B)</code>	86
6.3.2.19	<code>get_dim_subsys(idx sz, idx N)</code>	86
6.3.2.20	<code>get_num_subsys(idx sz, idx d)</code>	86
6.3.2.21	<code>kron2(const Eigen::MatrixBase&lt; Derived1 &gt; &amp;A, const Eigen::MatrixBase&lt; Derived2 &gt; &amp;B)</code>	86
6.3.2.22	<code>multiidx2n(const idx *const midx, idx numdims, const idx *const dims) noexcept</code>	87
6.3.2.23	<code>n2multiidx(idx n, idx numdims, const idx *const dims, idx *result) noexcept</code>	87
6.3.2.24	<code>variadic_vector_emplace(std::vector&lt; T &gt; &amp;)</code>	87
6.3.2.25	<code>variadic_vector_emplace(std::vector&lt; T &gt; &amp;v, First &amp;&amp;first, Args &amp;&amp;...args)</code>	87
<b>7</b>	<b>Class Documentation</b>	<b>89</b>
7.1	<code>qpp::Codes</code> Class Reference	89
7.1.1	Detailed Description	90
7.1.2	Member Enumeration Documentation	90
7.1.2.1	Type	90
7.1.3	Constructor & Destructor Documentation	90
7.1.3.1	<code>Codes()</code>	90
7.1.3.2	<code>~Codes()=default</code>	90
7.1.4	Member Function Documentation	91
7.1.4.1	<code>codeword(Type type, idx i) const</code>	91
7.1.5	Friends And Related Function Documentation	91
7.1.5.1	<code>internal::Singleton&lt; const Codes &gt;</code>	91
7.2	<code>qpp::internal::Display_Impl_</code> Struct Reference	91
7.2.1	Member Function Documentation	92
7.2.1.1	<code>display_impl_(const T &amp;A, std::ostream &amp;os, double chop=qpp::chop) const</code>	92
7.3	<code>qpp::Exception</code> Class Reference	92
7.3.1	Detailed Description	93
7.3.2	Member Enumeration Documentation	93
7.3.2.1	Type	93
7.3.3	Constructor & Destructor Documentation	94

7.3.3.1	Exception(const std::string &where, const Type &type)	94
7.3.3.2	Exception(const std::string &where, const std::string &custom)	95
7.3.4	Member Function Documentation	95
7.3.4.1	construct_exception_msg_()	95
7.3.4.2	what() const noexcept override	95
7.3.5	Member Data Documentation	95
7.3.5.1	custom_	95
7.3.5.2	msg_	95
7.3.5.3	type_	95
7.3.5.4	where_	95
7.4	qpp::Gates Class Reference	95
7.4.1	Detailed Description	97
7.4.2	Constructor & Destructor Documentation	98
7.4.2.1	Gates()	98
7.4.2.2	~Gates()=default	98
7.4.3	Member Function Documentation	98
7.4.3.1	CTRL(const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &ctrl, const std::vector< idx > &subsys, idx N, idx d=2) const	98
7.4.3.2	expandout(const Eigen::MatrixBase< Derived > &A, idx pos, const std::vector< idx > &dims) const	98
7.4.3.3	Fd(idx D) const	99
7.4.3.4	Id(idx D) const	99
7.4.3.5	Rn(double theta, const std::vector< double > &n) const	99
7.4.3.6	Xd(idx D) const	100
7.4.3.7	Zd(idx D) const	100
7.4.4	Friends And Related Function Documentation	100
7.4.4.1	internal::Singleton< const Gates >	100
7.4.5	Member Data Documentation	100
7.4.5.1	CNOT	100
7.4.5.2	CNOTba	100
7.4.5.3	CZ	100
7.4.5.4	FRED	100
7.4.5.5	H	101
7.4.5.6	Id2	101
7.4.5.7	S	101
7.4.5.8	SWAP	101
7.4.5.9	T	101
7.4.5.10	TOF	101
7.4.5.11	X	101
7.4.5.12	Y	101

7.4.5.13	Z	101
7.5	qpp::IDisplay Class Reference	101
7.5.1	Detailed Description	103
7.5.2	Constructor & Destructor Documentation	103
7.5.2.1	IDisplay()=default	103
7.5.2.2	IDisplay(const IDisplay &)=default	103
7.5.2.3	IDisplay(IDisplay &&)=default	103
7.5.2.4	~IDisplay()=default	103
7.5.3	Member Function Documentation	103
7.5.3.1	display(std::ostream &os) const =0	103
7.5.3.2	operator=(const IDisplay &)=default	103
7.5.3.3	operator=(IDisplay &&)=default	103
7.5.4	Friends And Related Function Documentation	103
7.5.4.1	operator<<	103
7.6	qpp::Init Class Reference	104
7.6.1	Detailed Description	105
7.6.2	Constructor & Destructor Documentation	105
7.6.2.1	Init()	105
7.6.2.2	~Init()	105
7.6.3	Friends And Related Function Documentation	105
7.6.3.1	internal::Singleton< const Init >	105
7.7	qpp::internal::IOManipEigen Class Reference	105
7.7.1	Constructor & Destructor Documentation	106
7.7.1.1	IOManipEigen(const Eigen::MatrixBase< Derived > &A, double chop=qpp::chop)	106
7.7.1.2	IOManipEigen(const cplx z, double chop=qpp::chop)	106
7.7.2	Member Function Documentation	106
7.7.2.1	display(std::ostream &os) const override	106
7.7.3	Member Data Documentation	107
7.7.3.1	A_	107
7.7.3.2	chop_	107
7.8	qpp::internal::IOManipPointer< PointerType > Class Template Reference	107
7.8.1	Constructor & Destructor Documentation	108
7.8.1.1	IOManipPointer(const PointerType *p, idx N, const std::string &separator, const std::string &start="", const std::string &end=""]")	108
7.8.1.2	IOManipPointer(const IOManipPointer &)=default	108
7.8.2	Member Function Documentation	108
7.8.2.1	display(std::ostream &os) const override	108
7.8.2.2	operator=(const IOManipPointer &)=default	108
7.8.3	Member Data Documentation	108
7.8.3.1	end_	108

7.8.3.2	<code>N_</code>	108
7.8.3.3	<code>p_</code>	108
7.8.3.4	<code>separator_</code>	109
7.8.3.5	<code>start_</code>	109
7.9	<code>qpp::internal::IOManipRange&lt; InputIterator &gt;</code> Class Template Reference	109
7.9.1	Constructor & Destructor Documentation	110
7.9.1.1	<code>IOManipRange(InputIterator first, InputIterator last, const std::string &amp;separator, const std::string &amp;start="", const std::string &amp;end="")</code>	110
7.9.1.2	<code>IOManipRange(const IOManipRange &amp;)=default</code>	110
7.9.2	Member Function Documentation	110
7.9.2.1	<code>display(std::ostream &amp;os) const</code> override	110
7.9.2.2	<code>operator=(const IOManipRange &amp;)=default</code>	110
7.9.3	Member Data Documentation	110
7.9.3.1	<code>end_</code>	110
7.9.3.2	<code>first_</code>	110
7.9.3.3	<code>last_</code>	110
7.9.3.4	<code>separator_</code>	111
7.9.3.5	<code>start_</code>	111
7.10	<code>qpp::is_complex&lt; T &gt;</code> Struct Template Reference	111
7.10.1	Detailed Description	111
7.11	<code>qpp::is_complex&lt; std::complex&lt; T &gt; &gt;</code> Struct Template Reference	112
7.11.1	Detailed Description	112
7.12	<code>qpp::is_iterable&lt; T, typename &gt;</code> Struct Template Reference	113
7.12.1	Detailed Description	113
7.13	<code>qpp::is_iterable&lt; T, to_void&lt; decltype(std::declval&lt; T &gt;().begin()), decltype(std::declval&lt; T &gt;().end()), typename T::value_type &gt; &gt;</code> Struct Template Reference	114
7.13.1	Detailed Description	115
7.14	<code>qpp::is_matrix_expression&lt; Derived &gt;</code> Struct Template Reference	115
7.14.1	Detailed Description	116
7.15	<code>qpp::is_matrix_expression&lt; typename Eigen::MatrixBase&lt; Derived &gt; &gt;</code> Struct Template Reference	116
7.15.1	Detailed Description	117
7.16	<code>qpp::RandomDevices</code> Class Reference	117
7.16.1	Detailed Description	119
7.16.2	Constructor & Destructor Documentation	119
7.16.2.1	<code>RandomDevices()</code>	119
7.16.2.2	<code>~RandomDevices()=default</code>	119
7.16.3	Friends And Related Function Documentation	119
7.16.3.1	<code>internal::Singleton&lt; RandomDevices &gt;</code>	119
7.16.4	Member Data Documentation	119
7.16.4.1	<code>rd_</code>	119

7.16.4.2	rng_ . . . . .	119
7.17	qpp::internal::Singleton< T > Class Template Reference . . . . .	119
7.17.1	Detailed Description . . . . .	120
7.17.2	Constructor & Destructor Documentation . . . . .	120
7.17.2.1	Singleton() noexcept=default . . . . .	120
7.17.2.2	Singleton(const Singleton &)=delete . . . . .	120
7.17.2.3	~Singleton()=default . . . . .	120
7.17.3	Member Function Documentation . . . . .	121
7.17.3.1	get_instance() noexcept(std::is_nothrow_constructible< T >::value) . . . . .	121
7.17.3.2	get_thread_local_instance() noexcept(std::is_nothrow_constructible< T >::value) . . . . .	121
7.17.3.3	operator=(const Singleton &)=delete . . . . .	121
7.18	qpp::States Class Reference . . . . .	121
7.18.1	Detailed Description . . . . .	123
7.18.2	Constructor & Destructor Documentation . . . . .	123
7.18.2.1	States() . . . . .	123
7.18.2.2	~States()=default . . . . .	123
7.18.3	Member Function Documentation . . . . .	123
7.18.3.1	mes(idx d=2) const . . . . .	123
7.18.4	Friends And Related Function Documentation . . . . .	123
7.18.4.1	internal::Singleton< const States > . . . . .	123
7.18.5	Member Data Documentation . . . . .	123
7.18.5.1	b00 . . . . .	124
7.18.5.2	b01 . . . . .	124
7.18.5.3	b10 . . . . .	124
7.18.5.4	b11 . . . . .	124
7.18.5.5	GHZ . . . . .	124
7.18.5.6	pb00 . . . . .	124
7.18.5.7	pb01 . . . . .	124
7.18.5.8	pb10 . . . . .	124
7.18.5.9	pb11 . . . . .	124
7.18.5.10	pGHZ . . . . .	124
7.18.5.11	pW . . . . .	124
7.18.5.12	px0 . . . . .	124
7.18.5.13	px1 . . . . .	125
7.18.5.14	py0 . . . . .	125
7.18.5.15	py1 . . . . .	125
7.18.5.16	pz0 . . . . .	125
7.18.5.17	pz1 . . . . .	125
7.18.5.18	W . . . . .	125
7.18.5.19	x0 . . . . .	125

7.18.5.20 x1 . . . . .	125
7.18.5.21 y0 . . . . .	125
7.18.5.22 y1 . . . . .	125
7.18.5.23 z0 . . . . .	125
7.18.5.24 z1 . . . . .	125
7.19 qpp::Timer< T, CLOCK_T > Class Template Reference . . . . .	126
7.19.1 Detailed Description . . . . .	127
7.19.2 Constructor & Destructor Documentation . . . . .	127
7.19.2.1 Timer() noexcept . . . . .	127
7.19.2.2 Timer(const Timer &)=default . . . . .	127
7.19.2.3 Timer(Timer &&)=default . . . . .	127
7.19.2.4 ~Timer()=default . . . . .	127
7.19.3 Member Function Documentation . . . . .	128
7.19.3.1 display(std::ostream &os) const override . . . . .	128
7.19.3.2 get_duration() const noexcept . . . . .	128
7.19.3.3 operator=(const Timer &)=default . . . . .	128
7.19.3.4 operator=(Timer &&)=default . . . . .	128
7.19.3.5 tic() noexcept . . . . .	128
7.19.3.6 tics() const noexcept . . . . .	128
7.19.3.7 toc() noexcept . . . . .	129
7.19.4 Member Data Documentation . . . . .	129
7.19.4.1 end_ . . . . .	129
7.19.4.2 start_ . . . . .	129
<b>8 File Documentation</b> . . . . .	<b>131</b>
8.1 classes/codes.h File Reference . . . . .	131
8.1.1 Detailed Description . . . . .	131
8.2 classes/exception.h File Reference . . . . .	131
8.2.1 Detailed Description . . . . .	132
8.3 classes/gates.h File Reference . . . . .	132
8.3.1 Detailed Description . . . . .	133
8.4 classes/ideisplay.h File Reference . . . . .	133
8.4.1 Detailed Description . . . . .	133
8.5 classes/init.h File Reference . . . . .	134
8.5.1 Detailed Description . . . . .	134
8.6 classes/random_devices.h File Reference . . . . .	134
8.6.1 Detailed Description . . . . .	135
8.7 classes/states.h File Reference . . . . .	135
8.7.1 Detailed Description . . . . .	135
8.8 classes/timer.h File Reference . . . . .	136

8.8.1 Detailed Description . . . . .	136
8.9 constants.h File Reference . . . . .	136
8.9.1 Detailed Description . . . . .	137
8.10 entanglement.h File Reference . . . . .	137
8.10.1 Detailed Description . . . . .	139
8.11 entropies.h File Reference . . . . .	139
8.11.1 Detailed Description . . . . .	140
8.12 experimental/experimental.h File Reference . . . . .	140
8.12.1 Detailed Description . . . . .	140
8.13 functions.h File Reference . . . . .	140
8.13.1 Detailed Description . . . . .	144
8.14 input_output.h File Reference . . . . .	145
8.14.1 Detailed Description . . . . .	146
8.15 instruments.h File Reference . . . . .	146
8.15.1 Detailed Description . . . . .	147
8.16 internal/classes/iomanip.h File Reference . . . . .	147
8.16.1 Detailed Description . . . . .	148
8.17 internal/classes/singleton.h File Reference . . . . .	148
8.17.1 Detailed Description . . . . .	149
8.18 internal/util.h File Reference . . . . .	149
8.18.1 Detailed Description . . . . .	150
8.19 macros.h File Reference . . . . .	151
8.19.1 Detailed Description . . . . .	151
8.19.2 Macro Definition Documentation . . . . .	151
8.19.2.1 ERROR . . . . .	151
8.19.2.2 ERRORLN . . . . .	151
8.19.2.3 PRINT . . . . .	151
8.19.2.4 PRINTLN . . . . .	151
8.20 MATLAB/matlab.h File Reference . . . . .	152
8.20.1 Detailed Description . . . . .	152
8.21 number_theory.h File Reference . . . . .	152
8.21.1 Detailed Description . . . . .	154
8.22 operations.h File Reference . . . . .	154
8.22.1 Detailed Description . . . . .	156
8.23 qpp.h File Reference . . . . .	156
8.23.1 Detailed Description . . . . .	158
8.23.2 Macro Definition Documentation . . . . .	158
8.23.2.1 QPP_UNUSED_ . . . . .	158
8.24 random.h File Reference . . . . .	158
8.24.1 Detailed Description . . . . .	159



---

8.25 statistics.h File Reference . . . . .	159
8.25.1 Detailed Description . . . . .	160
8.26 traits.h File Reference . . . . .	161
8.26.1 Detailed Description . . . . .	161
8.27 types.h File Reference . . . . .	162
8.27.1 Detailed Description . . . . .	163
<b>Index</b>	<b>165</b>



# Chapter 1

## Quantum++

### Version 1.0.0-devel - development

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.

If you are interesting in contributing, please contact me. To contribute, you need to have a good 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

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 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`.
- If your compiler does not support [OpenMP](#) (as it is the case e.g with [clang++](#) pre version 3.7), disable [OpenMP](#) in your build, as otherwise the linker may not find the [gomp](#) library.
- If you run the program on [OS X](#) 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_OSX_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](#) 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 . . . . .	13
<a href="#">qpp::experimental</a>	Experimental/test functions/classes, do not use or modify . . . . .	84
<a href="#">qpp::internal</a>	Internal utility functions, do not use/modify . . . . .	84





## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

qpp::internal::Display_Impl_ . . . . .	91
qpp::internal::IOManipEigen . . . . .	105
std::exception	
qpp::Exception . . . . .	92
false_type	
qpp::is_complex< T > . . . . .	111
qpp::is_iterable< T, typename > . . . . .	113
qpp::is_matrix_expression< Derived > . . . . .	115
qpp::IDisplay . . . . .	101
qpp::internal::IOManipEigen . . . . .	105
qpp::internal::IOManipPointer< PointerType > . . . . .	107
qpp::internal::IOManipRange< InputIterator > . . . . .	109
qpp::Timer< T, CLOCK_T > . . . . .	126
qpp::internal::Singleton< T > . . . . .	119
qpp::internal::Singleton< const Codes > . . . . .	119
qpp::Codes . . . . .	89
qpp::internal::Singleton< const Gates > . . . . .	119
qpp::Gates . . . . .	95
qpp::internal::Singleton< const Init > . . . . .	119
qpp::Init . . . . .	104
qpp::internal::Singleton< const States > . . . . .	119
qpp::States . . . . .	121
qpp::internal::Singleton< RandomDevices > . . . . .	119
qpp::RandomDevices . . . . .	117
true_type	
qpp::is_complex< std::complex< T > > . . . . .	112
qpp::is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), typename T::value_type > > . . . . .	114
qpp::is_matrix_expression< typename Eigen::MatrixBase< Derived > > . . . . .	116



## 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::internal::Display_Impl_</a>	. . . . .	91
<a href="#">qpp::Exception</a>	Generates custom exceptions, used when validating function parameters . . . . .	92
<a href="#">qpp::Gates</a>	Const Singleton class that implements most commonly used gates . . . . .	95
<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> . . . . .	101
<a href="#">qpp::Init</a>	Const Singleton class that performs additional initializations/cleanups . . . . .	104
<a href="#">qpp::internal::IOManipEigen</a>	. . . . .	105
<a href="#">qpp::internal::IOManipPointer&lt; PointerType &gt;</a>	. . . . .	107
<a href="#">qpp::internal::IOManipRange&lt; InputIterator &gt;</a>	. . . . .	109
<a href="#">qpp::is_complex&lt; T &gt;</a>	Checks whether the type is a complex type . . . . .	111
<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 . . . . .	112
<a href="#">qpp::is_iterable&lt; T, typename &gt;</a>	Checks whether <i>T</i> is compatible with an STL-like iterable container . . . . .	113
<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 . . . . .	114
<a href="#">qpp::is_matrix_expression&lt; Derived &gt;</a>	Checks whether the type is an Eigen matrix expression . . . . .	115
<a href="#">qpp::is_matrix_expression&lt; typename Eigen::MatrixBase&lt; Derived &gt; &gt;</a>	Checks whether the type is an Eigen matrix expression, specialization for Eigen matrix expres- sions . . . . .	116
<a href="#">qpp::RandomDevices</a>	Singleton class that manages the source of randomness in the library . . . . .	117
<a href="#">qpp::internal::Singleton&lt; T &gt;</a>	<a href="#">Singleton</a> policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern) . . . . .	119
<a href="#">qpp::States</a>	Const Singleton class that implements most commonly used states . . . . .	121

[qpp::Timer< T, CLOCK\\_T >](#)  
    [Chronometer](#) . . . . . [126](#)

## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">constants.h</a>	
Constants . . . . .	136
<a href="#">entanglement.h</a>	
Entanglement functions . . . . .	137
<a href="#">entropies.h</a>	
Entropy functions . . . . .	139
<a href="#">functions.h</a>	
Generic quantum computing functions . . . . .	140
<a href="#">input_output.h</a>	
Input/output functions . . . . .	145
<a href="#">instruments.h</a>	
Measurement functions . . . . .	146
<a href="#">macros.h</a>	
Preprocessor macros . . . . .	151
<a href="#">number_theory.h</a>	
Number theory functions . . . . .	152
<a href="#">operations.h</a>	
Quantum operation functions . . . . .	154
<a href="#">qpp.h</a>	
Quantum++ main header file, includes all other necessary headers . . . . .	156
<a href="#">random.h</a>	
Randomness-related functions . . . . .	158
<a href="#">statistics.h</a>	
Statistics functions . . . . .	159
<a href="#">traits.h</a>	
Type traits . . . . .	161
<a href="#">types.h</a>	
Type aliases . . . . .	162
classes/ <a href="#">codes.h</a>	
Quantum error correcting codes . . . . .	131
classes/ <a href="#">exception.h</a>	
Exceptions . . . . .	131
classes/ <a href="#">gates.h</a>	
Quantum gates . . . . .	132
classes/ <a href="#">display.h</a>	
Display interface via the non-virtual interface (NVI) . . . . .	133
classes/ <a href="#">init.h</a>	
Initialization . . . . .	134

classes/ <a href="#">random_devices.h</a>	
Random devices . . . . .	134
classes/ <a href="#">states.h</a>	
Quantum states . . . . .	135
classes/ <a href="#">timer.h</a>	
Timing . . . . .	136
experimental/ <a href="#">experimental.h</a>	
Experimental/test functions/classes . . . . .	140
internal/ <a href="#">util.h</a>	
Internal utility functions . . . . .	149
internal/classes/ <a href="#">iomanip.h</a>	
Input/output manipulators . . . . .	147
internal/classes/ <a href="#">singleton.h</a>	
Singleton pattern via CRTP . . . . .	148
MATLAB/ <a href="#">matlab.h</a>	
Input/output interfacing with MATLAB . . . . .	152

## Chapter 6

# Namespace Documentation

### 6.1 qpp Namespace Reference

Quantum++ main namespace.

#### Namespaces

- [experimental](#)  
*Experimental/test functions/classes, do not use or modify.*
- [internal](#)  
*Internal utility functions, do not use/modify.*

#### Classes

- class [Codes](#)  
*const Singleton class that defines quantum error correcting codes*
- class [Exception](#)  
*Generates custom exceptions, used when validating function parameters.*
- class [Gates](#)  
*const Singleton class that implements most commonly used gates*
- class [IDisplay](#)  
*Abstract class (interface) that mandates the definition of virtual `std::ostream& display(std::ostream& os) const`.*
- class [Init](#)  
*const Singleton class that performs additional initializations/cleanups*
- struct [is\\_complex](#)  
*Checks whether the type is a complex type.*
- struct [is\\_complex< std::complex< T > >](#)  
*Checks whether the type is a complex number type, specialization for complex types.*
- struct [is\\_iterable](#)  
*Checks whether T is compatible with an STL-like iterable container.*
- struct [is\\_iterable< T, to\\_void< decltype\(std::declval< T >\(\).begin\(\)\), decltype\(std::declval< T >\(\).end\(\)\), typename T::value\\_type > >](#)  
*Checks whether T is compatible with an STL-like iterable container, specialization for STL-like iterable containers.*
- struct [is\\_matrix\\_expression](#)  
*Checks whether the type is an Eigen matrix expression.*
- struct [is\\_matrix\\_expression< typename Eigen::MatrixBase< Derived > >](#)  
*Checks whether the type is an Eigen matrix expression, specialization for Eigen matrix expressions.*

- 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 > &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 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.*
- `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 >`  
`Derived loadMATLABmatrix` (const std::string &, const std::string &)  
*Loads an Eigen dynamic matrix from a MATLAB .mat file, generic version.*
- `template<>`  
`dmat loadMATLABmatrix` (const std::string &mat\_file, const std::string &var\_name)  
*Loads an Eigen dynamic matrix from a MATLAB .mat file, specialization for double matrices (qpp::dmat)*
- `template<>`  
`cmat loadMATLABmatrix` (const std::string &mat\_file, const std::string &var\_name)  
*Loads an Eigen dynamic matrix from a MATLAB .mat file, specialization for complex matrices (qpp::cmat)*
- `template<typename Derived >`  
`void saveMATLABmatrix` (const Eigen::MatrixBase< Derived > &, const std::string &, const std::string &, const std::string &)  
*Saves an Eigen dynamic matrix to a MATLAB .mat file, generic version.*
- `template<>`  
`void saveMATLABmatrix` (const Eigen::MatrixBase< dmat > &A, const std::string &mat\_file, const std::string &var\_name, const std::string &mode)  
*Saves an Eigen dynamic matrix to a MATLAB .mat file, specialization for double matrices (qpp::dmat)*
- `template<>`  
`void saveMATLABmatrix` (const Eigen::MatrixBase< cmat > &A, const std::string &mat\_file, const std::string &var\_name, const std::string &mode)  
*Saves an Eigen dynamic matrix to a MATLAB .mat file, specialization for complex matrices (qpp::cmat)*
- `std::vector< int > x2contfrac` (double x, idx N, idx cut=1e5)  
*Simple continued fraction expansion.*
- `double contfrac2x` (const std::vector< int > &cf, idx N)  
*Real representation of a simple continued fraction.*
- `double contfrac2x` (const std::vector< int > &cf)  
*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)  
*Generates a random unitary matrix.*
  - `cmat randV` (`idx` Din, `idx` Dout)  
*Generates a random isometry matrix.*
  - `std::vector< cmat > randkraus` (`idx` N, `idx` D)  
*Generates a set of random Kraus operators.*
  - `cmat randH` (`idx` D)  
*Generates a random Hermitian matrix.*
  - `ket randket` (`idx` D)  
*Generates a random normalized ket (pure state vector)*
  - `cmat randrho` (`idx` D)  
*Generates a random density matrix.*
  - `std::vector< idx > randperm` (`idx` N)  
*Generates a random uniformly distributed permutation.*
  - `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>
auto colvect = dyn_col_vect<float>(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>
auto mat = dyn_mat<float>(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>
auto rowvect = dyn_row_vect<float>(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>	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

The 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 ) [inline]`

Real representation of a simple continued fraction.

## See also

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

## Parameters

<i>cf</i>	Integer vector containing the simple continued fraction expansion
<i>N</i>	Number of terms considered in the continued fraction expansion. If <i>N</i> is greater than the size of <i>cf</i> , then all terms in <i>cf</i> are considered.

## Returns

Real representation of the simple continued fraction

6.1.3.24 `double qpp::contfrac2x ( const std::vector< int > & cf ) [inline]`

Real representation of a simple continued fraction.

See also

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

Parameters

<i>cf</i>	Integer vector containing the simple continued fraction expansion
-----------	---

Returns

Real representation of the simple continued fraction

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

Correlation.

Parameters

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

Returns

Correlation of *X* and *Y*

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

Matrix cos.

Parameters

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

Returns

Matrix cosine of *A*

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

Covariance.

Parameters

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

$X$	Random variable values represented by an STL-like container
$Y$	Random variable values represented by an STL-like container

**Returns**

Covariance of  $X$  and  $Y$

**6.1.3.28** `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

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

Determinant.

**Parameters**

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

**Returns**

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

**6.1.3.30** `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*

6.1.3.31 `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

6.1.3.32 `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

6.1.3.33 `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

6.1.3.34 `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

$A$	Eigen expression
$n$	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  $A$

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

Eigen expression ostream manipulator.

Parameters

$A$	Eigen expression
$chop$	Set to zero the elements smaller in absolute value than $chop$

Returns

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

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

Complex number ostream manipulator.

Parameters

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

Returns

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

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

Range ostream manipulator.

Parameters



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

## Returns

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

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

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

## Parameters

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

## Returns

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

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

C-style pointer ostream manipulator.

## Parameters

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

## Returns

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

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

Extended greatest common divisor of two integers.

## See also

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

## Parameters

$a$	Integer
$b$	Integer

## Returns

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

6.1.3.41 `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

$A$	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

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

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

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

## See also

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

## Parameters

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

## Returns

Entanglement, with the logarithm in base 2

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

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

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

## See also

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

## Parameters

$A$	Eigen expression
$d$	Subsystem dimensions

## Returns

Entanglement, with the logarithm in base 2

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

von-Neumann entropy of the density matrix  $A$

## Parameters

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

## Returns

von-Neumann entropy, with the logarithm in base 2

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

Shannon entropy of the probability distribution  $prob$ .

## Parameters

$prob$	Real probability vector
--------	-------------------------

## Returns

Shannon entropy, with the logarithm in base 2

6.1.3.46 `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

6.1.3.47 `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

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

Matrix exponential.

## Parameters

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

## Returns

Matrix exponential of  $A$

**6.1.3.49** `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

**6.1.3.50** `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

$f(A)$

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

Greatest common divisor of two integers.

## See also

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

## Parameters

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

## Returns

Greatest common divisor of *a* and *b*

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

Greatest common divisor of a list of integers.

## See also

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

## Parameters

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

## Returns

Greatest common divisor of all numbers in *as*

**6.1.3.53** `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

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

## Returns

G-concurrence

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

Gram-Schmidt orthogonalization.

## Parameters

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

## Returns

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

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

Gram-Schmidt orthogonalization.

## Parameters

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

## Returns

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

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

Gram-Schmidt orthogonalization.

## Parameters

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

## Returns

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

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

Full eigen decomposition of Hermitian expression.

## See also

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

## Parameters

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

## Returns

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

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

Hermitian eigenvalues.

See also

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

Parameters

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

Returns

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

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

Hermitian eigenvectors.

See also

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

Parameters

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

Returns

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

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

Inverse.

Parameters

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

Returns

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

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

Inverse permutation.

Parameters

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

Returns

Inverse of the permutation *perm*

6.1.3.62 `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, 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

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

**6.1.3.63** `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

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

**6.1.3.64** `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

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

Choi matrix.

## See also

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

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

## Note

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

## Parameters

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

## Returns

Choi matrix

**6.1.3.66** `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

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

## Returns

Superoperator matrix

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

Kronecker product.

See also

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

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

## Parameters

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

## Returns

Its argument *head*

**6.1.3.68** `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

**6.1.3.69** `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

**6.1.3.70** `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 {A1, A2, ... ,Ak}
-----------	---

**Returns**

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

**6.1.3.71** `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  $A$

### 6.1.3.72 `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  $b$

### 6.1.3.73 `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  $as$

### 6.1.3.74 `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"
auto mat = load<cmat>("input.bin");
```

## Parameters

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

6.1.3.75 `template<typename Derived > Derived qpp::loadMATLABmatrix ( const std::string & , const std::string & )`

Loads an Eigen dynamic matrix from a MATLAB .mat file, generic version.

See also

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

This is the generic version that always throws [qpp::Exception::Type::UNDEFINED\\_TYPE](#). It is specialized only for [qpp::dmat](#) and [qpp::cmat](#) (the only matrix types that can be loaded)

6.1.3.76 `template<> dmat qpp::loadMATLABmatrix ( const std::string & mat_file, const std::string & var_name )`  
[inline]

Loads an Eigen dynamic matrix from a MATLAB .mat file, specialization for double matrices ([qpp::dmat](#))

See also

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

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"
auto mat = loadMATLABmatrix<dmat>("input.mat");
```

## Note

If *var\_name* is a complex matrix, only the real part is loaded

## 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 double dynamic matrix ([qpp::dmat](#))

6.1.3.77 `template<> cmatrix qpp::loadMATLABmatrix ( const std::string & mat_file, const std::string & var_name )`  
[inline]

Loads an Eigen dynamic matrix from a MATLAB .mat file, specialization for complex matrices ([qpp::cmatrix](#))

See also

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

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

Example:

```
// loads a previously saved Eigen dynamic complex matrix
// from the MATLAB file "input.mat"
auto mat = loadMATLABmatrix<cmatrix>("input.mat");
```

## 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 complex dynamic matrix ([qpp::cmat](#))

6.1.3.78 `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*

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

Matrix logarithm.

## Parameters

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

## Returns

Matrix logarithm of *A*

6.1.3.80 `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

6.1.3.81 `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

**6.1.3.82** `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  $X$

**6.1.3.83** `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  $Y$

**6.1.3.84** `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

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

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

## Parameters

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

## Returns

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

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

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

## Parameters

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

## Returns

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

6.1.3.87 `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, 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

$A$	Eigen expression
$Ks$	Set of Kraus operators
$subsys$	Subsystem indexes that are measured
$dims$	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

6.1.3.88 `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, 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

```
6.1.3.89 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

```
6.1.3.90 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

6.1.3.91 `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

6.1.3.92 `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

6.1.3.93 `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

6.1.3.94 `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

**6.1.3.95** `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

**6.1.3.96** `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

**6.1.3.97** `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  $a^{-1} \bmod p$

### 6.1.3.98 `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

### 6.1.3.99 `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

$a^n \bmod p$

### 6.1.3.100 `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

**6.1.3.101** `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

**6.1.3.102** `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

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

Non-negative integer index to multi-index.

**See also**

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

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

## Parameters

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

## Returns

Multi-index of the same size as *dims*

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

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

## Parameters

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

## Returns

Negativity

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

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

## Parameters

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

## Returns

Negativity

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

Frobenius norm.

## Parameters

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

## Returns

Frobenius norm of *A*

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

*D*-th root of unity.

## Parameters

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

## Returns

D-th root of unity  $\exp(2\pi i/D)$

**6.1.3.108** `constexpr cplx qpp::operator""_i ( unsigned long long int x ) [inline], [noexcept]`

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

Example:

```
auto z = 4_i; // type of z is std::complex<double>
```

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

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

Example:

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

**6.1.3.110** `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  $A$

**6.1.3.111** `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*

6.1.3.112 `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*

6.1.3.113 `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

6.1.3.114 `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

6.1.3.115 `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  $A$

6.1.3.116 `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  $A$

6.1.3.117 `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  $A$

6.1.3.118 `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  $A$

6.1.3.119 `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  $A$

6.1.3.120 `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  $A$

6.1.3.121 `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  $A$

6.1.3.122 `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  $A$

6.1.3.123 `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

**6.1.3.124** `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

**6.1.3.125** `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)

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

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

**Parameters**

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

**Returns**

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

**6.1.3.127** `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`

### 6.1.3.128 `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)
auto 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

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

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

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

Example:

```
// generates a 3 x 3 random Eigen::MatrixXcd,
// with entries (both real and imaginary) uniformly distributed in [-1,1)
auto 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

### 6.1.3.130 `cmat qpp::randH ( idx D ) [inline]`

Generates a random Hermitian matrix.

#### Parameters

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

#### Returns

Random Hermitian matrix

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

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

## Parameters

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

## Returns

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

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

Generates a random normalized ket (pure state vector)

## Parameters

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

## Returns

Random normalized ket

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

Generates a set of random Kraus operators.

## Note

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

## Parameters

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

## Returns

Set of *N* Kraus operators satisfying the closure condition

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

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

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

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

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

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

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

Example:

```
// generates a 3 x 3 random Eigen::MatrixXd,
// with entries normally distributed in N(0,2)
auto 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

**6.1.3.136** `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)
auto 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

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

**6.1.3.138** `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  $N$

### 6.1.3.139 `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  $[a, b]$

### 6.1.3.140 `cmat qpp::randrho ( idx $D$ ) [inline]`

Generates a random density matrix.

## Parameters

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

## Returns

Random density matrix

### 6.1.3.141 `cmat qpp::randU ( idx $D$ ) [inline]`

Generates a random unitary matrix.

## Parameters

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

## Returns

Random unitary

### 6.1.3.142 `cmat qpp::randV ( idx $D_{in}$ , idx $D_{out}$ ) [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

**6.1.3.143** `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

**6.1.3.144** `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

**6.1.3.145** `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*

6.1.3.146 `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

6.1.3.147 `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*, as a dynamic column vector over the same scalar field as *A*

6.1.3.148 `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

6.1.3.149 `template<typename Derived > void qpp::saveMATLABmatrix ( const Eigen::MatrixBase< Derived > & , const std::string & , const std::string & , const std::string & )`

Saves an Eigen dynamic matrix to a MATLAB .mat file, generic version.

See also

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

This is the generic version that always throws [qpp::Exception::Type::UNDEFINED\\_TYPE](#). It is specialized only for [qpp::dmat](#) and [qpp::cmat](#) (the only matrix types that can be saved)

6.1.3.150 `template<> void qpp::saveMATLABmatrix ( const Eigen::MatrixBase< dmat > & A, const std::string & mat_file, const std::string & var_name, const std::string & mode ) [inline]`

Saves an Eigen dynamic matrix to a MATLAB .mat file, specialization for double matrices ([qpp::dmat](#))

See also

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

## 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

6.1.3.151 `template<> void qpp::saveMATLABmatrix ( const Eigen::MatrixBase< cmat > & A, const std::string & mat_file, const std::string & var_name, const std::string & mode ) [inline]`

Saves an Eigen dynamic matrix to a MATLAB .mat file, specialization for complex matrices ([qpp::cmat](#))

See also

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

## 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

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

Schatten matrix norm.

## Parameters

$A$	Eigen expression
$p$	Real number, greater or equal to 1, use <code>qpp::infty</code> for $p = \infty$

## Returns

Schatten- $p$  matrix norm of  $A$

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

Schmidt basis on Alice side.

## Parameters

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

## Returns

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

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

Schmidt basis on Alice side.

## Parameters

$A$	Eigen expression
$d$	Subsystem dimensions

## Returns

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

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

Schmidt basis on Bob side.

## Parameters

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

## Returns

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

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

Schmidt basis on Bob side.

## Parameters

$A$	Eigen expression
$d$	Subsystem dimensions

## Returns

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

6.1.3.157 `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

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

## Returns

Schmidt coefficients of  $A$ , as a real dynamic column vector

6.1.3.158 `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  $A$ , as a real dynamic column vector

**6.1.3.159** `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  $A$

**6.1.3.160** `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  $A$

**6.1.3.161** `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

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

Returns

Standard deviation of  $X$

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

Matrix sin.



## Parameters

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

## Returns

Matrix sine of  $A$

6.1.3.163 `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  $A^z$

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

Matrix square root.

## Parameters

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

## Returns

Matrix square root of  $A$

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

Element-wise sum of  $A$ .

## Parameters

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

## Returns

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

6.1.3.166 `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

6.1.3.167 `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

6.1.3.168 `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

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

Singular values.

## Parameters

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

## Returns

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

6.1.3.170 `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

6.1.3.171 `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  $A$

6.1.3.172 `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  $A$

6.1.3.173 `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  $A$

6.1.3.174 `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  $A$

6.1.3.175 `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  $A$

6.1.3.176 `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  $A$

6.1.3.177 `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

6.1.3.178 `double qpp::tsallis ( const std::vector< double > & prob, double q ) [inline]`

Tsallis-  $q$  entropy of the probability distribution  $prob$ , for  $q \geq 0$ .

## Note

When  $q \rightarrow 1$  the Tsallis entropy converges to the Shannon entropy, with the logarithm in base  $e$

## Parameters

<i>prob</i>	Real probability vector
<i>q</i>	Non-negative real number

## Returns

Tsallis-  $q$  entropy

6.1.3.179 `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  $N$

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

Variance.

## Parameters

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

## Returns

Variance of  $X$

6.1.3.181 `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
<i>cut</i>	Stop the expansion when the next term is greater than <i>cut</i>

## 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::experimental Namespace Reference

Experimental/test functions/classes, do not use or modify.

### 6.2.1 Detailed Description

Experimental/test functions/classes, do not use or modify.

## 6.3 qpp::internal Namespace Reference

Internal utility functions, do not use/modify.

### 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)

### 6.3.1 Detailed Description

Internal utility functions, do not use/modify.

### 6.3.2 Function Documentation

- 6.3.2.1 `template<typename Derived > bool qpp::internal::check_cvector ( const Eigen::MatrixBase< Derived > & A )`
- 6.3.2.2 `bool qpp::internal::check_dims ( const std::vector< idx > & dims ) [inline]`
- 6.3.2.3 `template<typename Derived > bool qpp::internal::check_dims_match_cvect ( const std::vector< idx > & dims, const Eigen::MatrixBase< Derived > & A )`
- 6.3.2.4 `template<typename Derived > bool qpp::internal::check_dims_match_mat ( const std::vector< idx > & dims, const Eigen::MatrixBase< Derived > & A )`
- 6.3.2.5 `template<typename Derived > bool qpp::internal::check_dims_match_rvect ( const std::vector< idx > & dims, const Eigen::MatrixBase< Derived > & A )`
- 6.3.2.6 `bool qpp::internal::check_eq_dims ( const std::vector< idx > & dims, idx dim ) [inline], [noexcept]`
- 6.3.2.7 `template<typename T1, typename T2 > bool qpp::internal::check_matching_sizes ( const T1 & lhs, const T2 & rhs ) [noexcept]`
- 6.3.2.8 `template<typename T > bool qpp::internal::check_nonzero_size ( const T & x ) [noexcept]`
- 6.3.2.9 `bool qpp::internal::check_perm ( const std::vector< idx > & perm ) [inline]`
- 6.3.2.10 `template<typename Derived > bool qpp::internal::check_qubit_cvector ( const Eigen::MatrixBase< Derived > & A ) [noexcept]`
- 6.3.2.11 `template<typename Derived > bool qpp::internal::check_qubit_matrix ( const Eigen::MatrixBase< Derived > & A ) [noexcept]`
- 6.3.2.12 `template<typename Derived > bool qpp::internal::check_qubit_rvector ( const Eigen::MatrixBase< Derived > & A ) [noexcept]`
- 6.3.2.13 `template<typename Derived > bool qpp::internal::check_qubit_vector ( const Eigen::MatrixBase< Derived > & A ) [noexcept]`
- 6.3.2.14 `template<typename Derived > bool qpp::internal::check_rvector ( const Eigen::MatrixBase< Derived > & A )`
- 6.3.2.15 `template<typename Derived > bool qpp::internal::check_square_mat ( const Eigen::MatrixBase< Derived > & A )`
- 6.3.2.16 `bool qpp::internal::check_subsys_match_dims ( const std::vector< idx > & subsys, const std::vector< idx > & dims ) [inline]`
- 6.3.2.17 `template<typename Derived > bool qpp::internal::check_vector ( const Eigen::MatrixBase< Derived > & A )`
- 6.3.2.18 `template<typename Derived1, typename Derived2 > dyn_mat<typename Derived1::Scalar> qpp::internal::dirsum2 ( const Eigen::MatrixBase< Derived1 > & A, const Eigen::MatrixBase< Derived2 > & B )`
- 6.3.2.19 `idx qpp::internal::get_dim_subsys ( idx sz, idx N ) [inline]`
- 6.3.2.20 `idx qpp::internal::get_num_subsys ( idx sz, idx d ) [inline]`
- 6.3.2.21 `template<typename Derived1, typename Derived2 > dyn_mat<typename Derived1::Scalar> qpp::internal::kron2 ( const Eigen::MatrixBase< Derived1 > & A, const Eigen::MatrixBase< Derived2 > & B )`



6.3.2.22 `idx qpp::internal::multiidx2n ( const idx *const midx, idx numdims, const idx *const dims ) [inline],  
[noexcept]`

6.3.2.23 `void qpp::internal::n2multiidx ( idx n, idx numdims, const idx *const dims, idx * result ) [inline],  
[noexcept]`

6.3.2.24 `template<typename T> void qpp::internal::variadic_vector_emplace ( std::vector< T> & )`

6.3.2.25 `template<typename T, typename First, typename... Args> void qpp::internal::variadic_vector_emplace ( std::vector< T> & v, First && first, Args &&... args )`



## Chapter 7

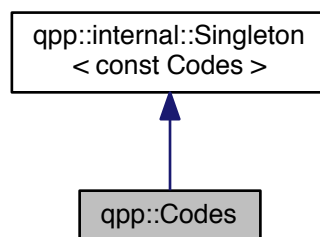
# Class Documentation

### 7.1 qpp::Codes Class Reference

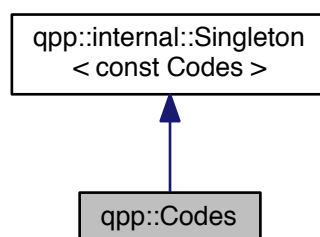
const Singleton class that defines quantum error correcting codes

```
#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]

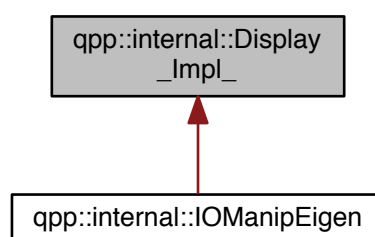
The documentation for this class was generated from the following file:

- classes/[codes.h](#)

## 7.2 qpp::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

### 7.2.1 Member Function Documentation

7.2.1.1 `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](#)

## 7.3 qpp::Exception Class Reference

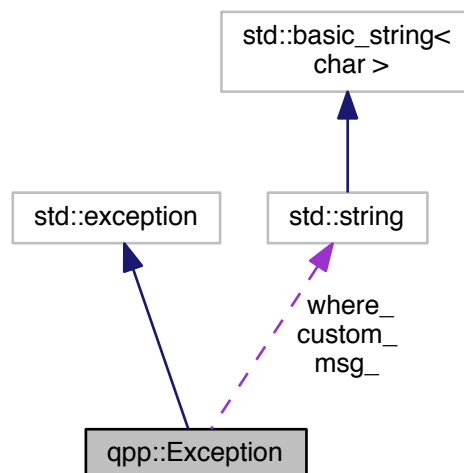
Generates custom exceptions, used when validating function parameters.

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

Inheritance diagram for qpp::Exception:



Collaboration diagram for qpp::Exception:



## Public Types

- enum `Type` {  
`Type::UNKNOWN_EXCEPTION = 1`, `Type::ZERO_SIZE`, `Type::MATRIX_NOT_SQUARE`, `Type::MATRIX_↵  
NOT_CVECTOR`,  
`Type::MATRIX_NOT_RVECTOR`, `Type::MATRIX_NOT_VECTOR`, `Type::MATRIX_NOT_SQUARE_OR_C↵  
VECTOR`, `Type::MATRIX_NOT_SQUARE_OR_RVECTOR`,  
`Type::MATRIX_NOT_SQUARE_OR_VECTOR`, `Type::MATRIX_MISMATCH_SUBSYS`, `Type::DIMS_INVA↵  
LID`, `Type::DIMS_NOT_EQUAL`,  
`Type::DIMS_MISMATCH_MATRIX`, `Type::DIMS_MISMATCH_CVECTOR`, `Type::DIMS_MISMATCH_RVE↵  
CTOR`, `Type::DIMS_MISMATCH_VECTOR`,  
`Type::SUBSYS_MISMATCH_DIMS`, `Type::PERM_INVALID`, `Type::PERM_MISMATCH_DIMS`, `Type::NOT↵  
_QUBIT_MATRIX`,  
`Type::NOT_QUBIT_CVECTOR`, `Type::NOT_QUBIT_RVECTOR`, `Type::NOT_QUBIT_VECTOR`, `Type::NO↵  
T_QUBIT_SUBSYS`,  
`Type::NOT_BIPARTITE`, `Type::NO_CODEWORD`, `Type::OUT_OF_RANGE`, `Type::TYPE_MISMATCH`,  
`Type::SIZE_MISMATCH`, `Type::UNDEFINED_TYPE`, `Type::CUSTOM_EXCEPTION` }

*Exception types, add more here if needed.*

## Public Member Functions

- `Exception` (const std::string &where, const `Type` &type)  
*Constructs an exception.*
- `Exception` (const std::string &where, const std::string &custom)  
*Constructs an exception.*
- virtual const char \* `what` () const noexcept override  
*Overrides std::exception::what()*

## Private Member Functions

- void `construct_exception_msg` ()  
*Constructs the exception description from its type.*

## Private Attributes

- std::string `where_`
- std::string `msg_`
- `Type` `type_`
- std::string `custom_`

### 7.3.1 Detailed Description

Generates custom exceptions, used when validating function parameters.

Customize this class if more exceptions are needed

### 7.3.2 Member Enumeration Documentation

#### 7.3.2.1 enum qpp::Exception::Type [strong]

`Exception` types, add more here if needed.

See also

[qpp::Exception::construct\\_exception\\_msg\\_\(\)](#)

Enumerator

**UNKNOWN\_EXCEPTION** Unknown exception

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

**MATRIX\_NOT\_SQUARE** `Eigen::Matrix` is not square

**MATRIX\_NOT\_CVECTOR** `Eigen::Matrix` is not a column vector

**MATRIX\_NOT\_RVECTOR** `Eigen::Matrix` is not a row vector

**MATRIX\_NOT\_VECTOR** `Eigen::Matrix` is not a row/column vector

**MATRIX\_NOT\_SQUARE\_OR\_CVECTOR** `Eigen::Matrix` is not square nor a column vector

**MATRIX\_NOT\_SQUARE\_OR\_RVECTOR** `Eigen::Matrix` is not square nor a row vector

**MATRIX\_NOT\_SQUARE\_OR\_VECTOR** `Eigen::Matrix` is not square nor a row/column vector

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

**DIMS\_INVALID** `std::vector<idx>` of dimensions has zero size or contains zeros

**DIMS\_NOT\_EQUAL** Local/global dimensions are not equal

**DIMS\_MISMATCH\_MATRIX** Product of the elements of `std::vector<idx>` of dimensions is not equal to the number of rows of `Eigen::Matrix` (assumed to be a square matrix)

**DIMS\_MISMATCH\_CVECTOR** Product of the elements of `std::vector<idx>` of dimensions is not equal to the number of elements of `Eigen::Matrix` (assumed to be a column vector)

**DIMS\_MISMATCH\_RVECTOR** Product of the elements of `std::vector<idx>` of dimensions is not equal to the number of elements of `Eigen::Matrix` (assumed to be a row vector)

**DIMS\_MISMATCH\_VECTOR** Product of the elements of `std::vector<idx>` of dimensions is not equal to the number of elements of `Eigen::Matrix` (assumed to be a row/column vector)

**SUBSYS\_MISMATCH\_DIMS** `std::vector<idx>` of subsystem labels has duplicates, or has entries that are larger than the size of the `std::vector<idx>` of dimensions

**PERM\_INVALID** `std::vector<idx>` does not represent a valid permutation

**PERM\_MISMATCH\_DIMS** Size of the `std::vector<idx>` representing the permutation is different from the size of the `std::vector<idx>` of dimensions

**NOT\_QUBIT\_MATRIX** `Eigen::Matrix` is not 2 x 2

**NOT\_QUBIT\_CVECTOR** `Eigen::Matrix` is not 2 x 1

**NOT\_QUBIT\_RVECTOR** `Eigen::Matrix` is not 1 x 2

**NOT\_QUBIT\_VECTOR** `Eigen::Matrix` is not 1 x 2 nor 2 x 1

**NOT\_QUBIT\_SUBSYS** Subsystems are not 2-dimensional

**NOT\_BIPARTITE** `std::vector<idx>` of dimensions has size different from 2

**NO\_CODEWORD** Codeword does not exist, thrown when calling [qpp::Codes::codeword\(\)](#) with invalid index *i*

**OUT\_OF\_RANGE** Parameter out of range

**TYPE\_MISMATCH** Scalar types do not match

**SIZE\_MISMATCH** Sizes do not match

**UNDEFINED\_TYPE** Templated specialization not defined for this type

**CUSTOM\_EXCEPTION** Custom exception, the user must provide a custom message

### 7.3.3 Constructor & Destructor Documentation

7.3.3.1 `qpp::Exception::Exception ( const std::string & where, const Type & type )` `[inline]`

Constructs an exception.



## Parameters

<i>where</i>	Text representing where the exception occurred
<i>type</i>	<a href="#">Exception</a> type, defined in <a href="#">qpp::Exception::Type</a>

7.3.3.2 `qpp::Exception::Exception ( const std::string & where, const std::string & custom )` `[inline]`

Constructs an exception.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

## Parameters

<i>where</i>	Text representing where the exception occurred
<i>custom</i>	<a href="#">Exception</a> description

## 7.3.4 Member Function Documentation

7.3.4.1 `void qpp::Exception::construct_exception_msg_ ( )` `[inline]`, `[private]`

Constructs the exception description from its type.

See also

[qpp::Exception::Type](#)

Must modify the code of this function if more exceptions are added

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

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

Returns

[Exception](#) description

## 7.3.5 Member Data Documentation

7.3.5.1 `std::string qpp::Exception::custom_` `[private]`7.3.5.2 `std::string qpp::Exception::msg_` `[private]`7.3.5.3 `Type qpp::Exception::type_` `[private]`7.3.5.4 `std::string qpp::Exception::where_` `[private]`

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

- `classes/exception.h`

7.4 `qpp::Gates` Class Reference

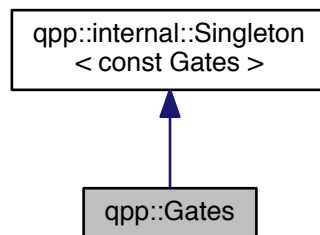
const Singleton class that implements most commonly used gates

```
#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) const  
*Generalized Z gate for qudits.*
- `cmat Fd` (idx D) const  
*Fourier transform gate for qudits.*
- `cmat Xd` (idx D) const  
*Generalized X gate for qudits.*
- `template<typename Derived = Eigen::MatrixXcd>`  
`Derived Id` (idx D) 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.*

## 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

### 7.4.1 Detailed Description

const Singleton class that implements most commonly used gates

## 7.4.2 Constructor & Destructor Documentation

### 7.4.2.1 `qpp::Gates::Gates ( ) [inline], [private]`

Initializes the gates.

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

Default destructor.

## 7.4.3 Member Function Documentation

### 7.4.3.1 `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  $A$

### 7.4.3.2 `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\(I, I, ..., I, A, I, ..., I\)](#).

Parameters

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

<i>pos</i>	Position
<i>dims</i>	Dimensions of the multi-partite system

**Returns**

Tensor product  $I \otimes \dots \otimes I \otimes A \otimes I \otimes \dots \otimes I$ , with  $A$  on position  $pos$ , as a dynamic matrix over the same scalar field as  $A$

**7.4.3.3 cmat qpp::Gates::Fd ( idx  $D$  ) const [inline]**

Fourier transform gate for qudits.

**Note**

Defined as  $F = \sum_{jk} \exp(2\pi i jk/D) |j\rangle\langle k|$

**Parameters**

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

**Returns**

Fourier transform gate for qudits

**7.4.3.4 template<typename Derived = Eigen::MatrixXcd> Derived qpp::Gates::Id ( idx  $D$  ) const [inline]**

Identity gate.

**Note**

Can change the return type from complex matrix (default) by explicitly specifying the template parameter

**Parameters**

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

**Returns**

Identity gate

**7.4.3.5 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**

$\theta$	Rotation angle
$n$	3-dimensional real (unit) vector

**Returns**

Rotation gate

#### 7.4.3.6 `cmat qpp::Gates::Xd ( idx D ) const [inline]`

Generalized X gate for qudits.

##### Note

Defined as  $X = \sum_j |j \oplus 1\rangle\langle j|$ , i.e. raising operator  $X|j\rangle = |j \oplus 1\rangle$

##### Parameters

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

##### Returns

Generalized X gate for qudits

#### 7.4.3.7 `cmat qpp::Gates::Zd ( idx D ) const [inline]`

Generalized Z gate for qudits.

##### Note

Defined as  $Z = \sum_j \exp(2\pi i j / D) |j\rangle\langle j|$

##### Parameters

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

##### Returns

Generalized Z gate for qudits

### 7.4.4 Friends And Related Function Documentation

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

### 7.4.5 Member Data Documentation

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

Controlled-NOT control target gate.

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

Controlled-NOT target control gate.

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

Controlled-Phase gate.

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

Fredkin gate.

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

Hadamard gate.

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

Identity gate.

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

S gate.

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

SWAP gate.

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

T gate.

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

Toffoli gate.

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

Pauli Sigma-X gate.

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

Pauli Sigma-Y gate.

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

Pauli Sigma-Z gate.

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

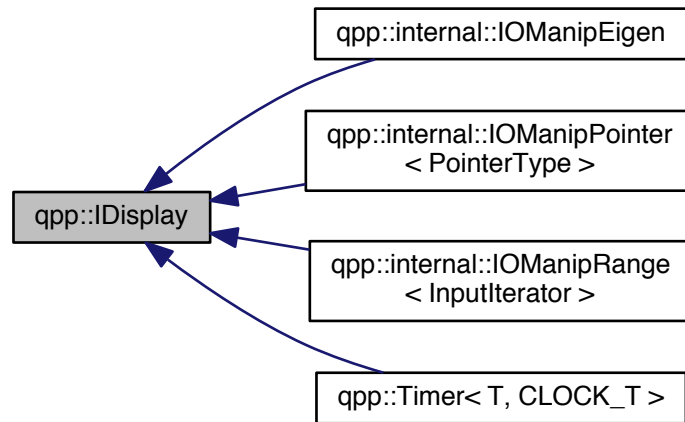
- [classes/gates.h](#)

## 7.5 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.*



### 7.5.1 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.

### 7.5.2 Constructor & Destructor Documentation

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

Default constructor.

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

Default copy constructor.

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

Default move constructor.

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

Default virtual destructor.

### 7.5.3 Member Function Documentation

#### 7.5.3.1 `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 >`.

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

Default copy assignment operator.

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

Default move assignment operator.

### 7.5.4 Friends And Related Function Documentation

#### 7.5.4.1 `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:

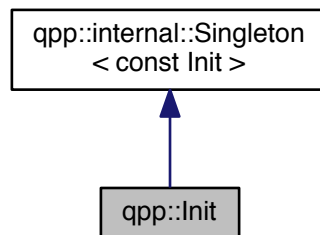
- [classes/ideplay.h](#)

## 7.6 qpp::Init Class Reference

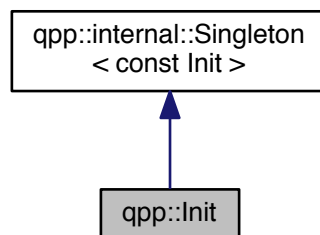
const Singleton class that performs additional initializations/cleanups

```
#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

### 7.6.1 Detailed Description

const Singleton class that performs additional initializations/cleanups

### 7.6.2 Constructor & Destructor Documentation

#### 7.6.2.1 qpp::Init::Init( ) [inline],[private]

Additional initializations.

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

Cleanups.

### 7.6.3 Friends And Related Function Documentation

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

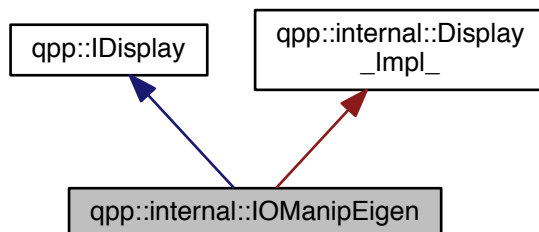
The documentation for this class was generated from the following file:

- [classes/init.h](#)

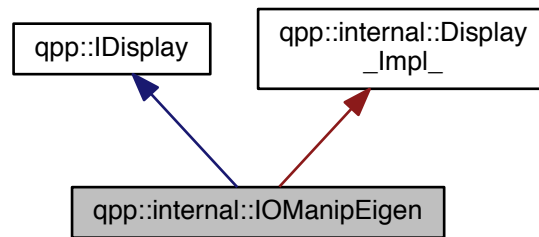
## 7.7 qpp::internal::IOManipEigen Class Reference

```
#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_`

### 7.7.1 Constructor & Destructor Documentation

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

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

### 7.7.2 Member Function Documentation

7.7.2.1 `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`.

### 7.7.3 Member Data Documentation

7.7.3.1 `cmat qpp::internal::IManipEigen::A_` [private]

7.7.3.2 `double qpp::internal::IManipEigen::chop_` [private]

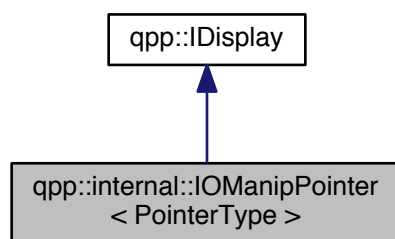
The documentation for this class was generated from the following file:

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

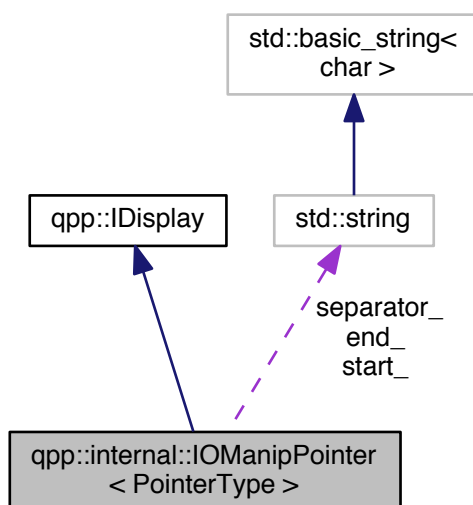
## 7.8 qpp::internal::IManipPointer< PointerType > Class Template Reference

```
#include <internal/classes/iomanip.h>
```

Inheritance diagram for `qpp::internal::IManipPointer< PointerType >`:



Collaboration diagram for `qpp::internal::IManipPointer< 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\\_](#)

## 7.8.1 Constructor & Destructor Documentation

- 7.8.1.1 `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]`
- 7.8.1.2 `template<typename PointerType> qpp::internal::IOManipPointer< PointerType >::IOManipPointer ( const IOManipPointer< PointerType > & ) [default]`

## 7.8.2 Member Function Documentation

- 7.8.2.1 `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](#).

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

## 7.8.3 Member Data Documentation

- 7.8.3.1 `template<typename PointerType> std::string qpp::internal::IOManipPointer< PointerType >::end_ [private]`
- 7.8.3.2 `template<typename PointerType> idx qpp::internal::IOManipPointer< PointerType >::N_ [private]`
- 7.8.3.3 `template<typename PointerType> const PointerType* qpp::internal::IOManipPointer< PointerType >::p_ [private]`

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

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

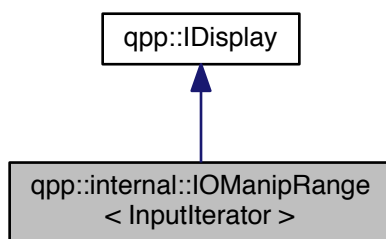
The documentation for this class was generated from the following file:

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

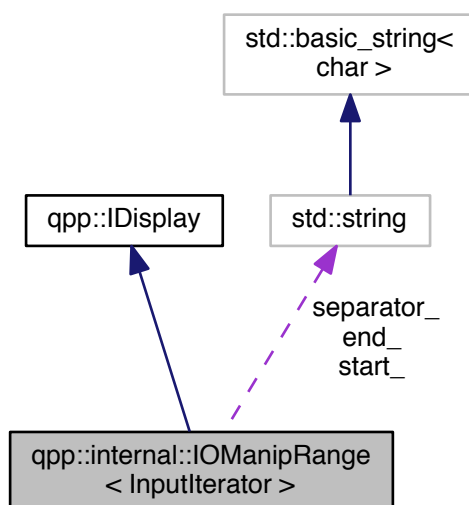
## 7.9 qpp::internal::IOManipRange< InputIterator > Class Template Reference

```
#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\\_](#)

### 7.9.1 Constructor & Destructor Documentation

- 7.9.1.1 `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]`
- 7.9.1.2 `template<typename InputIterator> qpp::internal::IOManipRange< InputIterator >::IOManipRange ( const IOManipRange< InputIterator > & ) [default]`

### 7.9.2 Member Function Documentation

- 7.9.2.1 `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](#).

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

### 7.9.3 Member Data Documentation

- 7.9.3.1 `template<typename InputIterator> std::string qpp::internal::IOManipRange< InputIterator >::end_ [private]`
- 7.9.3.2 `template<typename InputIterator> InputIterator qpp::internal::IOManipRange< InputIterator >::first_ [private]`
- 7.9.3.3 `template<typename InputIterator> InputIterator qpp::internal::IOManipRange< InputIterator >::last_ [private]`



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

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

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

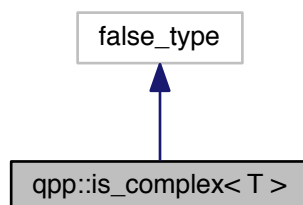
- [internal/classes/iomanip.h](#)

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

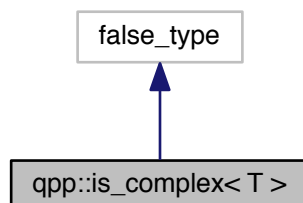
Checks whether the type is a complex type.

`#include <traits.h>`

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



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



### 7.10.1 Detailed Description

`template<typename T> struct qpp::is_complex< T >`

Checks whether the type is a complex type.

Provides the member constant *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:

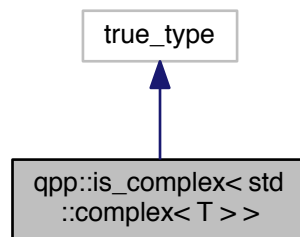
- [traits.h](#)

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

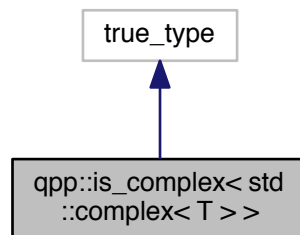
Checks whether the type is a complex number type, specialization for complex types.

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

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



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



### 7.11.1 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:

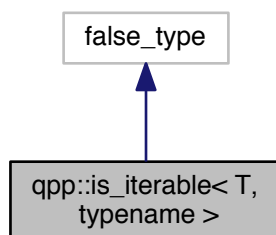
- [traits.h](#)

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

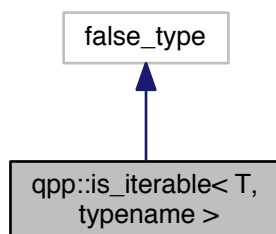
Checks whether *T* is compatible with an STL-like iterable container.

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

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



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



### 7.12.1 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 member constant *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:

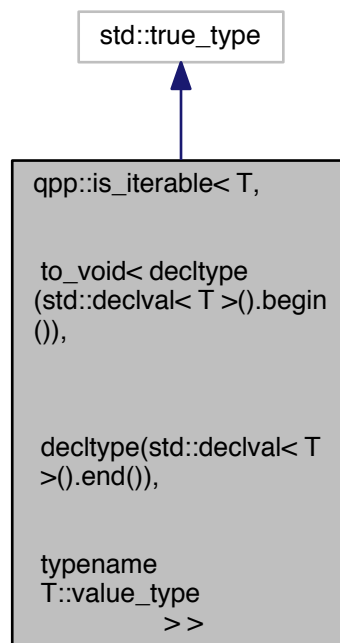
- [traits.h](#)

### 7.13 `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.

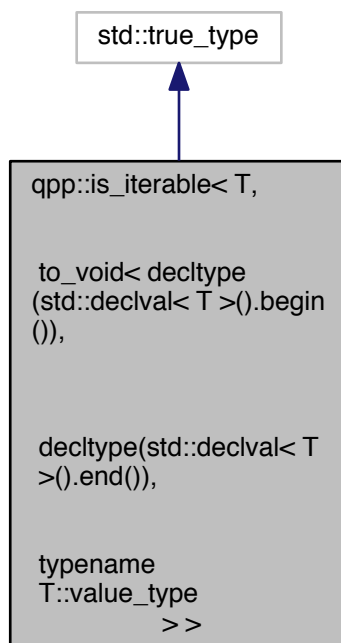
```
#include <traits.h>
```

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 > >:
```



### 7.13.1 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:

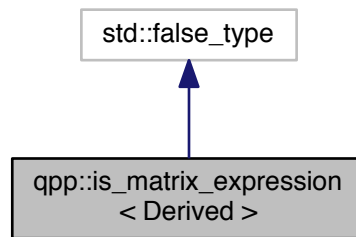
- [traits.h](#)

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

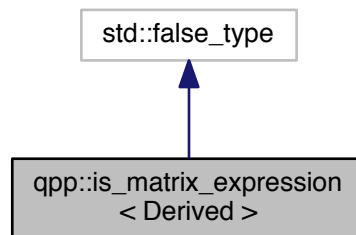
Checks whether the type is an Eigen matrix expression.

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

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



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



### 7.14.1 Detailed Description

```
template<typename Derived> struct qpp::is_matrix_expression< Derived >
```

Checks whether the type is an Eigen matrix expression.

Provides the member constant *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:

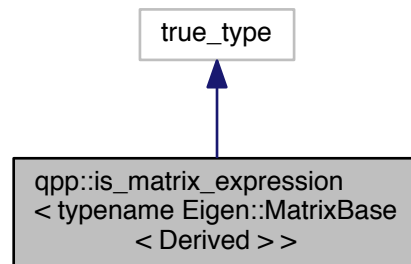
- [traits.h](#)

## 7.15 `qpp::is_matrix_expression< typename Eigen::MatrixBase< Derived > >` Struct Template Reference

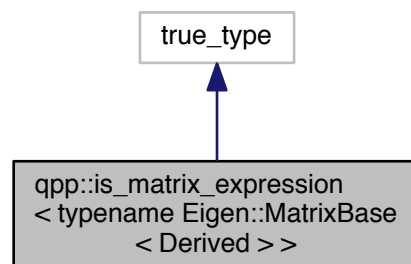
Checks whether the type is an Eigen matrix expression, specialization for Eigen matrix expressions.

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

Inheritance diagram for `qpp::is_matrix_expression< typename Eigen::MatrixBase< Derived > >`:



Collaboration diagram for `qpp::is_matrix_expression< typename Eigen::MatrixBase< Derived > >`:



### 7.15.1 Detailed Description

```
template<typename Derived> struct qpp::is_matrix_expression< typename Eigen::MatrixBase< Derived > >
```

Checks whether the type is an Eigen matrix expression, specialization for Eigen matrix expressions.

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

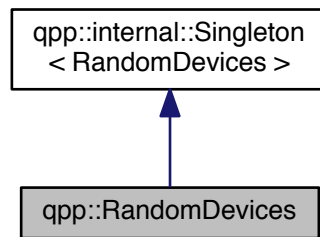
- [traits.h](#)

## 7.16 qpp::RandomDevices Class Reference

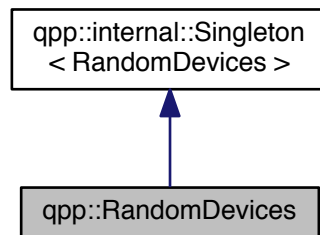
Singleton class that manages the source of randomness in the library.

```
#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

### 7.16.1 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!

### 7.16.2 Constructor & Destructor Documentation

#### 7.16.2.1 `qpp::RandomDevices::RandomDevices ( ) [inline], [private]`

Initializes and seeds the random number generators.

#### 7.16.2.2 `qpp::RandomDevices::~~RandomDevices ( ) [private], [default]`

Default destructor.

### 7.16.3 Friends And Related Function Documentation

#### 7.16.3.1 `friend class internal::Singleton< RandomDevices > [friend]`

### 7.16.4 Member Data Documentation

#### 7.16.4.1 `std::random_device qpp::RandomDevices::rd_ [private]`

used to seed `std::mt19937 rng_`

#### 7.16.4.2 `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`

## 7.17 `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

### 7.17.1 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.

### 7.17.2 Constructor & Destructor Documentation

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

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

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

### 7.17.3 Member Function Documentation

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

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

7.17.3.3 `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:

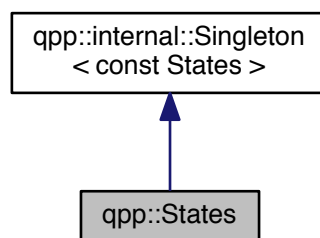
- [internal/classes/singleton.h](#)

## 7.18 qpp::States Class Reference

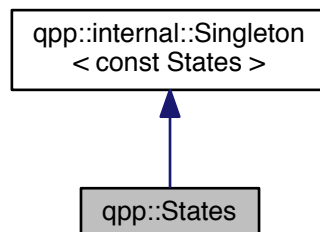
const Singleton class that implements most commonly used states

```
#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.*

## 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

### 7.18.1 Detailed Description

const Singleton class that implements most commonly used states

### 7.18.2 Constructor & Destructor Documentation

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

Initialize the states

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

Default destructor.

### 7.18.3 Member Function Documentation

#### 7.18.3.1 `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

### 7.18.4 Friends And Related Function Documentation

#### 7.18.4.1 `friend class internal::Singleton< const States > [friend]`

### 7.18.5 Member Data Documentation

7.18.5.1 `ket qpp::States::b00 {ket::Zero(4)}`

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

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

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

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

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

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

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

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

GHZ state.

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

Projector onto the Bell-00 state.

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

Projector onto the Bell-01 state.

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

Projector onto the Bell-10 state.

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

Projector onto the Bell-11 state.

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

Projector onto the GHZ state.

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

Projector onto the W state.

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

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

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

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

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

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

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

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

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

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

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

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

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

W state.

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

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

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

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

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

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

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

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

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

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

7.18.5.24 `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:

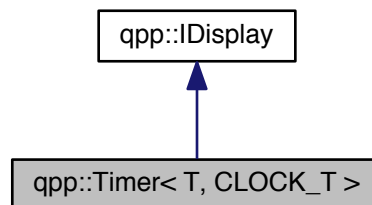
- [classes/states.h](#)

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

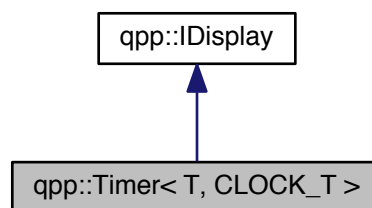
Chronometer.

```
#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*

### 7.19.1 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

### 7.19.2 Constructor & Destructor Documentation

7.19.2.1 `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.

7.19.2.2 `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.

7.19.2.3 `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.

7.19.2.4 `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.

### 7.19.3 Member Function Documentation

7.19.3.1 `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

<code>os</code>	Output stream
-----------------	---------------

Returns

Writes to the output stream the number of tics (specified by T) that passed between the instantiation/reset and invocation of [qpp::Timer::toc\(\)](#).

Implements [qpp::IDisplay](#).

7.19.3.2 `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

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

Returns

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

7.19.3.3 `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.

7.19.3.4 `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.

7.19.3.5 `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

7.19.3.6 `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\(\)](#)

**7.19.3.7** `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

**7.19.4 Member Data Documentation**

**7.19.4.1** `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]`

**7.19.4.2** `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](#)



## 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](#)  
*Generates custom exceptions, used when validating function parameters.*

## Namespaces

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

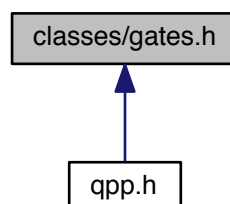
### 8.2.1 Detailed Description

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.*

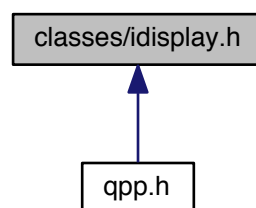
### 8.3.1 Detailed Description

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.*

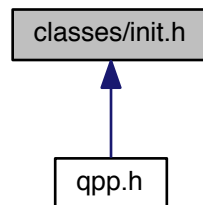
### 8.4.1 Detailed Description

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.*

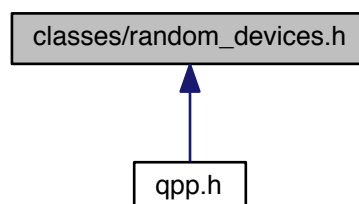
### 8.5.1 Detailed Description

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.*

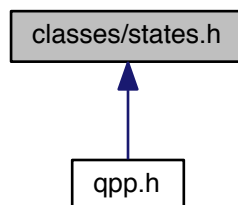
### 8.6.1 Detailed Description

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.*

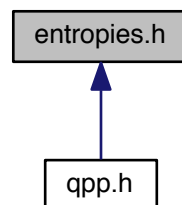
### 8.10.1 Detailed Description

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.*

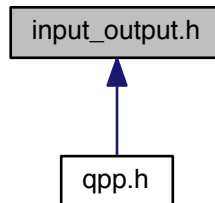
### 8.13.1 Detailed Description

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/modify.*

### 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/modify.*

### 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/modify.*

## 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

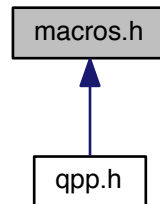
Internal utility functions.



## 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 >`  
`Derived qpp::loadMATLABmatrix (const std::string &, const std::string &)`  
*Loads an Eigen dynamic matrix from a MATLAB .mat file, generic version.*
- `template<>`  
`dmat qpp::loadMATLABmatrix (const std::string &mat_file, const std::string &var_name)`  
*Loads an Eigen dynamic matrix from a MATLAB .mat file, specialization for double matrices ([qpp::dmat](#))*
- `template<>`  
`cmat qpp::loadMATLABmatrix (const std::string &mat_file, const std::string &var_name)`  
*Loads an Eigen dynamic matrix from a MATLAB .mat file, specialization for complex matrices ([qpp::cmat](#))*
- `template<typename Derived >`  
`void qpp::saveMATLABmatrix (const Eigen::MatrixBase< Derived > &, const std::string &, const std::string &, const std::string &)`  
*Saves an Eigen dynamic matrix to a MATLAB .mat file, generic version.*
- `template<>`  
`void qpp::saveMATLABmatrix (const Eigen::MatrixBase< dmat > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)`  
*Saves an Eigen dynamic matrix to a MATLAB .mat file, specialization for double matrices ([qpp::dmat](#))*
- `template<>`  
`void qpp::saveMATLABmatrix (const Eigen::MatrixBase< cmatrix > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)`  
*Saves an Eigen dynamic matrix to a MATLAB .mat file, specialization for complex matrices ([qpp::cmat](#))*

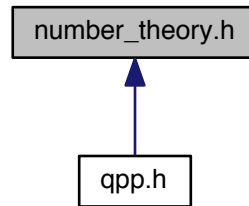
### 8.20.1 Detailed Description

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)`  
*Real representation of a simple continued fraction.*
- `double qpp::contfrac2x (const std::vector< int > &cf)`  
*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].*

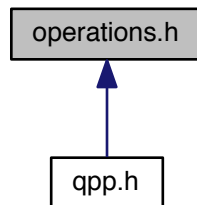
### 8.21.1 Detailed Description

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)  
*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)  
*Generates a set of random Kraus operators.*
- `cmat qpp::randH` (idx D)  
*Generates a random Hermitian matrix.*
- `ket qpp::randket` (idx D)  
*Generates a random normalized ket (pure state vector)*
- `cmat qpp::randrho` (idx D)  
*Generates a random density matrix.*
- `std::vector< idx > qpp::randperm` (idx N)  
*Generates a random uniformly distributed permutation.*

### 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.*

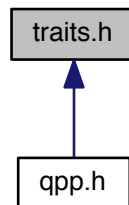
### 8.25.1 Detailed Description

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_matrix_expression< typename Eigen::MatrixBase< Derived > >`  
*Checks whether the type is an Eigen matrix expression, specialization for Eigen matrix expressions.*
- 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`.*

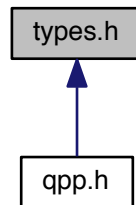
#### 8.26.1 Detailed Description

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, [98](#)
- ~IDisplay
  - qpp::IDisplay, [103](#)
- ~Init
  - qpp::Init, [105](#)
- ~RandomDevices
  - qpp::RandomDevices, [119](#)
- ~Singleton
  - qpp::internal::Singleton, [120](#)
- ~States
  - qpp::States, [123](#)
- ~Timer
  - qpp::Timer, [127](#)
- A\_
  - qpp::internal::IOManipEigen, [107](#)
- absm
  - qpp, [26](#)
- abssq
  - qpp, [27](#)
- adjoint
  - qpp, [27](#)
- anticomm
  - qpp, [28](#)
- apply
  - qpp, [28](#), [29](#)
- applyCTRL
  - qpp, [30](#)
- avg
  - qpp, [31](#)
- b00
  - qpp::States, [123](#)
- b01
  - qpp::States, [124](#)
- b10
  - qpp::States, [124](#)
- b11
  - qpp::States, [124](#)
- bigint
  - qpp, [25](#)
- bloch2rho
  - qpp, [31](#)
- bra
  - qpp, [25](#)
- CNOT
  - qpp::Gates, [100](#)
- CNOTba
  - qpp::Gates, [100](#)
- CTRL
  - qpp::Gates, [98](#)
- CUSTOM\_EXCEPTION
  - qpp::Exception, [94](#)
- CZ
  - qpp::Gates, [100](#)
- check\_cvector
  - qpp::internal, [86](#)
- check\_dims
  - qpp::internal, [86](#)
- check\_dims\_match\_cvect
  - qpp::internal, [86](#)
- check\_dims\_match\_mat
  - qpp::internal, [86](#)
- check\_dims\_match\_rvect
  - qpp::internal, [86](#)
- check\_eq\_dims
  - qpp::internal, [86](#)
- check\_matching\_sizes
  - qpp::internal, [86](#)
- check\_nonzero\_size
  - qpp::internal, [86](#)
- check\_perm
  - qpp::internal, [86](#)
- check\_qubit\_cvector
  - qpp::internal, [86](#)
- check\_qubit\_matrix
  - qpp::internal, [86](#)
- check\_qubit\_rvector
  - qpp::internal, [86](#)
- check\_qubit\_vector
  - qpp::internal, [86](#)
- check\_rvector
  - qpp::internal, [86](#)
- check\_square\_mat
  - qpp::internal, [86](#)
- check\_subsys\_match\_dims
  - qpp::internal, [86](#)
- check\_vector
  - qpp::internal, [86](#)
- choi2kraus
  - qpp, [31](#)
- choi2super
  - qpp, [32](#)
- chop
  - qpp, [83](#)

chop\_  
     qpp::internal::LOManipEigen, 107  
 classes/codes.h, 131  
 classes/exception.h, 131  
 classes/gates.h, 132  
 classes/ideisplay.h, 133  
 classes/init.h, 134  
 classes/random\_devices.h, 134  
 classes/states.h, 135  
 classes/timer.h, 136  
 cmat  
     qpp, 25  
 Codes  
     qpp::Codes, 90  
 codeword  
     qpp::Codes, 91  
 comm  
     qpp, 32  
 complement  
     qpp, 32  
 compperm  
     qpp, 32  
 concurrence  
     qpp, 34  
 conjugate  
     qpp, 34  
 constants.h, 136  
 construct\_exception\_msg\_  
     qpp::Exception, 95  
 contrfac2x  
     qpp, 34  
 cor  
     qpp, 35  
 cosm  
     qpp, 35  
 cov  
     qpp, 35  
 cplx  
     qpp, 25  
 custom\_  
     qpp::Exception, 95  
 cwise  
     qpp, 36  
  
 DIMS\_INVALID  
     qpp::Exception, 94  
 DIMS\_MISMATCH\_CVECTOR  
     qpp::Exception, 94  
 DIMS\_MISMATCH\_MATRIX  
     qpp::Exception, 94  
 DIMS\_MISMATCH\_RVECTOR  
     qpp::Exception, 94  
 DIMS\_MISMATCH\_VECTOR  
     qpp::Exception, 94  
 DIMS\_NOT\_EQUAL  
     qpp::Exception, 94  
 det  
     qpp, 36  
 dirsum  
     qpp, 36, 37  
 dirsum2  
     qpp::internal, 86  
 dirsumpow  
     qpp, 37  
 disp  
     qpp, 38, 39  
 display  
     qpp::IDisplay, 103  
     qpp::Timer, 128  
     qpp::internal::LOManipEigen, 106  
     qpp::internal::LOManipPointer, 108  
     qpp::internal::LOManipRange, 110  
 display\_impl\_  
     qpp::internal::Display\_Impl\_, 92  
 dmat  
     qpp, 25  
 dyn\_col\_vect  
     qpp, 25  
 dyn\_mat  
     qpp, 26  
 dyn\_row\_vect  
     qpp, 26  
  
 ERROR  
     macros.h, 151  
 ERRORLN  
     macros.h, 151  
 ee  
     qpp, 83  
 egcd  
     qpp, 39  
 eig  
     qpp, 40  
 end\_  
     qpp::Timer, 129  
     qpp::internal::LOManipPointer, 108  
     qpp::internal::LOManipRange, 110  
 entanglement  
     qpp, 40  
 entanglement.h, 137  
 entropies.h, 139  
 entropy  
     qpp, 41  
 eps  
     qpp, 84  
 evals  
     qpp, 41  
 evects  
     qpp, 41  
 Exception  
     qpp::Exception, 94, 95  
 expandout  
     qpp::Gates, 98  
 experimental/experimental.h, 140  
 expm  
     qpp, 42  
  
 FIVE\_QUBIT



- qpp::Codes, 90
- FRED
  - qpp::Gates, 100
- factors
  - qpp, 42
- Fd
  - qpp::Gates, 99
- first\_
  - qpp::internal::IOManipRange, 110
- functions.h, 140
- funm
  - qpp, 42
- GHZ
  - qpp::States, 124
- Gates
  - qpp::Gates, 98
- gcd
  - qpp, 42, 43
- gconcurrency
  - qpp, 43
- get\_dim\_subsys
  - qpp::internal, 86
- get\_duration
  - qpp::Timer, 128
- get\_instance
  - qpp::internal::Singleton, 121
- get\_num\_subsys
  - qpp::internal, 86
- get\_thread\_local\_instance
  - qpp::internal::Singleton, 121
- grams
  - qpp, 43, 44
- H
  - qpp::Gates, 100
- heig
  - qpp, 44
- hevals
  - qpp, 44
- hevects
  - qpp, 45
- IDisplay
  - qpp::IDisplay, 103
- IOManipEigen
  - qpp::internal::IOManipEigen, 106
- IOManipPointer
  - qpp::internal::IOManipPointer, 108
- IOManipRange
  - qpp::internal::IOManipRange, 110
- Id
  - qpp::Gates, 99
- Id2
  - qpp::Gates, 101
- idx
  - qpp, 26
- infty
  - qpp, 84
- Init
  - qpp::Init, 105
- input\_output.h, 145
- instruments.h, 146
- internal/classes/iomanip.h, 147
- internal/classes/singleton.h, 148
- internal/util.h, 149
- internal::Singleton< const Codes >
  - qpp::Codes, 91
- internal::Singleton< const Gates >
  - qpp::Gates, 100
- internal::Singleton< const Init >
  - qpp::Init, 105
- internal::Singleton< const States >
  - qpp::States, 123
- internal::Singleton< RandomDevices >
  - qpp::RandomDevices, 119
- inverse
  - qpp, 45
- invperm
  - qpp, 45
- ip
  - qpp, 45, 47
- isprime
  - qpp, 47
- ket
  - qpp, 26
- kraus2choi
  - qpp, 47
- kraus2super
  - qpp, 48
- kron
  - qpp, 48, 49
- kron2
  - qpp::internal, 86
- kronpow
  - qpp, 49
- last\_
  - qpp::internal::IOManipRange, 110
- lcm
  - qpp, 50
- load
  - qpp, 50
- loadMATLABmatrix
  - qpp, 51
- logdet
  - qpp, 52
- logm
  - qpp, 52
- lognegativity
  - qpp, 52
- MATLAB/matlab.h, 152
- MATRIX\_MISMATCH\_SUBSYS
  - qpp::Exception, 94
- MATRIX\_NOT\_CVECTOR
  - qpp::Exception, 94

MATRIX\_NOT\_RVECTOR  
     qpp::Exception, 94  
 MATRIX\_NOT\_SQUARE  
     qpp::Exception, 94  
 MATRIX\_NOT\_SQUARE\_OR\_CVECTOR  
     qpp::Exception, 94  
 MATRIX\_NOT\_SQUARE\_OR\_RVECTOR  
     qpp::Exception, 94  
 MATRIX\_NOT\_SQUARE\_OR\_VECTOR  
     qpp::Exception, 94  
 MATRIX\_NOT\_VECTOR  
     qpp::Exception, 94  
 macros.h, 151  
     ERROR, 151  
     ERRORLN, 151  
     PRINT, 151  
     PRINTLN, 151  
 marginalX  
     qpp, 53  
 marginalY  
     qpp, 53  
 maxn  
     qpp, 84  
 measure  
     qpp, 53–56  
 measure\_seq  
     qpp, 57  
 mes  
     qpp::States, 123  
 mket  
     qpp, 58  
 modinv  
     qpp, 58  
 modmul  
     qpp, 59  
 modpow  
     qpp, 59  
 mprj  
     qpp, 59, 60  
 msg\_  
     qpp::Exception, 95  
 multiidx2n  
     qpp, 60  
     qpp::internal, 86  
 n2multiidx  
     qpp, 60  
     qpp::internal, 87  
 N\_  
     qpp::internal::IOManipPointer, 108  
 NINE\_QUBIT\_SHOR  
     qpp::Codes, 90  
 NO\_CODEWORD  
     qpp::Exception, 94  
 NOT\_BIPARTITE  
     qpp::Exception, 94  
 NOT\_QUBIT\_CVECTOR  
     qpp::Exception, 94  
 NOT\_QUBIT\_MATRIX  
     qpp::Exception, 94  
 NOT\_QUBIT\_RVECTOR  
     qpp::Exception, 94  
 NOT\_QUBIT\_SUBSYS  
     qpp::Exception, 94  
 NOT\_QUBIT\_VECTOR  
     qpp::Exception, 94  
 negativity  
     qpp, 61  
 norm  
     qpp, 61  
 number\_theory.h, 152  
 OUT\_OF\_RANGE  
     qpp::Exception, 94  
 omega  
     qpp, 61  
 operations.h, 154  
 operator<<  
     qpp::IDisplay, 103  
 operator=  
     qpp::IDisplay, 103  
     qpp::Timer, 128  
     qpp::internal::IOManipPointer, 108  
     qpp::internal::IOManipRange, 110  
     qpp::internal::Singleton, 121  
 operator""\_i  
     qpp, 62  
 p\_  
     qpp::internal::IOManipPointer, 108  
 PERM\_INVALID  
     qpp::Exception, 94  
 PERM\_MISMATCH\_DIMS  
     qpp::Exception, 94  
 pGHZ  
     qpp::States, 124  
 PRINT  
     macros.h, 151  
 PRINTLN  
     macros.h, 151  
 pW  
     qpp::States, 124  
 pb00  
     qpp::States, 124  
 pb01  
     qpp::States, 124  
 pb10  
     qpp::States, 124  
 pb11  
     qpp::States, 124  
 pi  
     qpp, 84  
 powm  
     qpp, 62  
 prj  
     qpp, 62  
 prod  
     qpp, 63

- ptrace
  - qpp, [63](#), [64](#)
- ptrace1
  - qpp, [64](#)
- ptrace2
  - qpp, [65](#)
- ptranspose
  - qpp, [66](#)
- px0
  - qpp::States, [124](#)
- px1
  - qpp::States, [124](#)
- py0
  - qpp::States, [125](#)
- py1
  - qpp::States, [125](#)
- pz0
  - qpp::States, [125](#)
- pz1
  - qpp::States, [125](#)
- QPP\_UNUSED\_
  - qpp.h, [158](#)
- qmutualinfo
  - qpp, [66](#), [67](#)
- qpp, [13](#)
  - absm, [26](#)
  - abssq, [27](#)
  - adjoint, [27](#)
  - anticomm, [28](#)
  - apply, [28](#), [29](#)
  - applyCTRL, [30](#)
  - avg, [31](#)
  - bigint, [25](#)
  - bloch2rho, [31](#)
  - bra, [25](#)
  - choi2kraus, [31](#)
  - choi2super, [32](#)
  - chop, [83](#)
  - cmat, [25](#)
  - comm, [32](#)
  - complement, [32](#)
  - comperm, [32](#)
  - concurrence, [34](#)
  - conjugate, [34](#)
  - confrac2x, [34](#)
  - cor, [35](#)
  - cosm, [35](#)
  - cov, [35](#)
  - cplx, [25](#)
  - cwise, [36](#)
  - det, [36](#)
  - dirsum, [36](#), [37](#)
  - dirsumpow, [37](#)
  - disp, [38](#), [39](#)
  - dmat, [25](#)
  - dyn\_col\_vect, [25](#)
  - dyn\_mat, [26](#)
  - dyn\_row\_vect, [26](#)
  - ee, [83](#)
  - egcd, [39](#)
  - eig, [40](#)
  - entanglement, [40](#)
  - entropy, [41](#)
  - eps, [84](#)
  - evals, [41](#)
  - evects, [41](#)
  - expm, [42](#)
  - factors, [42](#)
  - funm, [42](#)
  - gcd, [42](#), [43](#)
  - gconcurrency, [43](#)
  - grams, [43](#), [44](#)
  - heig, [44](#)
  - hevals, [44](#)
  - hevects, [45](#)
  - idx, [26](#)
  - infty, [84](#)
  - inverse, [45](#)
  - invperm, [45](#)
  - ip, [45](#), [47](#)
  - isprime, [47](#)
  - ket, [26](#)
  - kraus2choi, [47](#)
  - kraus2super, [48](#)
  - kron, [48](#), [49](#)
  - kronpow, [49](#)
  - lcm, [50](#)
  - load, [50](#)
  - loadMATLABmatrix, [51](#)
  - logdet, [52](#)
  - logm, [52](#)
  - lognegativity, [52](#)
  - marginalX, [53](#)
  - marginalY, [53](#)
  - maxn, [84](#)
  - measure, [53–56](#)
  - measure\_seq, [57](#)
  - mket, [58](#)
  - modinv, [58](#)
  - modmul, [59](#)
  - modpow, [59](#)
  - mprj, [59](#), [60](#)
  - multiidx2n, [60](#)
  - n2multiidx, [60](#)
  - negativity, [61](#)
  - norm, [61](#)
  - omega, [61](#)
  - operator""\_i, [62](#)
  - pi, [84](#)
  - powm, [62](#)
  - prj, [62](#)
  - prod, [63](#)
  - ptrace, [63](#), [64](#)
  - ptrace1, [64](#)
  - ptrace2, [65](#)
  - ptranspose, [66](#)

- qmutualinfo, 66, 67
- rand, 67, 68
- randH, 68
- randU, 72
- randV, 72
- randidx, 68
- randket, 70
- randkraus, 70
- randn, 70, 71
- randperm, 71
- randprime, 72
- randrho, 72
- renyi, 73
- reshape, 73
- rho2bloch, 74
- rho2pure, 74
- save, 74
- saveMATLABmatrix, 75
- schatten, 75
- schmidtA, 76
- schmidtB, 76
- schmidtcoeffs, 77
- schmidtprobs, 77, 78
- sigma, 78
- sinm, 78
- spectralpowm, 79
- sqrtn, 79
- sum, 79, 80
- super2choi, 80
- svals, 80
- svd, 80
- svdU, 81
- svdV, 81
- syspermute, 81
- to\_void, 26
- trace, 82
- transpose, 82
- tsallis, 82
- uniform, 83
- var, 83
- x2confrac, 83
- qpp.h, 156
  - QPP\_UNUSED\_, 158
- 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::Exception, 92
  - CUSTOM\_EXCEPTION, 94
  - construct\_exception\_msg\_, 95
  - custom\_, 95
  - DIMS\_INVALID, 94
  - DIMS\_MISMATCH\_CVECTOR, 94
  - DIMS\_MISMATCH\_MATRIX, 94
  - DIMS\_MISMATCH\_RVECTOR, 94
  - DIMS\_MISMATCH\_VECTOR, 94
  - DIMS\_NOT\_EQUAL, 94
  - Exception, 94, 95
  - MATRIX\_MISMATCH\_SUBSYS, 94
  - MATRIX\_NOT\_CVECTOR, 94
  - MATRIX\_NOT\_RVECTOR, 94
  - MATRIX\_NOT\_SQUARE, 94
  - MATRIX\_NOT\_SQUARE\_OR\_CVECTOR, 94
  - MATRIX\_NOT\_SQUARE\_OR\_RVECTOR, 94
  - MATRIX\_NOT\_SQUARE\_OR\_VECTOR, 94
  - MATRIX\_NOT\_VECTOR, 94
  - msg\_, 95
  - NO\_CODEWORD, 94
  - NOT\_BIPARTITE, 94
  - NOT\_QUBIT\_CVECTOR, 94
  - NOT\_QUBIT\_MATRIX, 94
  - NOT\_QUBIT\_RVECTOR, 94
  - NOT\_QUBIT\_SUBSYS, 94
  - NOT\_QUBIT\_VECTOR, 94
  - OUT\_OF\_RANGE, 94
  - PERM\_INVALID, 94
  - PERM\_MISMATCH\_DIMS, 94
  - SIZE\_MISMATCH, 94
  - SUBSYS\_MISMATCH\_DIMS, 94
  - TYPE\_MISMATCH, 94
  - Type, 93
  - type\_, 95
  - UNDEFINED\_TYPE, 94
  - UNKNOWN\_EXCEPTION, 94
  - what, 95
  - where\_, 95
  - ZERO\_SIZE, 94
- qpp::Gates, 95
  - ~Gates, 98
  - CNOT, 100
  - CNOTba, 100
  - CTRL, 98
  - CZ, 100
  - expandout, 98
  - FRED, 100
  - Fd, 99
  - Gates, 98
  - H, 100
  - Id, 99
  - Id2, 101
  - internal::Singleton< const Gates >, 100
  - Rn, 99
  - S, 101
  - SWAP, 101
  - T, 101
  - TOF, 101
  - X, 101
  - Xd, 99
  - Y, 101
  - Z, 101
  - Zd, 100

- qpp::IDisplay, 101
  - ~IDisplay, 103
  - display, 103
  - IDisplay, 103
  - operator<<, 103
  - operator=, 103
- qpp::Init, 104
  - ~Init, 105
  - Init, 105
  - internal::Singleton< const Init >, 105
- qpp::RandomDevices, 117
  - ~RandomDevices, 119
  - internal::Singleton< RandomDevices >, 119
  - RandomDevices, 119
  - rd\_, 119
  - rng\_, 119
- qpp::States, 121
  - ~States, 123
  - b00, 123
  - b01, 124
  - b10, 124
  - b11, 124
  - GHZ, 124
  - internal::Singleton< const States >, 123
  - mes, 123
  - pGHZ, 124
  - pW, 124
  - pb00, 124
  - pb01, 124
  - pb10, 124
  - pb11, 124
  - px0, 124
  - px1, 124
  - py0, 125
  - py1, 125
  - pz0, 125
  - pz1, 125
  - States, 123
  - W, 125
  - x0, 125
  - x1, 125
  - y0, 125
  - y1, 125
  - z0, 125
  - z1, 125
- qpp::Timer
  - ~Timer, 127
  - display, 128
  - end\_, 129
  - get\_duration, 128
  - operator=, 128
  - start\_, 129
  - tic, 128
  - tics, 128
  - Timer, 127
  - toc, 129
- qpp::Timer< T, CLOCK\_T >, 126
- qpp::experimental, 84
- qpp::internal, 84
  - check\_cvector, 86
  - check\_dims, 86
  - check\_dims\_match\_cvect, 86
  - check\_dims\_match\_mat, 86
  - check\_dims\_match\_rvect, 86
  - check\_eq\_dims, 86
  - check\_matching\_sizes, 86
  - check\_nonzero\_size, 86
  - check\_perm, 86
  - check\_qubit\_cvector, 86
  - check\_qubit\_matrix, 86
  - check\_qubit\_rvector, 86
  - check\_qubit\_vector, 86
  - check\_rvector, 86
  - check\_square\_mat, 86
  - check\_subsys\_match\_dims, 86
  - check\_vector, 86
  - dirsum2, 86
  - get\_dim\_subsys, 86
  - get\_num\_subsys, 86
  - kron2, 86
  - multiidx2n, 86
  - n2multiidx, 87
  - variadic\_vector\_emplace, 87
- qpp::internal::Display\_Impl\_, 91
  - display\_impl\_, 92
- qpp::internal::IOManipEigen, 105
  - A\_, 107
  - chop\_, 107
  - display, 106
  - IOManipEigen, 106
- qpp::internal::IOManipPointer
  - display, 108
  - end\_, 108
  - IOManipPointer, 108
  - N\_, 108
  - operator=, 108
  - p\_, 108
  - separator\_, 108
  - start\_, 109
- qpp::internal::IOManipPointer< PointerType >, 107
- qpp::internal::IOManipRange
  - display, 110
  - end\_, 110
  - first\_, 110
  - IOManipRange, 110
  - last\_, 110
  - operator=, 110
  - separator\_, 110
  - start\_, 111
- qpp::internal::IOManipRange< InputIterator >, 109
- qpp::internal::Singleton
  - ~Singleton, 120
  - get\_instance, 121
  - get\_thread\_local\_instance, 121
  - operator=, 121
  - Singleton, 120

- qpp::internal::Singleton< T >, 119
- qpp::is\_complex< std::complex< T > >, 112
- qpp::is\_complex< T >, 111
- qpp::is\_iterable< T, to\_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), typename T::value\_type > >, 114
- qpp::is\_iterable< T, typename >, 113
- qpp::is\_matrix\_expression< Derived >, 115
- qpp::is\_matrix\_expression< typename Eigen::Matrix< Base< Derived > >, 116
- rand
  - qpp, 67, 68
- randH
  - qpp, 68
- randU
  - qpp, 72
- randV
  - qpp, 72
- randidx
  - qpp, 68
- randket
  - qpp, 70
- randkraus
  - qpp, 70
- randn
  - qpp, 70, 71
- random.h, 158
- RandomDevices
  - qpp::RandomDevices, 119
- randperm
  - qpp, 71
- randprime
  - qpp, 72
- randrho
  - qpp, 72
- rd\_
  - qpp::RandomDevices, 119
- renyi
  - qpp, 73
- reshape
  - qpp, 73
- rho2bloch
  - qpp, 74
- rho2pure
  - qpp, 74
- Rn
  - qpp::Gates, 99
- rng\_
  - qpp::RandomDevices, 119
- S
  - qpp::Gates, 101
- SEVEN\_QUBIT\_STEANE
  - qpp::Codes, 90
- SIZE\_MISMATCH
  - qpp::Exception, 94
- SUBSYS\_MISMATCH\_DIMS
  - qpp::Exception, 94
- SWAP
  - qpp::Gates, 101
- save
  - qpp, 74
- saveMATLABmatrix
  - qpp, 75
- schatten
  - qpp, 75
- schmidtA
  - qpp, 76
- schmidtB
  - qpp, 76
- schmidtcoeffs
  - qpp, 77
- schmidtprobs
  - qpp, 77, 78
- separator\_
  - qpp::internal::IOManipPointer, 108
  - qpp::internal::IOManipRange, 110
- sigma
  - qpp, 78
- Singleton
  - qpp::internal::Singleton, 120
- sinm
  - qpp, 78
- spectralpowm
  - qpp, 79
- sqrtn
  - qpp, 79
- start\_
  - qpp::Timer, 129
  - qpp::internal::IOManipPointer, 109
  - qpp::internal::IOManipRange, 111
- States
  - qpp::States, 123
- statistics.h, 159
- sum
  - qpp, 79, 80
- super2choi
  - qpp, 80
- svals
  - qpp, 80
- svd
  - qpp, 80
- svdU
  - qpp, 81
- svdV
  - qpp, 81
- syspermute
  - qpp, 81
- T
  - qpp::Gates, 101
- TOF
  - qpp::Gates, 101
- TYPE\_MISMATCH
  - qpp::Exception, 94
- tic
  - qpp::Timer, 128

tics  
    qpp::Timer, 128  
Timer  
    qpp::Timer, 127  
to\_void  
    qpp, 26  
toc  
    qpp::Timer, 129  
trace  
    qpp, 82  
traits.h, 161  
transpose  
    qpp, 82  
tsallis  
    qpp, 82  
Type  
    qpp::Codes, 90  
    qpp::Exception, 93  
type\_  
    qpp::Exception, 95  
types.h, 162  
  
UNDEFINED\_TYPE  
    qpp::Exception, 94  
UNKNOWN\_EXCEPTION  
    qpp::Exception, 94  
uniform  
    qpp, 83  
  
var  
    qpp, 83  
variadic\_vector\_emplace  
    qpp::internal, 87  
  
W  
    qpp::States, 125  
what  
    qpp::Exception, 95  
where\_  
    qpp::Exception, 95  
  
X  
    qpp::Gates, 101  
x0  
    qpp::States, 125  
x1  
    qpp::States, 125  
x2contfrac  
    qpp, 83  
Xd  
    qpp::Gates, 99  
  
Y  
    qpp::Gates, 101  
y0  
    qpp::States, 125  
y1  
    qpp::States, 125  
  
Z  
    qpp::Gates, 101  
z0  
    qpp::States, 125  
z1  
    qpp::States, 125  
ZERO\_SIZE  
    qpp::Exception, 94  
Zd  
    qpp::Gates, 100