quantum++ 0.1

Generated by Doxygen 1.8.7

Thu Nov 6 2014 12:56:16

## **Contents**

1	Qua	ntum++	- A C++1	1 quan	tum co	ompi	uting	g lib	rary							1
2	Nam	nespace	Index													5
	2.1	Names	space List							 	 	 	 	 	 	5
3	Hier	archica	l Index													7
	3.1	Class I	Hierarchy							 	 	 	 	 	 	7
4	Clas	s Index														9
	4.1	Class I	List							 	 	 	 	 	 	9
5	File	Index														11
	5.1	File Lis	st							 	 	 	 	 	 	11
6	Nam	nespace	Docume	ntation												13
	6.1	qpp Na	amespace	Refere	nce .					 	 	 	 	 	 	13
		6.1.1	Typedef	Docum	entatio	n				 	 	 	 	 	 	21
			6.1.1.1	bra .						 	 	 	 	 	 	21
			6.1.1.2	cmat						 	 	 	 	 	 	21
			6.1.1.3	cplx .						 	 	 	 	 	 	21
			6.1.1.4	dmat						 	 	 	 	 	 	21
			6.1.1.5	DynC	olVect					 	 	 	 	 	 	22
			6.1.1.6	DynN	lat					 	 	 	 	 	 	22
			6.1.1.7	DynR	lowVec	t				 	 	 	 	 	 	22
			6.1.1.8	ket .						 	 	 	 	 	 	22
		6.1.2	Function	Docum	nentatio	on .				 	 	 	 	 	 	22
			6.1.2.1	absm						 	 	 	 	 	 	22
			6.1.2.2	absso	<b>.</b>					 	 	 	 	 	 	23
			6.1.2.3	absso	<b>.</b>					 	 	 	 	 	 	23
			6.1.2.4	adjoir	nt					 	 	 	 	 	 	24
			6.1.2.5	antico	omm .					 	 	 	 	 	 	24
			6.1.2.6	apply						 	 	 	 	 	 	25
			0407		OTDI											

iv CONTENTS

6.1.2.8	channel	27
6.1.2.9	channel	28
6.1.2.10	choi	29
6.1.2.11	choi2kraus	30
6.1.2.12	comm	31
6.1.2.13	compperm	32
6.1.2.14	concurrence	33
6.1.2.15	conjugate	34
6.1.2.16	cosm	35
6.1.2.17	cwise	35
6.1.2.18	det	36
6.1.2.19	disp	36
6.1.2.20	disp	37
6.1.2.21	disp	37
6.1.2.22	disp	38
6.1.2.23	disp	38
6.1.2.24	displn	39
6.1.2.25	displn	39
6.1.2.26	displn	40
6.1.2.27	displn	41
6.1.2.28	displn	41
6.1.2.29	entanglement	42
6.1.2.30	evals	43
6.1.2.31	evects	44
6.1.2.32	expm	44
6.1.2.33	funm	45
6.1.2.34	gconcurrence	45
6.1.2.35	grams	46
6.1.2.36	grams	47
6.1.2.37	grams	47
6.1.2.38	hevals	48
6.1.2.39	hevects	48
6.1.2.40	inverse	49
6.1.2.41	invperm	49
6.1.2.42	kron	50
6.1.2.43	kron	50
6.1.2.44	kron	51
6.1.2.45	kron	51
6.1.2.46	kronpow	52
6.1.2.47	load	52

CONTENTS

6.1.2.48	loadMATLABmatrix	53
6.1.2.49	loadMATLABmatrix	53
6.1.2.50	loadMATLABmatrix	53
6.1.2.51	logdet	54
6.1.2.52	logm	54
6.1.2.53	lognegativity	55
6.1.2.54	measure	56
6.1.2.55	measure	56
6.1.2.56	measure	57
6.1.2.57	mket	58
6.1.2.58	mket	58
6.1.2.59	mprj	59
6.1.2.60	mprj	59
6.1.2.61	multiidx2n	60
6.1.2.62	n2multiidx	60
6.1.2.63	negativity	61
6.1.2.64	norm	62
6.1.2.65	omega	62
6.1.2.66	operator""""_i	63
6.1.2.67	operator"""_i	63
6.1.2.68	powm	63
6.1.2.69	prj	64
6.1.2.70	prod	64
6.1.2.71	prod	65
6.1.2.72	ptrace	65
6.1.2.73	ptrace1	66
6.1.2.74	ptrace2	67
6.1.2.75	ptranspose	68
6.1.2.76	qmutualinfo	69
6.1.2.77	rand	70
6.1.2.78	rand	70
6.1.2.79	rand	71
6.1.2.80	rand	71
6.1.2.81	randH	72
6.1.2.82	randint	72
6.1.2.83	randket	73
6.1.2.84	randkraus	73
6.1.2.85	randn	74
6.1.2.86		74
6.1.2.87	randn	75

vi CONTENTS

	6.1.2.88	randn	. /5
	6.1.2.89	randperm	. 76
	6.1.2.90	randrho	. 76
	6.1.2.91	randU	. 77
	6.1.2.92	$randV \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots$	. 77
	6.1.2.93	renyi	. 77
	6.1.2.94	reshape	. 78
	6.1.2.95	save	. 79
	6.1.2.96	saveMATLABmatrix	. 79
	6.1.2.97	saveMATLABmatrix	. 79
	6.1.2.98	saveMATLABmatrix	. 79
	6.1.2.99	schatten	. 80
	6.1.2.100	schmidtcoeff	. 81
	6.1.2.101	schmidtprob	. 81
	6.1.2.102	schmidtU	. 82
	6.1.2.103	schmidtV	. 83
	6.1.2.104	shannon	. 84
	6.1.2.105	sinm	. 85
	6.1.2.106	spectralpowm	. 85
	6.1.2.107	sqrtm	. 86
	6.1.2.108	sum	. 86
	6.1.2.109	sum	. 87
	6.1.2.110	super	. 87
	6.1.2.111	svals	. 88
	6.1.2.112	svdU	. 88
	6.1.2.113	svdV	. 89
	6.1.2.114	syspermute	. 89
	6.1.2.115	trace	. 90
	6.1.2.116	transpose	. 91
	6.1.2.117	tsallis	. 91
6.1.3	Variable D	Documentation	. 92
	6.1.3.1	chop	. 92
	6.1.3.2	codes	. 92
	6.1.3.3	ee	. 92
	6.1.3.4	eps	. 92
	6.1.3.5	$gt \ldots \ldots \ldots \ldots \ldots \ldots$	. 92
	6.1.3.6	infty	. 93
	6.1.3.7	init	. 93
	6.1.3.8	maxn	. 93
	6.1.3.9	pi	. 93

CONTENTS vii

			6.1.3.10 rdevs
			6.1.3.11 st
	6.2	qpp::ex	perimental Namespace Reference
		6.2.1	Detailed Description
		6.2.2	Function Documentation
			6.2.2.1 apply
			6.2.2.2 channel
			6.2.2.3 choi
			6.2.2.4 CTRL
			6.2.2.5 randkraus
			6.2.2.6 renyi_inf
			6.2.2.7 super
	6.3	qpp::in	ernal Namespace Reference
		6.3.1	Detailed Description
		6.3.2	Function Documentation
			6.3.2.1 _check_col_vector
			6.3.2.2 _check_dims
			6.3.2.3 _check_dims_match_cvect
			6.3.2.4 _check_dims_match_mat
			6.3.2.5 _check_dims_match_rvect
			6.3.2.6 _check_eq_dims
			6.3.2.7 _check_nonzero_size
			6.3.2.8 _check_perm
			6.3.2.9 _check_row_vector
			6.3.2.10 _check_square_mat
			6.3.2.11 _check_subsys_match_dims
			6.3.2.12 _check_vector
			6.3.2.13 _kron2
			6.3.2.14 _multiidx2n
			6.3.2.15 _n2multiidx
			6.3.2.16 variadic_vector_emplace
			6.3.2.17 variadic_vector_emplace
7	Clos	o Doou	nentation 10
'	7.1		des Class Reference
	7.1	7.1.1	Detailed Description
		7.1.2	Member Enumeration Documentation
		1.1.4	7.1.2.1 Type
		7.1.3	Constructor & Destructor Documentation
		1.1.3	7.1.3.1 Codes
			7.1.5.1 Godes

viii CONTENTS

	7.1.4	Member Function Documentation
		7.1.4.1 codeword
	7.1.5	Friends And Related Function Documentation
		7.1.5.1 internal::Singleton < const Codes >
7.2	qpp::E	xception Class Reference
	7.2.1	Detailed Description
	7.2.2	Member Enumeration Documentation
		7.2.2.1 Type
	7.2.3	Constructor & Destructor Documentation
		7.2.3.1 Exception
		7.2.3.2 Exception
	7.2.4	Member Function Documentation
		7.2.4.1 _construct_exception_msg
		7.2.4.2 what
	7.2.5	Member Data Documentation
		7.2.5.1 _custom
		7.2.5.2 _msg
		7.2.5.3 _type
		7.2.5.4 _where
7.3	qpp::G	ates Class Reference
	7.3.1	Detailed Description
	7.3.2	Constructor & Destructor Documentation
		7.3.2.1 Gates
	7.3.3	Member Function Documentation
		7.3.3.1 CTRL
		7.3.3.2 expandout
		7.3.3.3 Fd
		7.3.3.4 ld
		7.3.3.5 Rn
		7.3.3.6 Xd
		7.3.3.7 Zd
	7.3.4	Friends And Related Function Documentation
		7.3.4.1 internal::Singleton < const Gates >
	7.3.5	Member Data Documentation
		7.3.5.1 CNOTab
		7.3.5.2 CNOTba
		7.3.5.3 CZ
		7.3.5.4 FRED
		7.3.5.5 H
		7.3.5.6 ld2

CONTENTS

		7.3.5.7	S	118
		7.3.5.8	SWAP 1	18
		7.3.5.9	T	18
		7.3.5.10	TOF 1	18
		7.3.5.11	X	18
		7.3.5.12	Y	18
		7.3.5.13	<b>Z</b>	119
7.4	qpp::ln	it Class Re	eference	119
	7.4.1	Detailed	Description	20
	7.4.2	Construc	tor & Destructor Documentation	20
		7.4.2.1	Init	20
		7.4.2.2	∼Init	20
	7.4.3	Friends A	And Related Function Documentation	20
		7.4.3.1	internal::Singleton< const Init >	20
7.5	qpp::ex	xperimenta	al::Qudit Class Reference	20
	7.5.1	Construc	tor & Destructor Documentation	21
		7.5.1.1	Qudit	21
	7.5.2	Member	Function Documentation	21
		7.5.2.1	getD	21
		7.5.2.2	getRho	21
		7.5.2.3	measure	21
		7.5.2.4	measure	22
	7.5.3	Member	Data Documentation	22
		7.5.3.1	_D	22
		7.5.3.2	_rho	22
7.6	qpp::R	andomDe	vices Class Reference	22
	7.6.1	Detailed	Description	23
	7.6.2	Construc	tor & Destructor Documentation	23
		7.6.2.1	RandomDevices	23
	7.6.3	Friends A	And Related Function Documentation	24
		7.6.3.1	internal::Singleton< RandomDevices >	24
	7.6.4	Member	Data Documentation	24
		7.6.4.1	_rd	24
		7.6.4.2	_rng	24
7.7	qpp::in	ternal::Sin	gleton < T > Class Template Reference	24
	7.7.1	Detailed	Description	24
	7.7.2	Construc	tor & Destructor Documentation	25
		7.7.2.1	Singleton	25
		7.7.2.2	~Singleton	25
		7.7.2.3	Singleton	25

X CONTENTS

	7.7.3	Member Function Documentation
		7.7.3.1 get_instance
		7.7.3.2 operator=
7.8	qpp::St	tates Class Reference
	7.8.1	Detailed Description
	7.8.2	Constructor & Destructor Documentation
		7.8.2.1 States
	7.8.3	Friends And Related Function Documentation
		7.8.3.1 internal::Singleton < const States >
	7.8.4	Member Data Documentation
		7.8.4.1 b00
		7.8.4.2 b01
		7.8.4.3 b10
		7.8.4.4 b11
		7.8.4.5 GHZ
		7.8.4.6 pb00
		7.8.4.7 pb01
		7.8.4.8 pb10
		7.8.4.9 pb11
		7.8.4.10 pGHZ
		7.8.4.11 pW
		7.8.4.12 px0
		7.8.4.13 px1
		7.8.4.14 py0
		7.8.4.15 py1
		7.8.4.16 pz0
		7.8.4.17 pz1
		7.8.4.18 W
		7.8.4.19 x0
		7.8.4.20 x1
		7.8.4.21 y0
		7.8.4.22 y1
		7.8.4.23 z0
		7.8.4.24 z1
7.9	qpp::Ti	mer Class Reference
	7.9.1	Detailed Description
	7.9.2	Constructor & Destructor Documentation
		7.9.2.1 Timer
	7.9.3	Member Function Documentation
		7.9.3.1 seconds

CONTENTS xi

			7.9.3.2 tic	130
			7.9.3.3 toc	131
		7.9.4	Friends And Related Function Documentation	131
			7.9.4.1 operator<<	131
		7.9.5	Member Data Documentation	131
			7.9.5.1 _end	131
			7.9.5.2 _start	131
8	File I	Docume	pentation	133
Ŭ	8.1			133
	8.2		e/classes/exception.h File Reference	
	8.3			
			e/classes/gates.h File Reference	
	8.4		e/classes/init.h File Reference	
	8.5		e/classes/randevs.h File Reference	
	8.6		e/classes/singleton.h File Reference	
	8.7		e/classes/states.h File Reference	
	8.8	include	e/classes/timer.h File Reference	137
	8.9	include	e/constants.h File Reference	138
	8.10	include	e/entanglement.h File Reference	139
	8.11	include	e/entropies.h File Reference	140
	8.12	include	e/experimental/classes/qudit.h File Reference	141
	8.13	include	e/experimental/test.h File Reference	141
	8.14	include	e/functions.h File Reference	142
	8.15	include	e/internal/functions.h File Reference	146
	8.16	include	e/instruments.h File Reference	147
	8.17	include	e/io.h File Reference	148
	8.18	include	e/MATLAB/matlab.h File Reference	149
	8.19	include	e/operations.h File Reference	150
	8.20	include	e/qpp.h File Reference	152
	8.21	include	e/random.h File Reference	153
	8.22	include	e/types.h File Reference	154
			age.dox File Reference	
		•		
Inc	lex			156

### **Chapter 1**

## Quantum++ - A C++11 quantum computing library

```
Version
     0.1
Author
     Vlad Gheorghiu (vgheorgh@gmail.com)
Copyright
     (c) 2013 - 2014 Vlad Gheorghiu (vgheorgh@gmail.com)
An example is worth more than one thousand words:)
#include "gpp.h"
// #include "MATLAB/matlab.h" // support for MATLAB
using namespace qpp;
int main()
    // Qudit Teleportation
       ket mes_AB = ket::Zero(D * D); // maximally entangled state resource
for (std::size_t i = 0; i < D; i++)
   mes_AB += mket( { i, i }, D);</pre>
       mes_AB /= std::sqrt((double) D);
       cmat Bell_aA = adjoint( // circuit that measures in the qudit Bell basis
       std::cout << ">> Initial state: " << std::endl;
       displn(psi_a);
ket input_aAB = kron(psi_a, mes_AB); // joint input state aAB
        // output before measurement
       std::discrete_distribution<std::size_t> dd(measured_aA.first.begin(),
       measured_aA.first.end());
std::cout << ">> Measurement probabilities: ";
       displn(measured_aA.first, ", ");
       std::size_t m = dd(rdevs._rng); // sample
auto midx = n2multiidx(m, { D, D });
       std::cout << ">> Measurement result: ";
       displn(midx, " ");
        // conditional result on B before correction
       ket output_m_aAB = apply(output_aAB, prj(mket(midx, D)), { 0, 1 }, 3, D)
               / std::sqrt(measured_aA.first[m]);
       cmat correction_B = powm(gt.Zd(D), midx[0])
                 powm(adjoint(gt.Xd(D)), midx[1]); // correction operator
```

```
// apply correction on B
       output_aAB = apply(output_m_aAB, correction_B, { 2 }, 3, D);
       cmat rho_B = ptracel(prj(output_aAB), { D * D, D });
std::cout << ">> Bob's density operator: " << std::endl;</pre>
       displn(rho_B);
                                  ">> Norm difference: " << norm(rho_B - prj(psi_a))
       std::cout <<
                       << std::endl; // verification
// Oudit Dense Coding
       ket mes_AB = ket::Zero(D * D); // maximally entangled state resource
       for (std::size_t i = 0; i < D; i++)
  mes_AB += mket( { i, i }, D);</pre>
       mes_AB /= std::sqrt((double) D);
       cmat Bell_AB = adjoint( // circuit that measures in the qudit Bell basis
                      gt.CTRL(gt.Xd(D), { 0 }, { 1 }, 2, D)
                                          kron(gt.Fd(D), gt.Id(D)));
       // equal probabilities of choosing a message
       std::uniform_int_distribution<std::size_t> uid(0, D * D - 1);
std::size_t m_A = uid(rdevs._rng); // sample, obtain the message index
auto midx = n2multiidx(m_A, { D, D });
std::cout << ">> Alice sent: ";
displn(midx, " ");
       // Alice's operation
       cmat U_A = powm(gt.Zd(D), midx[0]) * powm(adjoint(
   gt.Xd(D)), midx[1]);
// Alice encodes the message
       ket psi_AB = apply(mes_AB, U_A, { 0 }, 2, D);
       // Bob measures the joint system in the qudit Bell basis
       psi\_AB = apply(psi\_AB, Bell\_AB, { 0, 1 }, 2, D);
       auto measured = measure(psi_AB, gt.Id(D * D));
std::cout << ">> Bob measurement probabilities: ";
displn(measured.first, ", ");
       // Bob samples according to the measurement probabilities
       std::discrete_distribution<std::size_t> dd(measured.first.begin(),
                      measured.first.end());
       std::size_t m_B = dd(rdevs._rng);
std::cout << ">> Bob received: ";
       displn(n2multiidx(m_B, { D, D }), " ");
// Grover's search algorithm, we time it
       Timer t; // set a timer
std::size_t n = 4; // number of qubits
       std::cout << std::endl << "**** Grover on n = " << n << " qubits ****"
                        << std::endl;
       // mark an element randomly
       std::uniform_int_distribution<std::size_t> uid(0, N - 1);
       std::size_t marked = uid(rdevs._rng);
std::cout << ">> Marked state: " << marked << " -> ";
displn(n2multiidx(marked, dims), " ");
        \begin{tabular}{ll} ket &psi = mket(n2multiidx(0, dims)); // computational |0>^\ootimes n | (0>) &psi = mket(n2multiidx(0, dims)); // computational |0>^\ootimes n | (0>) &psi = mket(n2multiidx(0, dims)); // computational |0>^\ootimes n | (0>) &psi = mket(n2multiidx(0, dims)); // computational |0>^\ootimes n | (0>) &psi = mket(n2multiidx(0, dims)); // computational |0>^\ootimes n | (0>) &psi = mket(n2multiidx(0, dims)); // computational |0>^\ootimes n | (0>) &psi = mket(n2multiidx(0, dims)); // computational |0>^\ootimes n | (0>) &psi = mket(n2multiidx(0, dims)); // computational |0>^\ootimes n | (0>) &psi = mket(n2multiidx(0, dims)); // computational |0> &psi = mket(n2multiidx(0, dims)); // computational |0> &psi = mket(n2multiidx(0, dims)); // computational | (0>) &psi = mket(n2multiidx(0, dims)); // computational | (0>)
       psi = (kronpow(gt.H, n) * psi).eval(); // apply H^\otimes n, no aliasing
       cmat G = 2 * prj(psi) - gt.Id(N); // Diffusion operator
        // number of queries
       std::size_t nqueries = std::ceil(pi * std::sqrt(N) / 4.);
std::cout << ">> We run " << nqueries << " queries" << std::endl;
for (std::size_t i = 0; i < nqueries; i++)</pre>
              psi(marked) = -psi(marked); // apply the oracle first, no aliasing psi = (G \star psi).eval(); // then the diffusion operator, no aliasing
       // we now measure the state in the computational basis
       std::cout << ">> Probability of all results: ";
displn(measured.first, ", ");
       std::cout << ">> Let's sample..." << std::endl;
       std::discrete_distribution<std::size_t> dd(measured.first.begin(),
                      measured.first.end());
       std::size_t result = dd(rdevs._rng);
if (result == marked)
               std::cout << ">> Hooray, we obtained the correct result: ";
               std::cout << ">> Not there yet... we obtained: ";
       std::cout << result << " -> ";
       displn(n2multiidx(result, dims), " ");
       // stop the timer and display it
```

```
std::cout << ">> It took " << t.toc()
              << " seconds to simulate Grover on " << n << " qubits."
              << std::endl;
}
// Entanglement
    std::cout << std::endl << "**** Entanglement ****" << std::endl;</pre>
    cmat rho = 0.2 * st.pb00 + 0.8 * st.pb11;
std::cout << ">> rho: " << std::endl;</pre>
    displn(rho):
    std::cout << ">>> Concurrence of rho: " << concurrence(rho) << std::endl;
std::cout << ">>> Negativity of rho: " << negativity(rho, { 2, 2 })</pre>
              << std::endl;
     std::cout << ">> Logarithimc negativity of rho: "
    << lognegativity(rho, { 2, 2 }) << std::endl;
ket psi = 0.8 * mket( { 0, 0 }) + 0.6 * mket( { 1, 1 });</pre>
    // apply some local random unitaries
psi = kron(randU(2), randU(2)) * psi;
     std::cout << ">> psi: " << std::endl;
    displn(psi);
std::cout << ">> Entanglement of psi: " << entanglement(psi, { 2, 2 })</pre>
              << std::endl;
    std::cout << ">> Concurrence of psi: " << concurrence(prj(psi))</pre>
              << std::endl;
     std::cout << ">> G-Concurrence of psi: " << gconcurrence(psi)
              << std::endl;
     std::cout << ">> Schmidt coefficients of psi: " << std::endl;
    displn(schmidtcoeff(psi, { 2, 2 }));
std::cout << ">>> Schmidt probabilities of psi: " << std::endl;</pre>
    displn(schmidtprob(psi, { 2, 2 }));
cmat U = schmidtU(psi, { 2, 2 });
cmat V = schmidtV(psi, { 2, 2 });
     std::cout << ">> Schmidt vectors on Alice's side: " << std::endl;</pre>
    displn(U);
    std::cout << ">> Schmidt vectors on Bob's side: " << std::endl;
    displn(V);
    std::cout << ">> State psi in the Schmidt basis: " << std::endl;
    displn(adjoint(kron(U, V)) * psi);
     // reconstructed state
    ket psi_from_schmidt = schmidtcoeff(psi, { 2, 2 })(0)
                kron(U.col(0), V.col(0))
               + schmidtcoeff(psi, { 2, 2 })(1) * kron(U.col(1), V.col(1));
    std::cout
              << ">> State psi reconstructed from the Schmidt decomposition: " \,
              << std::endl;
    displn(psi_from_schmidt);
std::cout << ">>> Norm difference: " << norm(psi - psi_from_schmidt)</pre>
              << std::endl;
}
// Quantum error correcting codes
     std::cout << std::endl << "**** Quantum error correcting codes ****"
              << std::endl;
    ket a0 = codes.codeword(Codes::Type::FIVE_QUBIT, 0);
     ket a1 = codes.codeword(Codes::Type::FIVE_QUBIT, 1);
    ket b0 = codes.codeword(Codes::Type::SEVEN_QUBIT_STEANE
  , 0);
    ket b1 = codes.codeword(Codes::Type::SEVEN_QUBIT_STEANE
  , 1);
    ket c0 = codes.codeword(Codes::Type::NINE QUBIT SHOR, 0
  );
    ket c1 = codes.codeword(Codes::Type::NINE_QUBIT_SHOR, 1
  );
    std::cout << ">> Five qubit [[5, 1, 3]] code. ";
    std::cout << "Checking codeword orthogonality." << std::endl;
std::cout << ">> <0L | 1L> = ";
    displn(adjoint(a0) * a1);
    std::cout << ">> Seven qubit [[7, 1, 3]] Steame code. ";
     std::cout << "Checking codeword orthogonality." << std::endl;
    std::cout << ">> <0L | 1L> = ";
    std::cout << ">> coul | lb = ";

displn(adjoint(b0) * b1);

std::cout << ">> Nine qubit [[9, 1, 3]] Shor code. ";

std::cout << "Checking codeword orthogonality." << std::endl;

std::cout << ">> <0L | lL> = ";
    displn(adjoint(c0) * c1);
}
```

1

Quantum++ - A C++11 quantum computing library
quantum empating instary

# Chapter 2

# Namespace Index

Z. I Mailles Dace Lis	2.1	Nam	espace	List
-----------------------	-----	-----	--------	------

Horo	ie ·	a liet	of :	ااد	namespaces	with	hriof	descript	lione:
пеге	15	a iisi	OI 8	all	namespaces	WILLI	briei	descrip	แบบเร.

qpp	
qpp::experimental	
qpp::internal	

6 Namespace Index

# **Chapter 3**

## **Hierarchical Index**

### 3.1 Class Hierarchy

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

ta::exception	
qpp::Exception	. 107
pp::experimental::Qudit	120
$pp$ ::internal::Singleton $T>\ldots\ldots\ldots\ldots\ldots$	124
${\sf pp::internal::Singleton} < {\sf const} \ {\sf Codes} > \dots $	124
qpp::Codes	. 105
pp::internal::Singleton $<$ const Gates $>$	124
qpp::Gates	. 111
pp::internal::Singleton $<$ const Init $>$	124
qpp::Init	. 119
pp::internal::Singleton < const States >	124
qpp::States	. 125
pp::internal::Singleton< RandomDevices >	124
qpp::RandomDevices	. 122
pp::Timer	129

8 **Hierarchical Index** 

# **Chapter 4**

## **Class Index**

### 4.1 Class List

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

ppCodes	
Const Singleton class that defines quantum error correcting codes	105
pp::Exception	
Generates custom exceptions, used when validating function parameters	107
pp::Gates	
Const Singleton class that implements most commonly used gates	111
pp::Init	
Const Singleton class that performs additional initializations/cleanups	119
pp::experimental::Qudit	120
pp::RandomDevices	
Singeleton class that manages the source of randomness in the library	122
pp::internal::Singleton < T >	
Singleton policy class, used internally to implement the singleton pattern via CRTP (Curiously	
recurring template pattern)	124
pp::States	
Const Singleton class that implements most commonly used states	125
pp::Timer	
Measures time	129

10 Class Index

# **Chapter 5**

## File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

include/constants.h
include/entanglement.h
include/entropies.h
include/functions.h
include/instruments.h
include/io.h
include/operations.h
include/qpp.h
include/random.h
include/types.h
include/classes/codes.h
include/classes/exception.h
include/classes/gates.h
include/classes/init.h
include/classes/randevs.h
include/classes/singleton.h
include/classes/states.h
include/classes/timer.h
include/experimental/test.h
include/experimental/classes/qudit.h
include/internal/functions.h
include/MATLAR/matlab h

12 File Index

### **Chapter 6**

### **Namespace Documentation**

#### 6.1 qpp Namespace Reference

#### **Namespaces**

- · experimental
- internal

#### 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 Init

const Singleton class that performs additional initializations/cleanups

· class RandomDevices

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

class States

const Singleton class that implements most commonly used states

· class Timer

Measures time.

#### **Typedefs**

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

Complex number in double precision.

```
• template<typename Scalar >
```

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

Dynamic Eigen matrix over the field specified by Scalar.

```
\bullet \ \ \text{template}{<} \text{typename Scalar} >
```

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

Dynamic Eigen column vector over the field specified by Scalar.

```
• template<typename Scalar >
```

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

Dynamic Eigen row vector over the field specified by Scalar.

using ket = DynColVect< cplx >

Complex (double precision) dynamic Eigen column vector.

• using bra = DynRowVect< cplx >

Complex (double precision) dynamic Eigen row vector.

using cmat = DynMat< cplx >

Complex (double precision) dynamic Eigen matrix.

using dmat = DynMat< double >

Real (double precision) dynamic Eigen matrix.

#### **Functions**

constexpr std::complex< double > operator""\_i (unsigned long long int x)

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

constexpr std::complex< double > operator""\_i (long double x)

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

std::complex < double > omega (std::size t D)

D-th root of unity.

• template<typename Derived >

DynColVect< cplx > schmidtcoeff (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size\_t > &dims)

Schmidt coefficients of the bi-partite pure state A.

template<typename Derived >

cmat schmidtU (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size\_t > &dims)

Schmidt basis on Alice's side.

• template<typename Derived >

cmat schmidtV (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size t > &dims)

Schmidt basis on Bob's side.

• template<typename Derived >

Schmidt probabilities of the bi-partite pure state A.

template<typename Derived >

 $\label{lem:double entanglement} \mbox{double entanglement (const Eigen::MatrixBase< Derived > \&A, const std::vector< std::size\_t > \&dims)}$ 

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< std::size\_t > &dims)

Negativity of the bi-partite mixed state A.

• template<typename Derived >

double lognegativity (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size\_t > &dims)

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 shannon (const Eigen::MatrixBase< Derived > &A)

Shannon/von-Neumann entropy of the probability distribution/density matrix A.

• template<typename Derived >

double renyi (const Eigen::MatrixBase< Derived > &A, double alpha)

```
Renyi- \alpha entropy of the probability distribution/density matrix A, for \alpha \geq 0.
• template<typename Derived >
  double tsallis (const Eigen::MatrixBase< Derived > &A, double alpha)
      Tsallis- \alpha entropy of the probability distribution/density matrix A, for \alpha \geq 0

    template<typename Derived >

  double qmutualinfo (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size t > &subsysA,
  const std::vector< std::size_t > &subsysB, const std::vector< std::size_t > &dims)
     Quantum mutual information between 2 subsystems of a composite system.

    template<typename Derived >

  DynMat< typename Derived::Scalar > transpose (const Eigen::MatrixBase< Derived > &A)
      Transpose.
• template<typename Derived >
  DynMat< typename Derived::Scalar > conjugate (const Eigen::MatrixBase< Derived > &A)
     Complex conjugate.

    template<typename Derived >

  DynMat< typename Derived::Scalar > adjoint (const Eigen::MatrixBase< Derived > &A)
     Adjoint.
• template<typename Derived >
  DynMat< 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.
 \bullet \ \ \mathsf{template} \mathord{<} \mathsf{typename} \ \mathsf{Derived} >
  DynColVect< cplx > evals (const Eigen::MatrixBase< Derived > &A)
     Eigenvalues.

    template<typename Derived >

  cmat evects (const Eigen::MatrixBase< Derived > &A)
     Eigenvectors.
• template<typename Derived >
  DynColVect< double > hevals (const Eigen::MatrixBase< Derived > &A)
     Hermitian eigenvalues.

    template<typename Derived >

  cmat hevects (const Eigen::MatrixBase< Derived > &A)
     Hermitian eigenvectors.

    template<typename Derived >

  DynColVect< 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 absolut 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)
• 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 >

  DynMat< typename Derived::Scalar > powm (const Eigen::MatrixBase< Derived > &A, std::size_t n)
     Matrix power.
• template<typename Derived >
  double schatten (const Eigen::MatrixBase< Derived > &A, std::size_t p)
• template<typename OutputScalar , typename Derived >
  DynMat< OutputScalar > cwise (const Eigen::MatrixBase< Derived > &A, OutputScalar(*f)(const typename
  Derived::Scalar &))
     Functor.
template<typename T >
  DynMat< typename T::Scalar > kron (const T &head)
     Kronecker product (variadic overload)
• template<typename T , typename... Args>
  DynMat< typename T::Scalar > kron (const T &head, const Args &...tail)
     Kronecker product (variadic overload)

    template<typename Derived >

  DynMat< typename Derived::Scalar > kron (const std::vector< Derived > &As)
     Kronecker product (std::vector overload)

    template<typename Derived >

  DynMat< typename Derived::Scalar > kron (const std::initializer list< Derived > &As)
```

Kronecker product (std::initializer list overload)

 template<typename Derived > DynMat< typename Derived::Scalar > kronpow (const Eigen::MatrixBase< Derived > &A, std::size t n) Kronecker power. template<typename Derived > DynMat< typename Derived::Scalar > reshape (const Eigen::MatrixBase< Derived > &A, std::size t rows, std::size t cols) Reshape. • template<typename Derived1 , typename Derived2 > DynMat< typename Derived1::Scalar > comm (const Eigen::MatrixBase< Derived1 > &A, const Eigen::← MatrixBase< Derived2 > &B) Commutator. template<typename Derived1 , typename Derived2 > DynMat< typename Derived1::Scalar > anticomm (const Eigen::MatrixBase< Derived1 > &A, const Eigen← ::MatrixBase< Derived2 > &B) Anti-commutator. template<typename Derived > DynMat< typename Derived::Scalar > prj (const Eigen::MatrixBase< Derived > &V) Projector. template<typename Derived > DynMat< typename Derived::Scalar > grams (const std::vector< Derived > &Vs) Gram-Schmidt orthogonalization (std::vector overload) template<typename Derived > DynMat< typename Derived::Scalar > grams (const std::initializer\_list< Derived > &Vs) Gram-Schmidt orthogonalization (std::initializer\_list overload) template<typename Derived > DynMat< typename Derived::Scalar > grams (const Eigen::MatrixBase< Derived > &A) Gram-Schmidt orthogonalization (Eigen expression (matrix) overload) std::vector< std::size\_t > n2multiidx (std::size\_t n, const std::vector< std::size\_t > &dims) Non-negative integer index to multi-index. std::size t multiidx2n (const std::vector < std::size t > &midx, const std::vector < std::size t > &dims) Multi-index to non-negative integer index. ket mket (const std::vector< std::size\_t > &mask, const std::vector< std::size\_t > &dims) Multi-partite qudit ket (different dimensions overload) ket mket (const std::vector< std::size t > &mask, std::size t d=2) Multi-partite qudit ket (same dimensions overload) cmat mprj (const std::vector< std::size\_t > &mask, const std::vector< std::size\_t > &dims) Projector onto multi-partite qudit ket (different dimensions overload) cmat mprj (const std::vector < std::size\_t > &mask, std::size\_t d=2) Projector onto multi-partite qudit ket (same dimensions overload) std::vector< std::size\_t > invperm (const std::vector< std::size\_t > &perm) Inverse permutation. • std::vector< std::size t > compperm (const std::vector< std::size t > &perm, const std::vector< std::size t > &sigma) Compose permutations. template<typename InputIterator >  ${\tt std::vector} < {\tt double} > {\tt abssq} \ ({\tt InputIterator} \ {\sf first}, \ {\tt InputIterator} \ {\sf last})$ Computes the absolut values squared of a range of complex numbers. template<typename Derived >  ${\it std::} {\it vector} < {\it double} > {\it abssq} \; ({\it const} \; {\it Eigen::} {\it MatrixBase} < {\it Derived} > \& {\it V})$ Computes the absolut values squared of a column vector. template<typename InputIterator >

Element-wise sum of a range.

auto sum (InputIterator first, InputIterator last) -> typename InputIterator::value\_type

template<typename InputIterator >
 auto prod (InputIterator first, InputIterator last) -> typename InputIterator::value\_type

Element-wise product of a range.

template<typename Derived >

```
std::pair < 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::pair < 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 (std::initializer\_list overload)

template<typename Derived >

```
std::pair < std::vector < double >
```

, std::vector< cmat >> measure (const Eigen::MatrixBase< Derived > &A, const cmat &M)

Measures the state A in the orthonormal basis specified by the eigenvectors of M.

• template<typename InputIterator >

void disp (const InputIterator &first, const InputIterator &last, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

Displays a range. Does not add a newline.

template<typename InputIterator >

std::ostream & displn (const InputIterator &first, const InputIterator &last, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

Displays a range. Adds a newline.

• template<typename T >

std::ostream & disp (const T &x, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

Displays a standard container that supports std::begin, std::end and forward iteration. Does not add a newline.

• template<typename T >

std::ostream & displn (const T &x, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

Displays a standard container that supports std::begin, std::end and forward iteration. Adds a newline.

• template<typename T >

std::ostream & disp (const T \*x, const std::size\_t n, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

Displays a C-style array. Does not add a newline.

 $\bullet \ \ template {<} typename \ T >$ 

std::ostream & displn (const T \*x, const std::size\_t n, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

Displays a C-style array. Adds a newline.

• template<typename Derived >

std::ostream & disp (const Eigen::MatrixBase< Derived > &A, double chop=qpp::chop, std::ostream &os=std::cout)

Displays an Eigen expression in matrix friendly form. Does not add a new line.

template<typename Derived >

std::ostream & displn (const Eigen::MatrixBase< Derived > &A, double chop=qpp::chop, std::ostream &os=std::cout)

Displays an Eigen expression in matrix friendly form. Adds a newline.

• std::ostream & disp (const cplx z, double chop=qpp::chop, std::ostream &os=std::cout)

Displays a number (implicitly converted to std::complex<double>) in friendly form. Does not add a new line.

• std::ostream & displn (const cplx z, double chop=qpp::chop, std::ostream &os=std::cout)

Displays a number (implicitly converted to std::complex<double>) in friendly form. Adds a new line.

• 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 >

DynMat< typename Derived::Scalar > load (const std::string &fname)

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

template<typename Derived >

Derived loadMATLABmatrix (const std::string &mat\_file, const std::string &var\_name)

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 > &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, 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)

template<typename Derived1 , typename Derived2 >

DynMat< typename Derived1::Scalar > applyCTRL (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< std::size\_t > &ctrl, const std::vector< std::size\_t > &subsys, std::size\_t n, std::size\_t d=2)

Applies the controlled-gate A to the part subsys of a multipartite state vector or density matrix.

- template < typename Derived 1 , typename Derived 2 >

DynMat< typename Derived1::Scalar > apply (const Eigen::MatrixBase< Derived1 > &state, const Eigen 
 ::MatrixBase< Derived2 > &A, const std::vector< std::size\_t > &subsys, std::size\_t n, std::size\_t d=2)

Applies the gate A to the part subsys of a multipartite state vector or density matrix.

 $\bullet \ \ \text{template}{<} \text{typename Derived} >$ 

```
cmat channel (const Eigen::MatrixBase< Derived > &rho, const std::vector< cmat > &Ks)
```

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

template<typename Derived >

cmat channel (const Eigen::MatrixBase< Derived > &rho, const std::vector< cmat > &Ks, const std::vector<
std::size t > &subsys, std::size t n, std::size t d=2)

Applies the channel specified by the set of Kraus operators Ks to the part of the density matrix rho specified by subsys.

cmat super (const std::vector < cmat > &Ks)

Superoperator matrix representation.

cmat choi (const std::vector< cmat > &Ks)

Choi matrix representation.

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

Extracts orthogonal Kraus operators from Choi matrix.

template<typename Derived >

DynMat< typename Derived::Scalar > ptrace1 (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size t > &dims)

Partial trace.

template<typename Derived >

DynMat< typename Derived::Scalar > ptrace2 (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size t > &dims)

Partial trace.

template<typename Derived >

DynMat< typename Derived::Scalar > ptrace (const Eigen::MatrixBase< Derived > &A, const std::vector<
std::size\_t > &subsys, const std::vector< std::size\_t > &dims)

Partial trace.

• template<typename Derived >

DynMat< typename Derived::Scalar > ptranspose (const Eigen::MatrixBase< Derived > &A, const std
::vector< std::size t > &subsys, const std::vector< std::size t > &dims)

Partial transpose.

• template<typename Derived >

DynMat< typename Derived::Scalar > syspermute (const Eigen::MatrixBase< Derived > &A, const std
::vector< std::size\_t > &perm, const std::vector< std::size\_t > &dims)

System permutation.

template<typename Derived >

Derived rand (std::size t rows, std::size t cols, double a=0, double b=1)

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

template<>

dmat rand (std::size\_t rows, std::size\_t 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 (std::size t rows, std::size t 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)

• double rand (double a=0, double b=1)

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

int randint (int a=std::numeric\_limits< int >::min(), int b=std::numeric\_limits< int >::max())

Generates a random integer (int) uniformly distributed in the interval [a, b].

 $\bullet \ \ \text{template}{<} \text{typename Derived}>$ 

Derived randn (std::size\_t rows, std::size\_t cols, double mean=0, double sigma=1)

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

template<>

dmat randn (std::size\_t rows, std::size\_t cols, double mean, double sigma)

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

template<>

cmat randn (std::size\_t rows, std::size\_t cols, double mean, double sigma)

Generates a random complex matrix with entries (both real and imaginary) normally distributed in N(mean, 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(mean, sigma)

• cmat randU (std::size\_t D)

Generates a random unitary matrix.

cmat randV (std::size\_t Din, std::size\_t Dout)

Generates a random isometry matrix.

• std::vector < cmat > randkraus (std::size\_t N, std::size\_t D)

Generates a set of random Kraus operators.

cmat randH (std::size\_t D)

Generates a random Hermitian matrix.

ket randket (std::size\_t D)

Generates a random normalized ket (pure state vector)

cmat randrho (std::size\_t D)

Generates a random density matrix.

std::vector< std::size\_t > randperm (std::size\_t n)

Generates a random uniformly distributed permutation.

#### **Variables**

constexpr double chop = 1e-10

Used in qpp::disp() and qpp::displn() for setting to zero numbers that have their absolute value smaller than qpp::ct← ::chop.

• constexpr double eps = 1e-12

Used to decide whether a number or expression in double precision is zero or not.

constexpr std::size\_t maxn = 64

Maximum number of qubits.

constexpr double pi = 3.141592653589793238462643383279502884

 $\pi$ 

constexpr double ee = 2.718281828459045235360287471352662497

Base of natural logarithm, e.

constexpr std::size\_t infty = -1

Used to denote infinity.

• const Init & init = Init::get\_instance()

qpp::Init const Singleton

const Codes & codes = Codes::get\_instance()

qpp::Codes const Singleton

const Gates & gt = Gates::get instance()

qpp::Gates const Singleton

const States & st = States::get\_instance()

qpp::States const Singleton

RandomDevices & rdevs = RandomDevices::get\_instance()

qpp::RandomDevices Singleton

#### 6.1.1 Typedef Documentation

6.1.1.1 using qpp::bra = typedef DynRowVect<cplx>

Complex (double precision) dynamic Eigen row vector.

6.1.1.2 using qpp::cmat = typedef DynMat<cplx>

Complex (double precision) dynamic Eigen matrix.

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

Complex number in double precision.

6.1.1.4 using qpp::dmat = typedef DynMat<double>

Real (double precision) dynamic Eigen matrix.

6.1.1.5 template < typename Scalar > using qpp::DynColVect = typedef Eigen::Matrix < Scalar, Eigen::Dynamic, 1>

Dynamic Eigen column vector over the field specified by Scalar.

Example:

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

6.1.1.6 template<typename Scalar > using qpp::DynMat = typedef Eigen::Matrix<Scalar, Eigen::Dynamic, Eigen::Dynamic>

Dynamic Eigen matrix over the field specified by Scalar.

Example:

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

6.1.1.7 template < typename Scalar > using qpp::DynRowVect = typedef Eigen::Matrix < Scalar, 1, Eigen::Dynamic >

Dynamic Eigen row vector over the field specified by Scalar.

Example:

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

6.1.1.8 using qpp::ket = typedef DynColVect<cplx>

Complex (double precision) dynamic Eigen column vector.

- 6.1.2 Function Documentation
- 6.1.2.1 template < typename Derived > cmat qpp::absm ( const Eigen::MatrixBase < Derived > & A )

Matrix absolut value.

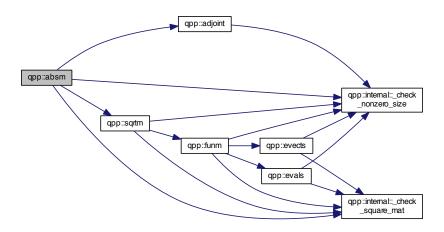
**Parameters** 

Α	Eigen expression	
---	------------------	--

#### Returns

Matrix absolut value of A

Here is the call graph for this function:



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

Computes the absolut values squared of a range of complex numbers.

#### **Parameters**

first	Iterator to the first element of the range
last	Iterator to the last element of the range

#### Returns

Real vector consisting of the range's absolut values squared

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

Computes the absolut values squared of a column vector.

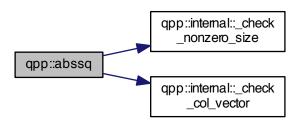
#### **Parameters**

V	Eigen expression

#### Returns

Real vector consisting of the absolut values squared

Here is the call graph for this function:



6.1.2.4 template<typename Derived > DynMat<typename Derived::Scalar> qpp::adjoint ( const Eigen::MatrixBase< Derived > & A )

#### Adjoint.

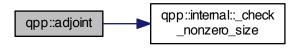
#### **Parameters**

Α	Eigen expression

#### Returns

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

Here is the call graph for this function:



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

#### Anti-commutator.

Anti-commutator  $\{A,B\} = AB + BA$ 

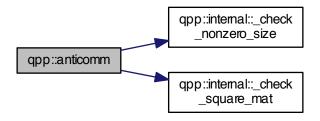
Both A and B must be Eigen expressions over the same scalar field

Α	Eigen expression
В	Eigen expression

#### Returns

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

Here is the call graph for this function:



6.1.2.6 template<typename Derived1 , typename Derived2 > DynMat<typename Derived1::Scalar> qpp::apply ( const Eigen::MatrixBase< Derived1 > & state, const Eigen::MatrixBase< Derived2 > & A, const std::vector< std::size\_t > & subsys, std::size\_t n, std::size\_t d = 2)

Applies the gate A to the part subsys of a multipartite state vector or density matrix.

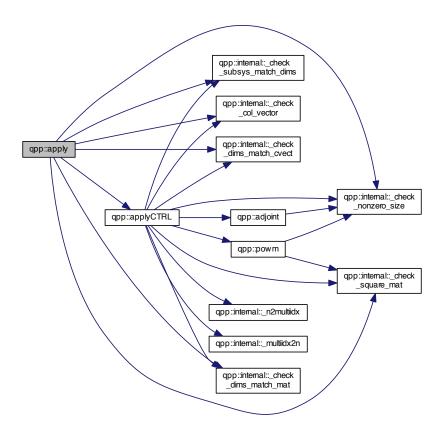
# Note

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

state	Eigen expression
Α	Eigen expression
subsys	Subsystem indexes where the gate A is applied
n	Total number of subsystems
d	Local dimensions of all local Hilbert spaces (must all be equal)

Gate A applied to the part subsys of state

Here is the call graph for this function:



6.1.2.7 template<typename Derived1 , typename Derived2 > DynMat<typename Derived1::Scalar> qpp::applyCTRL ( const Eigen::MatrixBase< Derived1 > & state, const Eigen::MatrixBase< Derived2 > & A, const std::vector< std::size\_t > & ctrl, const std::vector< std::size\_t > & subsys, std::size\_t d = 2)

Applies the controlled-gate A to the part subsys of a multipartite state vector or density matrix.

# Note

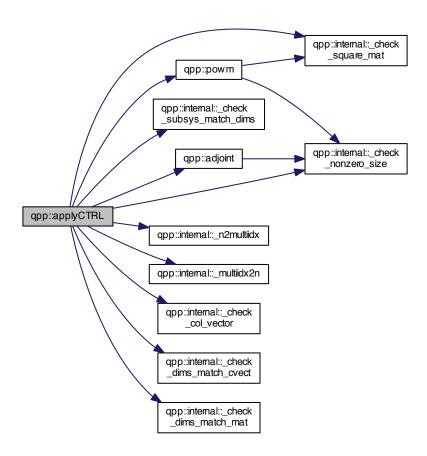
The dimension of the gate A must match the dimension of subsys

state	Eigen expression
Α	Eigen expression
ctrl	Control subsystem indexes
subsys	Subsystem indexes where the gate A is applied

n	Total number of subsystems
d	Local dimensions of all local Hilbert spaces (must all be equal)

CTRL-A gate applied to the part subsys of state

Here is the call graph for this function:



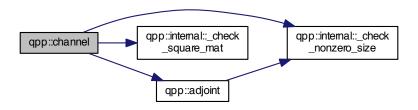
6.1.2.8 template<typename Derived > cmat qpp::channel ( const Eigen::MatrixBase< Derived > & rho, const std::vector< cmat > & Ks )

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

rho	Eigen expression
Ks	Set of Kraus operators

Output density matrix after the action of the channel

Here is the call graph for this function:



6.1.2.9 template<typename Derived > cmat qpp::channel ( const Eigen::MatrixBase< Derived > & rho, const std::vector< cmat > & Ks, const std::vector< std::size\_t > & subsys, std::size\_t n, std::size\_t d = 2 )

Applies the channel specified by the set of Kraus operators *Ks* to the part of the density matrix *rho* specified by *subsys*.

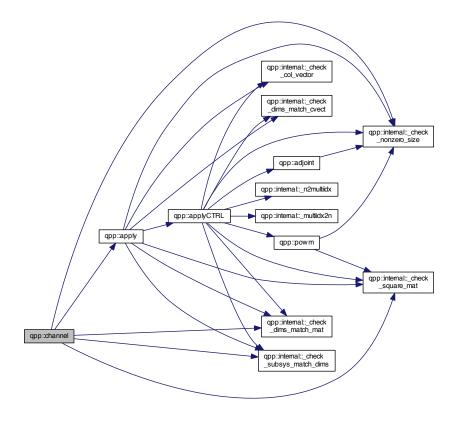
## **Parameters**

rho	Eigen expression
Ks	Set of Kraus operators
subsys	Subsystems' indexes
n	Total number of subsystems
d	Local dimensions of all local Hilbert spaces (must all be equal)

# Returns

Output density matrix after the action of the channel

Here is the call graph for this function:



# 6.1.2.10 cmat qpp::choi ( const std::vector < cmat > & Ks )

Choi matrix representation.

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

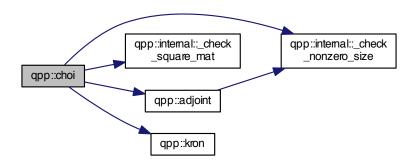
# Note

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

Ks	Set of Kraus operators

Choi matrix representation

Here is the call graph for this function:



6.1.2.11 std::vector<cmat> qpp::choi2kraus ( const cmat & A )

Extracts orthogonal Kraus operators from Choi matrix.

Extracts a set of orthogonal (under Hilbert-Schmidt operator norm) Kraus operators from the Choi representation *A* of the channel

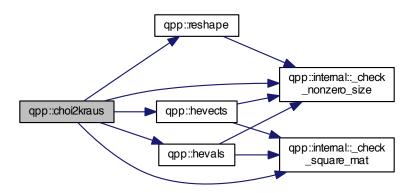
Note

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

Α	Choi matrix
---	-------------

Set of Kraus operators

Here is the call graph for this function:



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

Commutator.

Commutator [A,B] = AB - BA

Both A and B must be Eigen expressions over the same scalar field

#### Parameters

Α	Eigen expression
В	Eigen expression

# Returns

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

Here is the call graph for this function:



6.1.2.13 std::vector < std::size\_t > & perm, const std::vector < std::size\_t > & perm, const std::vector < std::size\_t > & sigma )

Compose permutations.

perm	Permutation
sigma	Permutation

## Returns

Composition of the permutations *perm* o *sigma* = perm(sigma)

Here is the call graph for this function:



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

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

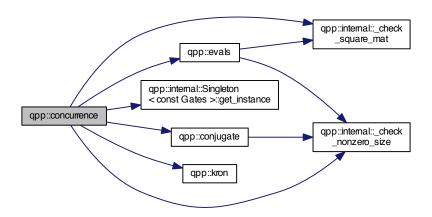
## **Parameters**

Α	Eigen expression
---	------------------

# Returns

Wootters concurrence

Here is the call graph for this function:



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

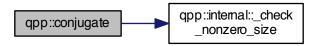
Complex conjugate.

Α	Eigen expression
---	------------------

## Returns

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

Here is the call graph for this function:



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

# Matrix cos.

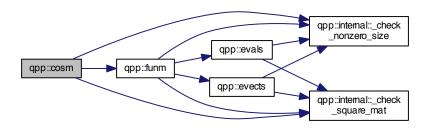
## **Parameters**

А	Figen expression
/1	Ligen expression

# Returns

Matrix cosine of A

Here is the call graph for this function:



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

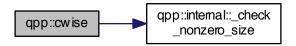
Functor.

Α	Eigen expression
f	Pointer-to-function from scalars of A to OutputScalar

#### Returns

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

Here is the call graph for this function:



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

#### Determinant.

## **Parameters**

Α	Eigen expression
---	------------------

# Returns

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

Here is the call graph for this function:



6.1.2.19 template<typename InputIterator > void qpp::disp ( const InputIterator & first, const InputIterator & last, const std::string & separator, const std::string & start = " [ ", const std::string & end = " ] ", std::ostream & os = std::cout )

Displays a range. Does not add a newline.

#### See also

qpp::displn()

first	Iterator to the first element of the range
last	Iterator to the last element of the range
separator	Separator
start	Left marking
end	Right marking
os	Output stream

#### Returns

Output stream

6.1.2.20 template<typename T > std::ostream& qpp::disp ( const T & x, const std::string & separator, const std::string & start = " [ ", const std::string & end = " ] ", std::ostream & os = std::cout )

Displays a standard container that supports std::begin, std::end and forward iteration. Does not add a newline.

# See also

qpp::displn()

## **Parameters**

X	Container
separator	Separator
start	Left marking
end	Right marking
os	Output stream

## Returns

Output stream

Here is the call graph for this function:



6.1.2.21 template<typename T > std::ostream& qpp::disp ( const T \* x, const std::size\_t n, const std::string & separator, const std::string & start = " [ ", const std::string & end = " ] ", std::ostream & os = std::cout )

Displays a C-style array. Does not add a newline.

# See also

qpp::displn()

X	Pointer to the first element
n	Number of elements to be displayed
separator	Separator
start	Left marking
end	Right marking
os	Output stream

# Returns

Output stream

6.1.2.22 template < typename Derived > std::ostream& qpp::disp ( const Eigen::MatrixBase < Derived > & A, double chop = qpp::chop, std::ostream & os = std::cout )

Displays an Eigen expression in matrix friendly form. Does not add a new line.

# See also

qpp::displn()

# **Parameters**

Α	Eigen expression
chop	Set to zero the elements smaller in absolute value than <i>chop</i>
os	Output stream

## Returns

Output stream

6.1.2.23 std::ostream& qpp::disp ( const cplx z, double chop = qpp::chop, std::ostream & os = std::cout )

Displays a number (implicitly converted to std::complex<double>) in friendly form. Does not add a new line.

## See also

qpp::displn()

Z	Real/complex number
chop	Set to zero the elements smaller in absolute value than <i>chop</i>
os	Output stream

Output stream

Here is the call graph for this function:



6.1.2.24 template < typename InputIterator > std::ostream & qpp::displn ( const InputIterator & first, const InputIterator & last, const std::string & separator, const std::string & start = " [ ", const std::string & end = " ] ", std::ostream & os = std::cout )

Displays a range. Adds a newline.

See also

qpp::disp()

#### **Parameters**

first	Iterator to the first element of the range
last	Iterator to the last element of the range
separator	Separator
start	Left marking
end	Right marking
os	Output stream

#### Returns

Output stream

Here is the call graph for this function:



6.1.2.25 template<typename T > std::ostream& qpp::displn ( const T & x, const std::string & separator, const std::string & start = " [ ", const std::string & end = " ] ", std::ostream & os = std::cout )

Displays a standard container that supports std::begin, std::end and forward iteration. Adds a newline.

# See also

qpp::disp()

## **Parameters**

X	Container
separator	Separator
start	Left marking
end	Right marking
os	Output stream

## Returns

Output stream

Here is the call graph for this function:



6.1.2.26 template < typename T > std::ostream& qpp::displn ( const T \* x, const std::size\_t n, const std::string & separator, const std::string & start = " [ ", const std::string & end = " ] ", std::ostream & os = std::cout )

Displays a C-style array. Adds a newline.

## See also

qpp::disp()

X	Pointer to the first element
n	Number of elements to be displayed
separator	Separator
start	Left marking
end	Right marking
os	Output stream

Output stream

Here is the call graph for this function:



6.1.2.27 template<typename Derived > std::ostream& qpp::displn ( const Eigen::MatrixBase< Derived > & A, double chop = qpp::chop, std::ostream & os = std::cout )

Displays an Eigen expression in matrix friendly form. Adds a newline.

See also

qpp::disp()

#### **Parameters**

Α	Eigen expression
chop	Set to zero the elements smaller in absolute value than <i>chop</i>
os	Output stream

#### Returns

Output stream

Here is the call graph for this function:



6.1.2.28 std::ostream& qpp::displn ( const cplx z, double chop = qpp::chop, std::ostream & os = std::cout )

Displays a number (implicitly converted to std::complex<double>) in friendly form. Adds a new line.

See also

qpp::disp()

Z	Real/complex number
chop	Set to zero the elements smaller in absolute value than <i>chop</i>
os	Output stream

# Returns

Output stream

Here is the call graph for this function:



6.1.2.29 template < typename Derived > double qpp::entanglement ( const Eigen::MatrixBase < Derived > & A, const std::vector < std::size\_t > & 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

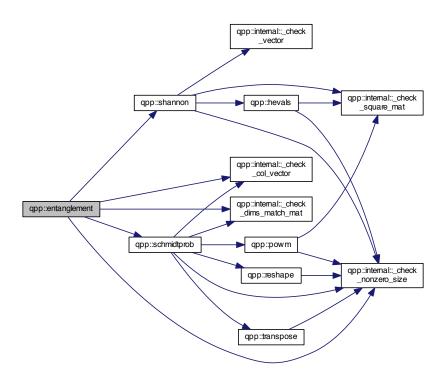
# See also

qpp::shannon()

Α	Eigen expression
dims	Subsystems' dimensions

Entanglement, with the logarithm in base 2

Here is the call graph for this function:



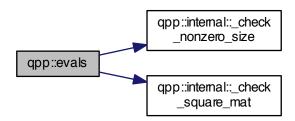
6.1.2.30 template<typename Derived > DynColVect<cplx> qpp::evals ( const Eigen::MatrixBase< Derived > & A )

Eigenvalues.

Α	Eigen expression

Eigenvalues of A, as a complex dynamic column vector

Here is the call graph for this function:



6.1.2.31 template<typename Derived > cmat qpp::evects ( const Eigen::MatrixBase< Derived > & A )

# Eigenvectors.

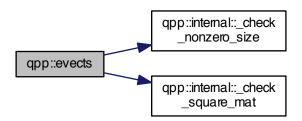
#### **Parameters**

Λ	Figure assessing
A	Eigen expression

## Returns

Eigenvectors of A, as columns of a complex matrix

Here is the call graph for this function:



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

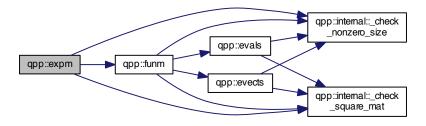
Matrix exponential.

Α	Eigen expression
---	------------------

## Returns

Matrix exponential of A

Here is the call graph for this function:



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

# Functional calculus f(A)

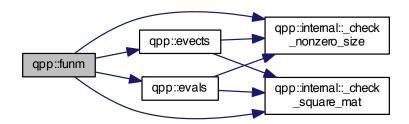
## **Parameters**

Α	Eigen expression
f	Pointer-to-function from complex to complex

# Returns

f(A)

Here is the call graph for this function:



6.1.2.34 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()

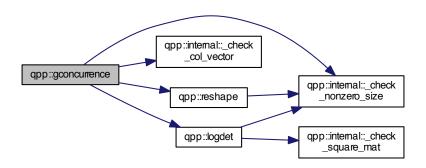
**Parameters** 

Α	Eigen expression

Returns

G-concurrence

Here is the call graph for this function:



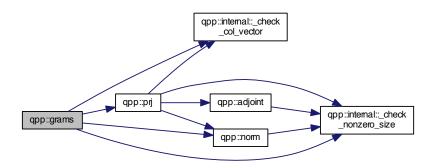
6.1.2.35 template<typename Derived > DynMat<typename Derived::Scalar> qpp::grams ( const std::vector< Derived > & Vs )

Gram-Schmidt orthogonalization (std::vector overload)

Vs	std::vector of Eigen expressions as column vectors
----	--

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

Here is the call graph for this function:



6.1.2.36 template<typename Derived > DynMat<typename Derived::Scalar> qpp::grams ( const std::initializer\_list< Derived > & Vs)

Gram-Schmidt orthogonalization (std::initializer\_list overload)

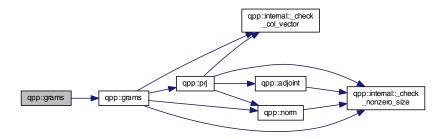
# **Parameters**

Vs	std::initializer_list of Eigen expressions as column vectors

# Returns

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

Here is the call graph for this function:



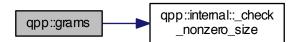
6.1.2.37 template<typename Derived > DynMat<typename Derived::Scalar> qpp::grams ( const Eigen::MatrixBase< Derived > & A )

Gram-Schmidt orthogonalization (Eigen expression (matrix) overload)

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

## Returns

Gram-Schmidt vectors of the columns of *A*, as columns of a dynamic matrix over the same scalar field as *A*Here is the call graph for this function:



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

Hermitian eigenvalues.

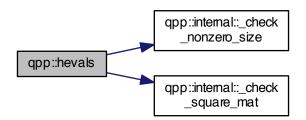
#### **Parameters**

Α	Eigen expression

# Returns

Eigenvalues of Hermitian A, as a real dynamic column vector

Here is the call graph for this function:



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

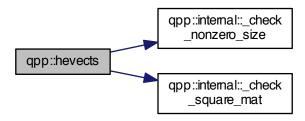
Hermitian eigenvectors.

Α	Eigen expression
---	------------------

## Returns

Eigenvectors of Hermitian A, as columns of a complex matrix

Here is the call graph for this function:



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

Inverse.

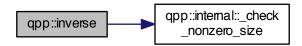
# **Parameters**

Α	Eigen expression

# Returns

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

Here is the call graph for this function:



6.1.2.41 std::vector<std::size\_t> qpp::invperm ( const std::vector< std::size\_t > & perm )

Inverse permutation.

perm	Permutation
------	-------------

# Returns

Inverse of the permutation perm

Here is the call graph for this function:



6.1.2.42 template<typename T > DynMat<typename T::Scalar> qpp::kron ( const T & head )

Kronecker product (variadic overload)

Used to stop the recursion for the variadic template version of qpp::kron()

# **Parameters**

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

# Returns

Its argument head

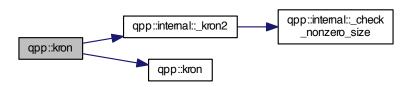
6.1.2.43 template<typename T , typename... Args> DynMat<typename T::Scalar> qpp::kron ( const T & head, const Args &... tail )

Kronecker product (variadic overload)

head	Eigen expression
tail	Variadic Eigen expression (zero or more parameters)

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

Here is the call graph for this function:



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

Kronecker product (std::vector overload)

## **Parameters**

As	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

Here is the call graph for this function:



6.1.2.45 template < typename Derived > DynMat < typename Derived::Scalar > qpp::kron ( const std::initializer\_list < Derived > & As )

Kronecker product (std::initializer\_list overload)

As	std::initializer_list of Eigen expressions, such as {A1, A2, ,Ak}

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

Here is the call graph for this function:



6.1.2.46 template<typename Derived > DynMat<typename Derived::Scalar> qpp::kronpow ( const Eigen::MatrixBase< Derived > & A, std::size\_t n )

Kronecker power.

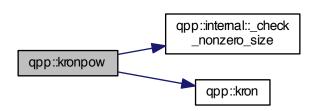
#### **Parameters**

Α	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

Here is the call graph for this function:



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

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

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");
```

#### See also

gpp::loadMATLABmatrix()

# **Parameters**

Α	Eigen expression
fname	Output file name

6.1.2.48 template < typename Derived > Derived qpp::loadMATLABmatrix ( const std::string & mat\_file, const std::string & var\_name )

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

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)

6.1.2.49 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)

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**

mat_file	MATALB .mat file
var_name	Variable name in the .mat file representing the matrix to be loaded

# Returns

Eigen double dynamic matrix (qpp::dmat)

6.1.2.50 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)

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<cmat>("input.mat");
```

mat_file	MATALB .mat file
var name	Variable name in the .mat file representing the matrix to be loaded

## Returns

Eigen complex dynamic matrix (qpp::cmat)

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

Logarithm of the determinant.

Especially useful when the determinant overflows/underflows

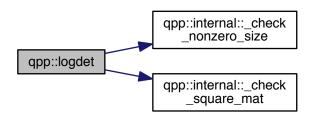
#### **Parameters**

Α	Eigen expression

# Returns

Logarithm of the determinant of A, as a scalar in the same scalar field as A

Here is the call graph for this function:



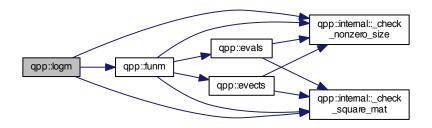
6.1.2.52 template < typename Derived > cmat qpp::logm ( const Eigen::MatrixBase < Derived > & A )

Matrix logarithm.

Α	Eigen expression

Matrix logarithm of A

Here is the call graph for this function:



6.1.2.53 template < typename Derived > double qpp::lognegativity ( const Eigen::MatrixBase < Derived > & A, const std::vector < std::size\_t > & dims )

Logarithmic negativity of the bi-partite mixed state A.

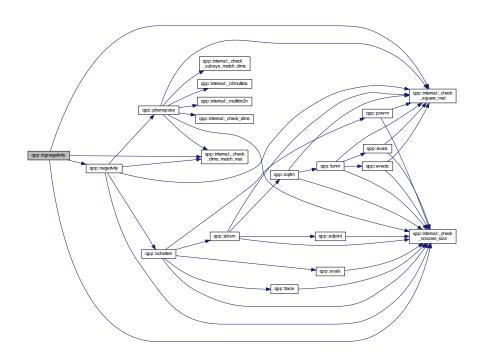
# **Parameters**

Α	Eigen expression
dims	Subsystems' dimensions

## Returns

Logarithmic negativity, with the logarithm in base 2

Here is the call graph for this function:



6.1.2.54 template < typename Derived > std::pair < 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.

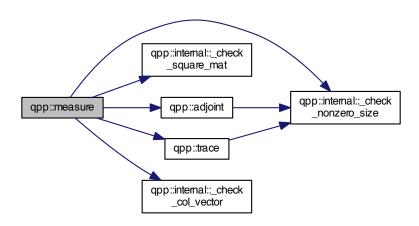
# **Parameters**

Α	Eigen expression
Ks	Set of Kraus operators

# Returns

Pair of vector of probabilities and vector of post-measurement normalized states

Here is the call graph for this function:

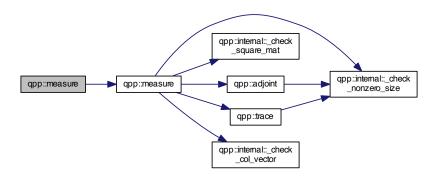


Measures the state A using the set of Kraus operators Ks (std::initializer\_list overload)

Α	Eigen expression
Ks	Set of Kraus operators

Pair of vector of probabilities and vector of post-measurement normalized states

Here is the call graph for this function:



6.1.2.56 template < typename Derived > std::pair < std::vector < double > , std::vector < cmat > > qpp::measure ( const Eigen::MatrixBase < Derived > & A, const cmat & M)

Measures the state A in the orthonormal basis specified by the eigenvectors of M.

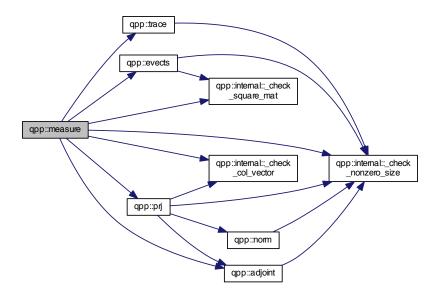
# **Parameters**

Α	Eigen expression
М	Normal matrix whose eigenvectors define the measurement basis

# Returns

Pair of vector of probabilities and vector of post-measurement normalized states

Here is the call graph for this function:



6.1.2.57 ket qpp::mket ( const std::vector < std::size\_t > & mask, const std::vector < std::size\_t > & dims )

Multi-partite qudit ket (different dimensions overload)

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

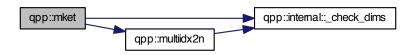
# Parameters

mask	std::vector of non-negative integers
dims	Dimensions of the multi-partite system

## Returns

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

Here is the call graph for this function:



6.1.2.58 ket qpp::mket ( const std::vector < std::size\_t > & mask, std::size\_t d = 2 )

Multi-partite qudit ket (same dimensions overload)

Constructs the multi-partite qudit ket  $|mask\rangle$ , 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

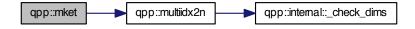
#### **Parameters**

mask	std::vector of non-negative integers
d	Subsystems' dimension

## Returns

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

Here is the call graph for this function:



6.1.2.59 cmat qpp::mprj ( const std::vector< std::size\_t > & mask, const std::vector< std::size\_t > & dims )

Projector onto multi-partite qudit ket (different dimensions overload)

Constructs the projector onto the multi-partite qudit ket  $|mask\rangle$ , 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	std::vector of non-negative integers
dims	Dimensions of the multi-partite system

## Returns

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

Here is the call graph for this function:



6.1.2.60 cmat qpp::mprj ( const std::vector< std::size\_t > & mask, std::size\_t d = 2 )

Projector onto multi-partite qudit ket (same dimensions overload)

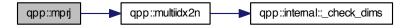
Constructs the projector onto the multi-partite qudit ket  $|mask\rangle$ , all subsystem having equal dimension d mask is a std::vector of non-negative integers, and each element in mask has to be strictly smaller than d

mask	std::vector of non-negative integers
d	Subsystems' dimension

# Returns

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

Here is the call graph for this function:



6.1.2.61 std::size\_t qpp::multiidx2n ( const std::vector < std::size\_t > & midx, const std::vector < std::size\_t > & dims )

Multi-index to non-negative integer index.

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

#### **Parameters**

midx	Multi-index
dims	Dimensions of the multi-partite system

## Returns

Non-negative integer index

Here is the call graph for this function:



6.1.2.62 std::vector<std::size\_t> qpp::n2multiidx ( std::size\_t n, const std::vector< std::size\_t > & dims )

Non-negative integer index to multi-index.

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

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

# Returns

Multi-index of the same size as dims

Here is the call graph for this function:



6.1.2.63 template < typename Derived > double qpp::negativity ( const Eigen::MatrixBase < Derived > & A, const std::vector < std::size\_t > & dims )

Negativity of the bi-partite mixed state A.

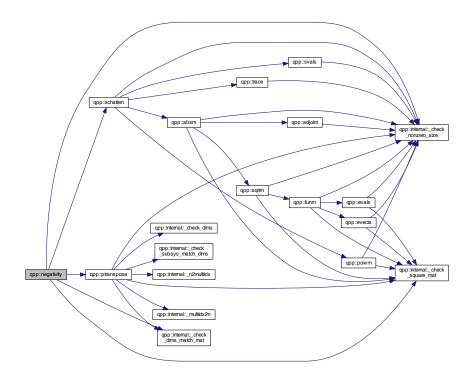
## **Parameters**

Α	Eigen expression
dims	Subsystems' dimensions

# Returns

Negativity

Here is the call graph for this function:



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

Frobenius norm.

### **Parameters**

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

## Returns

Frobenius norm of A, as a real number

Here is the call graph for this function:



6.1.2.65 std::complex<double> qpp::omega ( std::size\_t D )

D-th root of unity.

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

#### Returns

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

6.1.2.66 constexpr std::complex<double> qpp::operator""\_i ( unsigned long long int x )

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

# Example:

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

6.1.2.67 constexpr std::complex<double> qpp::operator""\_i ( long double x )

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

### Example:

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

6.1.2.68 template<typename Derived > DynMat<typename Derived::Scalar> qpp::powm ( const Eigen::MatrixBase< Derived > & A, std::size\_t n )

Matrix power.

Explicitly multiplies the matrix A with itself n times

By convention  $A^0 = I$ 

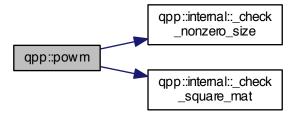
### **Parameters**

Α	Eigen expression
n	Non-negative integer

### Returns

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

Here is the call graph for this function:



6.1.2.69 template < typename Derived > DynMat < typename Derived::Scalar > qpp::prj ( const Eigen::MatrixBase < Derived > & V )

Projector.

Normalized projector onto state vector

**Parameters** 

V	Eigen expression
•	ge., exp. ese.e

## Returns

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

Here is the call graph for this function:



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

Element-wise product of A.

**Parameters** 

Α	Eigen expression

# Returns

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

Here is the call graph for this function:



6.1.2.71 template<typename InputIterator > auto qpp::prod ( InputIterator first, InputIterator last ) -> typename InputIterator::value\_type

Element-wise product of a range.

#### **Parameters**

first	Iterator to the first element of the range
last	Iterator to the last element of the range

## Returns

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

6.1.2.72 template < typename Derived > DynMat < typename Derived::Scalar > qpp::ptrace ( const Eigen::MatrixBase < Derived > & A, const std::vector < std::size\_t > & subsys, const std::vector < std::size\_t > & dims )

## Partial trace.

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

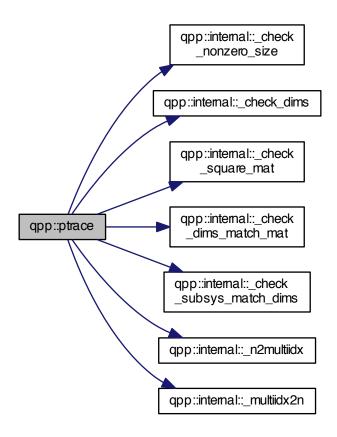
## Parameters

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

### Returns

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

Here is the call graph for this function:



6.1.2.73 template < typename Derived > DynMat < typename Derived::Scalar > qpp::ptrace1 ( const Eigen::MatrixBase < Derived > & A, const std::vector < std::size\_t > & dims )

Partial trace.

Partial trace of density matrix over the first subsystem in a bi-partite system

Α	Eigen expression
dims	Dimensions of bi-partite system (must be a std::vector with 2 elements)

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

Here is the call graph for this function:



6.1.2.74 template < typename Derived > DynMat < typename Derived::Scalar > qpp::ptrace2 ( const Eigen::MatrixBase < Derived > & A, const std::vector < std::size\_t > & dims)

# Partial trace.

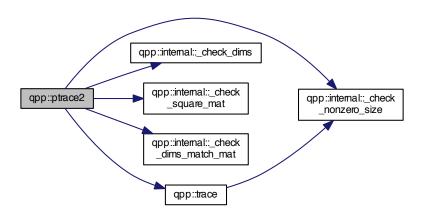
# **Parameters**

Α	Eigen expression
dims	Dimensions of bi-partite system (must be a std::vector with 2 elements)

### 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

Here is the call graph for this function:



# Partial transpose.

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

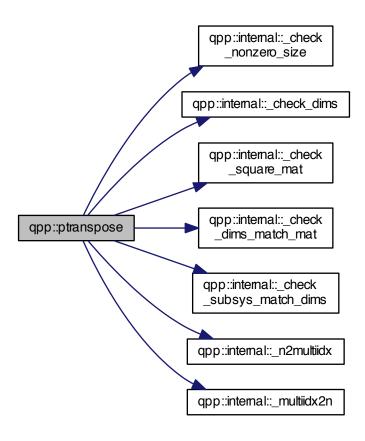
### **Parameters**

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

### Returns

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

Here is the call graph for this function:



6.1.2.76 template < typename Derived > double qpp::qmutualinfo ( const Eigen::MatrixBase < Derived > & A, const std::vector < std::size\_t > & subsysB, const std::vector < std::size\_t > & subsysB, const std::vector < std::size\_t > & dims )

Quantum mutual information between 2 subsystems of a composite system.

Α	Eigen expression
subsysA	Indexes of the first subsystem
subsysB	Indexes of the second subsystem
dims	Subsystems' dimensions

Mutual information between the 2 subsystems

Here is the call graph for this function:



6.1.2.77 template < typename Derived > Derived qpp::rand ( std::size\_t rows, std::size\_t 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* 

6.1.2.78 template <> dmat qpp::rand ( std::size\_t rows, std::size\_t 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*)

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);
```

rows	Number of rows of the random generated matrix
cols	Number of columns of the random generated matrix
а	Beginning of the interval, belongs to it
b	End of the interval, does not belong to it

### Returns

Random real matrix

6.1.2.79 template <> cmat qpp::rand ( std::size\_t rows, std::size\_t 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*)

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**

rows	Number of rows of the random generated matrix
cols	Number of columns of the random generated matrix
а	Beginning of the interval, belongs to it
b	End of the interval, does not belong to it

### Returns

Random complex matrix

Here is the call graph for this function:



6.1.2.80 double qpp::rand ( double a = 0, double b = 1 )

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

а	Beginning of the interval, belongs to it
b	End of the interval, does not belong to it

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

Here is the call graph for this function:



## 6.1.2.81 cmat qpp::randH ( std::size\_t D )

Generates a random Hermitian matrix.

### **Parameters**

D	Dimension of the Hilbert space

# Returns

Random Hermitian matrix

Here is the call graph for this function:



Generates a random integer (int) uniformly distributed in the interval [a, b].

а	Beginning of the interval, belongs to it
b	End of the interval, does not belong to it

## Returns

Random integer (int) uniformly distributed in the interval [a, b]

Here is the call graph for this function:



## 6.1.2.83 ket qpp::randket ( std::size\_t D )

Generates a random normalized ket (pure state vector)

#### **Parameters**

D	Dimension of the Hilbert space

### Returns

Random normalized ket

Here is the call graph for this function:



6.1.2.84 std::vector<cmat> qpp::randkraus ( std::size\_t N, std::size\_t D )

Generates a set of random Kraus operators.

Note

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

N	Number of Kraus operators
D	Dimension of the Hilbert space

#### Returns

Set of N Kraus operators satisfying the closure condition

Here is the call graph for this function:



6.1.2.85 template<typename Derived > Derived qpp::randn ( std::size\_t rows, std::size\_t 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* 

 $6.1.2.86 \quad template <> dmat\ qpp::randn\ (\ std::size\_t\ rows,\ std::size\_t\ cols,\ double\ \textit{mean},\ double\ \textit{sigma}\ )$ 

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);
```

rows	Number of rows of the random generated matrix
cols	Number of columns of the random generated matrix
mean	Mean
sigma	Standard deviation

Random real matrix

Here is the call graph for this function:



6.1.2.87 template <> cmat qpp::randn ( std::size\_t rows, std::size\_t cols, double mean, double sigma )

Generates a random complex matrix with entries (both real and imaginary) normally distributed in N(mean, 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**

rows	Number of rows of the random generated matrix
cols	Number of columns of the random generated matrix
mean	Mean
sigma	Standard deviation

# Returns

Random complex matrix

Here is the call graph for this function:



6.1.2.88 double qpp::randn ( double mean = 0, double sigma = 1 )

Generates a random real number (double) normally distributed in N(mean, sigma)

mean	Mean
sigma	Standard deviation

## Returns

Random real number normally distributed in N(mean, sigma)

Here is the call graph for this function:



6.1.2.89 std::vector<std::size\_t> qpp::randperm ( std::size\_t n )

Generates a random uniformly distributed permutation.

Uses Knuth's 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

Here is the call graph for this function:



6.1.2.90 cmat qpp::randrho ( std::size\_t D )

Generates a random density matrix.

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

### Returns

Random density matrix

6.1.2.91 cmat qpp::randU ( std::size\_t D )

Generates a random unitary matrix.

**Parameters** 

_		
	D	Dimension of the Hilbert space

# Returns

Random unitary

6.1.2.92 cmat qpp::randV ( std::size\_t Din, std::size\_t Dout )

Generates a random isometry matrix.

#### **Parameters**

Din	Size of the input Hilbert space
Dout	Size of the output Hilbert space

# Returns

Random isometry matrix

Here is the call graph for this function:



 $\textbf{6.1.2.93} \quad \textbf{template} < \textbf{typename Derived} > \textbf{double qpp::renyi} \, ( \, \, \textbf{const Eigen::MatrixBase} < \, \textbf{Derived} > \textbf{\&} \, \textbf{\textit{A}}, \, \, \textbf{double alpha} \, \, )$ 

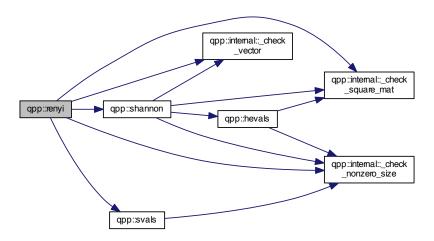
Renyi-  $\alpha$  entropy of the probability distribution/density matrix A, for  $\alpha \geq 0$ .

A	Eigen expression, representing a probability distribution (real dynamic column vector) or a
	density matrix (complex dynamic matrix)

- 1:- 1	Management to a second and the secon
ainna	Non-negative real number, use <i>qpp::infty</i> for $\alpha = \infty$
۵.۱۵.۱۵	rion negative real names, and apprintly to the

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

Here is the call graph for this function:



# Reshape.

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

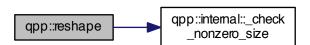
### **Parameters**

Α	Eigen expression
rows	Number of rows of the reshaped matrix
cols	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

Here is the call graph for this function:



6.1.2.95 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::saveMATLABmatrix()

#### **Parameters**

Α	Eigen expression
fname	Output file name

6.1.2.96 template<typename Derived > void qpp::saveMATLABmatrix ( const Eigen::MatrixBase< Derived > & A, const std::string & mat\_file, const std::string & mode )

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

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)

6.1.2.97 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)

#### **Parameters**

Α	Eigen expression over the complex field
mat_file	MATALB .mat file
var_name	Variable name in the .mat file representing the matrix to be saved
mode	Saving mode (append, overwrite etc.), see MATLAB's matOpen() documentation for details

Here is the call graph for this function:

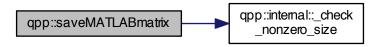


6.1.2.98 template<> void qpp::saveMATLABmatrix ( const Eigen::MatrixBase< cmat > & A, const std::string & mat\_file, const std::string & var\_name, const std::string & mode )

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

Α	Eigen expression over the complex field
mat_file	MATALB .mat file
var_name	Variable name in the .mat file representing the matrix to be saved
mode	Saving mode (append, overwrite etc.), see MATLAB's matOpen() documentation for details

Here is the call graph for this function:



6.1.2.99 template < typename Derived > double qpp::schatten ( const Eigen::MatrixBase < Derived > & A, std::size\_t p )

## Schatten norm.

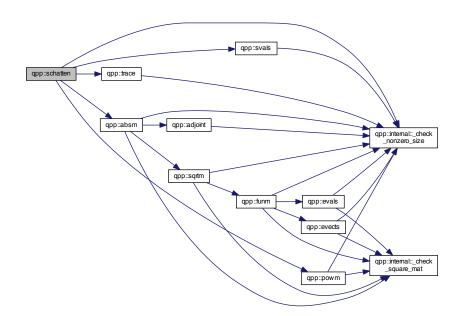
### **Parameters**

Α	Eigen expression
р	Integer, greater or equal to 1

### Returns

Schatten-p norm of A, as a real number

Here is the call graph for this function:



6.1.2.100 template<typename Derived > DynColVect<cplx> qpp::schmidtcoeff ( const Eigen::MatrixBase< Derived > & A, const std::vector< std::size\_t > & 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::schmidtprob()

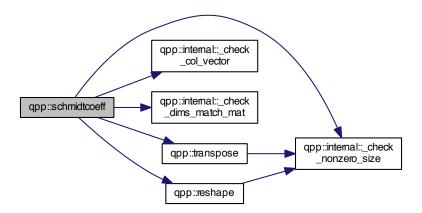
#### **Parameters**

Α	Eigen expression
dims	Subsystems' dimensions

### Returns

Schmidt coefficients of A, as a complex dynamic column vector

Here is the call graph for this function:



6.1.2.101 template<typename Derived > DynColVect<double> qpp::schmidtprob ( const Eigen::MatrixBase< Derived > & A, const std::vector< std::size t > & 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

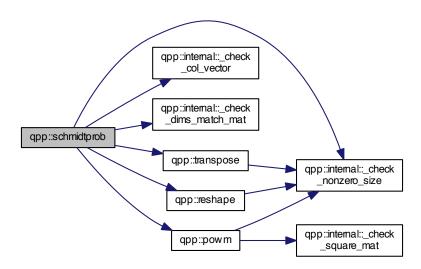
qpp::schmidtcoeff()

Α	Eigen expression
dims	Subsystems' dimensions

# Returns

Schmidt probabilites of A, as a real dynamic column vector

Here is the call graph for this function:



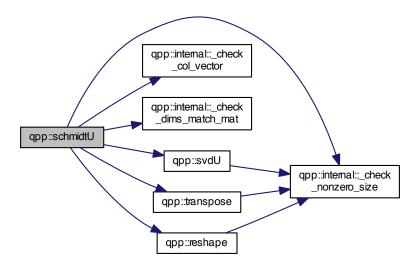
6.1.2.102 template < typename Derived > cmat qpp::schmidtU ( const Eigen::MatrixBase < Derived > & A, const std::vector < std::size\_t > & dims )

Schmidt basis on Alice's side.

Α	Eigen expression
dims	Subsystems' dimensions

Unitary matrix  $\boldsymbol{U}$  whose columns represent the Schmidt basis vectors on Alice's side.

Here is the call graph for this function:



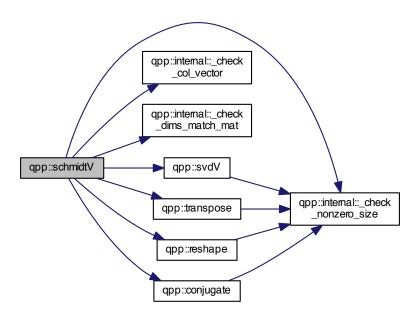
6.1.2.103 template < typename Derived > cmat qpp::schmidtV ( const Eigen::MatrixBase < Derived > & A, const std::vector < std::size\_t > & dims )

Schmidt basis on Bob's side.

Λ	Figen expression
Λ	Ligen expression
dims	Subsystems' dimensions

Unitary matrix  ${\it V}$  whose columns represent the Schmidt basis vectors on Bob's side.

Here is the call graph for this function:



6.1.2.104 template < typename Derived > double qpp::shannon ( const Eigen::MatrixBase < Derived > & A )

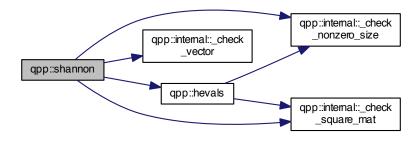
Shannon/von-Neumann entropy of the probability distribution/density matrix  $\boldsymbol{A}$ .

## **Parameters**

A Eigen expression, representing a probability distribution (real dynamic column vector) or a density matrix (complex dynamic matrix)

Shannon/von-Neumann entropy, with the logarithm in base 2

Here is the call graph for this function:



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

Matrix sin.

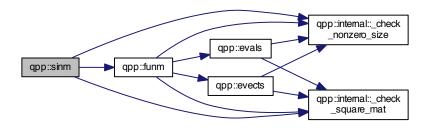
**Parameters** 

Α	Eigen expression

Returns

Matrix sine of A

Here is the call graph for this function:



 $6.1.2.106 \quad template < typename \ Derived > cmat \ qpp::spectralpowm ( \ const \ Eigen::MatrixBase < Derived > \& \ \textit{A}, \ const \ cplx \ \textit{z} \ )$ 

Matrix power.

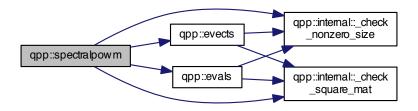
Uses the spectral decomposition of  $\emph{A}$  to compute the matrix power By convention  $\emph{A}^0 = \emph{I}$ 

Α	Eigen expression
Z	Complex number

## Returns

Matrix power  $A^z$ 

Here is the call graph for this function:



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

Matrix square root.

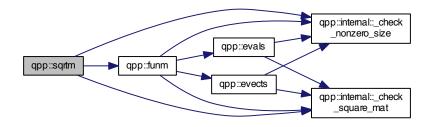
# **Parameters**

Α	Eigen expression

## Returns

Matrix square root of A

Here is the call graph for this function:



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

Element-wise sum of A.

Α	Eigen expression
---	------------------

#### Returns

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

Here is the call graph for this function:



6.1.2.109 template<typename InputIterator > auto qpp::sum ( InputIterator first, InputIterator last ) -> typename InputIterator::value\_type

Element-wise sum of a range.

### **Parameters**

first	Iterator to the first element of the range
last	Iterator to the last element of the range

### Returns

Element-wise sum of the range, as a scalar in the same scalar field as the range

6.1.2.110 cmat qpp::super ( const std::vector < cmat > & Ks )

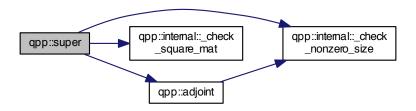
Superoperator matrix representation.

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.

Ks
----

Superoperator matrix representation

Here is the call graph for this function:



6.1.2.111 template<typename Derived > DynColVect<double> qpp::svals ( const Eigen::MatrixBase< Derived > & A )

## Singular values.

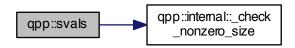
## **Parameters**

Α	Eigen expression

# Returns

Singular values of A, as a real dynamic column vector

Here is the call graph for this function:



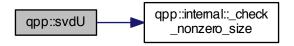
6.1.2.112 template<typename Derived > cmat qpp::svdU ( const Eigen::MatrixBase< Derived > & A )

Left singular vectors.

Α	Eigen expression

Complex dynamic matrix, whose columns are the left singular vectors of A

Here is the call graph for this function:



6.1.2.113 template<typename Derived > cmat qpp::svdV ( const Eigen::MatrixBase< Derived > & A )

Right singular vectors.

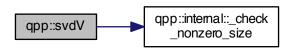
**Parameters** 

Α	Eigen expression

## Returns

Complex dynamic matrix, whose columns are the right singular vectors of A

Here is the call graph for this function:



6.1.2.114 template < typename Derived > DynMat < typename Derived::Scalar > qpp::syspermute ( const Eigen::MatrixBase < Derived > & A, const std::vector < std::size\_t > & perm, const std::vector < std::size\_t > & dims)

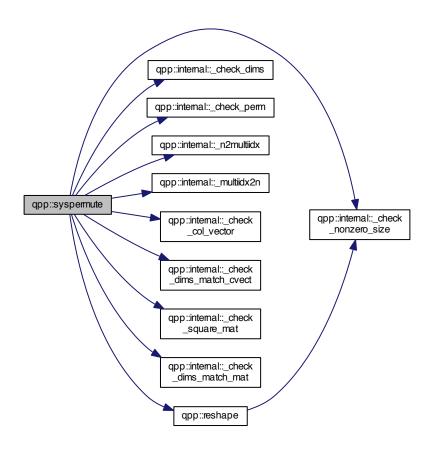
System permutation.

Permutes the subsystems in a state vector or density matrix The qubit perm[i] is permuted to the location i

Α	Eigen expression
perm	Permutation
dims	Subsystems' dimensions

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

Here is the call graph for this function:



6.1.2.115 template<typename Derived > Derived::Scalar qpp::trace ( const Eigen::MatrixBase< Derived > & A )

## Trace.

Α	Eigen expression

Trace of A, as a scalar in the same scalar field as A

Here is the call graph for this function:



6.1.2.116 template<typename Derived > DynMat<typename Derived::Scalar> qpp::transpose ( const Eigen::MatrixBase< Derived > & A )

Transpose.

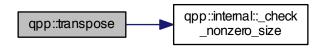
#### **Parameters**

Α	Eigen expression

#### Returns

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

Here is the call graph for this function:



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

Tsallis-  $\alpha$  entropy of the probability distribution/density matrix A, for  $\alpha \geq 0$ 

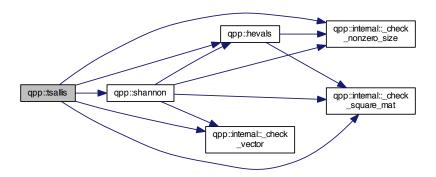
When  $\alpha \to 1$  the Tsallis entropy converges to the Shannon/von-Neumann entropy, with the logarithm in base e

A Eigen expression, representing a probability distribution (real dynamic column vector) or a density matrix (complex dynamic matrix)

alpha	Non-negative real number

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

Here is the call graph for this function:



# 6.1.3 Variable Documentation

# 6.1.3.1 constexpr double qpp::chop = 1e-10

Used in *qpp::disp()* and *qpp::displn()* for setting to zero numbers that have their absolute value smaller than *qpp ⇔ ::ct::chop*.

# 6.1.3.2 const Codes& qpp::codes = Codes::get\_instance()

qpp::Codes const Singleton

Initializes the codes, see the class *qpp::Codes* 

6.1.3.3 constexpr double qpp::ee = 2.718281828459045235360287471352662497

Base of natural logarithm, e.

# 6.1.3.4 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</pre>
```

# 6.1.3.5 const Gates& qpp::gt = Gates::get\_instance()

qpp::Gates const Singleton

Initializes the gates, see the class qpp::Gates

```
6.1.3.6 constexpr std::size_t qpp::infty = -1

Used to denote infinity.

6.1.3.7 const Init& qpp::init = Init::get_instance()

qpp::Init const Singleton

Additional initializations/cleanups

6.1.3.8 constexpr std::size_t qpp::maxn = 64

Maximum number of qubits.

Used internally to allocate arrays on the stack (for speed reasons)
```

6.1.3.10 RandomDevices& qpp::rdevs = RandomDevices::get\_instance()

6.1.3.9 constexpr double qpp::pi = 3.141592653589793238462643383279502884

qpp::RandomDevices Singleton

Initializes the random devices, see the class qpp::RandomDevices

6.1.3.11 const States& qpp::st = States::get\_instance()

qpp::States const Singleton

Initializes the states, see the class *qpp::States* 

# 6.2 qpp::experimental Namespace Reference

### Classes

 $\pi$ 

· class Qudit

# **Functions**

Applies the gate A to the part subsys of a multipartite state vector or density matrix.

template<typename Derived >
 cmat channel (const Eigen::MatrixBase< Derived > &rho, const std::vector< cmat > &Ks, const std::vector< std::size\_t > &subsys, const std::vector< std::size\_t > &dims)

Applies the channel specified by the set of Kraus operators Ks to the part of the density matrix rho specified by subsys.

cmat super (const std::vector< cmat > &Ks)

Superoperator matrix representation.

template<typename Derived >
 DynMat< typename Derived::Scalar > CTRL (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size\_t > &ctrl, const std::vector< std::size\_t > &subsys, std::size\_t n, std::size\_t d=2)

Generates the multipartite multiple-controlled-A gate in matrix form.

cmat choi (const std::vector< cmat > &Ks)

Choi matrix representation.

• std::vector< cmat > randkraus (std::size\_t n, std::size\_t D)

Generates a set of random Kraus operators.

template<typename Derived >
 double renyi\_inf (const Eigen::MatrixBase< Derived > &A)

Renyi- ∞ entropy (min entropy) of the probability distribution/density matrix A.

# 6.2.1 Detailed Description

Experimental/test functions, do not use/modify these functions/classes

### 6.2.2 Function Documentation

6.2.2.1 template < typename Derived1 , typename Derived2 > DynMat < typename Derived1::Scalar > qpp::experimental::apply ( const Eigen::MatrixBase < Derived1 > & state, const Eigen::MatrixBase < Derived2 > & A, const std::vector < std::size\_t > & subsys, const std::vector < std::size\_t > & dims )

Applies the gate A to the part subsys of a multipartite state vector or density matrix.

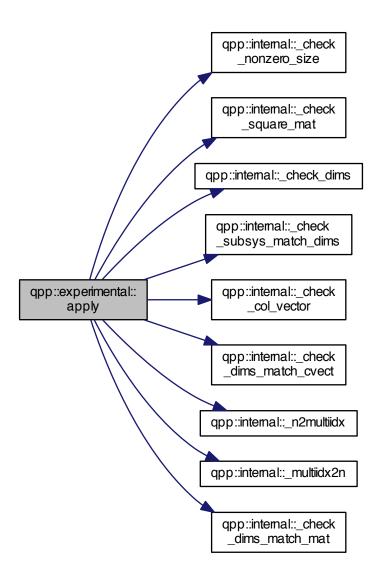
Note

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

state	Eigen expression
Α	Eigen expression
subsys	Subsystem indexes where the gate A is applied
dims	Local dimensions of all local Hilbert spaces (can be different)

Gate A applied to the part subsys of state

Here is the call graph for this function:



6.2.2.2 template<typename Derived > cmat qpp::experimental::channel ( const Eigen::MatrixBase< Derived > & rho, const std::vector< cmat > & Ks, const std::vector< std::size\_t > & subsys, const std::vector< std::size\_t > & dims )

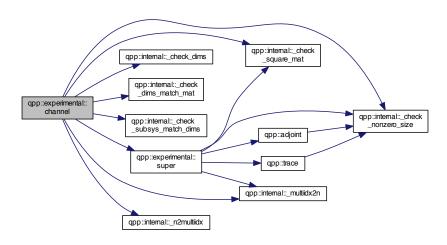
Applies the channel specified by the set of Kraus operators *Ks* to the part of the density matrix *rho* specified by *subsys*.

rho	Eigen expression
Ks	Set of Kraus operators
subsys	Subsystems' indexes
dims	Local dimensions of all local Hilbert spaces (can be different)

## Returns

Output density matrix after the action of the channel

Here is the call graph for this function:



6.2.2.3 cmat qpp::experimental::choi ( const std::vector< cmat > & Ks )

Choi matrix representation.

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

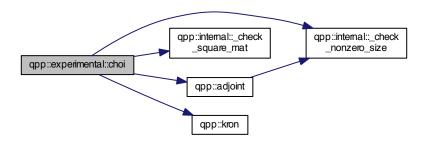
The superoperator matrix S and the Choi matrix C are related by  $S_{ab,mn} = C_{ma,nb}$ 

Ks	Set of Kraus operators

#### Returns

Choi matrix representation

Here is the call graph for this function:



6.2.2.4 template<typename Derived > DynMat<typename Derived::Scalar> qpp::experimental::CTRL ( const Eigen::MatrixBase< Derived > & A, const std::vector< std::size\_t > & ctrl, const std::vector< std::size\_t > & subsys, std::size\_t n, std::size\_t d = 2 )

Generates the multipartite multiple-controlled-A gate in matrix form.

### Note

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

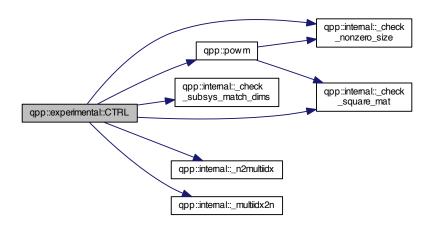
#### **Parameters**

A	Eigen expression
ctrl	Control subsystem indexes
subsys	Subsystem indexes where the gate A is applied
n	Total number of subsystes
d	Local dimensions of all local Hilbert spaces (must all be equal)

#### Returns

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

Here is the call graph for this function:



6.2.2.5 std::vector<cmat> qpp::experimental::randkraus ( std::size\_t n, std::size\_t D )

Generates a set of random Kraus operators.

Note

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

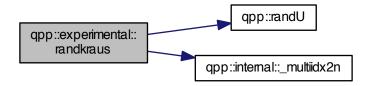
## **Parameters**

n	Number of Kraus operators
D	Dimension of the Hilbert space

#### Returns

Set of *n* Kraus operators satisfying the closure condition

Here is the call graph for this function:



 $\textbf{6.2.2.6} \quad \textbf{template} \\ < \textbf{typename Derived} \\ > \textbf{double qpp::experimental::renyi\_inf ( const Eigen::MatrixBase} \\ < \textbf{Derived} \\ > \textbf{\& A )} \\$ 

Renyi-  $\infty$  entropy (min entropy) of the probability distribution/density matrix A.

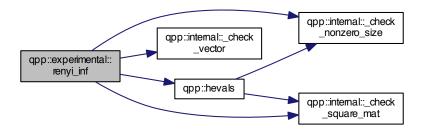
#### **Parameters**

Α	Eigen expression, representing a probability distribution (real dynamic column vector) or a
	density matrix (complex dynamic matrix)

### Returns

Renyi- ∞ entropy (min entropy), with the logarithm in base 2

Here is the call graph for this function:



### 6.2.2.7 cmat qpp::experimental::super ( const std::vector < cmat > & Ks )

Superoperator matrix representation.

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

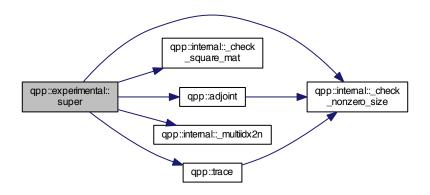
#### **Parameters**

Ks	Set of Kraus operators

# Returns

Superoperator matrix representation

Here is the call graph for this function:



# 6.3 qpp::internal Namespace Reference

#### Classes

· class Singleton

Singleton policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern)

#### **Functions**

- void n2multiidx (std::size t n, std::size t numdims, const std::size t \*dims, std::size t \*result)
- std::size\_t \_multiidx2n (const std::size\_t \*midx, std::size\_t numdims, const std::size\_t \*dims)
- template<typename Derived >

```
bool <u>_check_square_mat</u> (const Eigen::MatrixBase< Derived > &A)
```

template<typename Derived >

```
bool <u>check_vector</u> (const Eigen::MatrixBase< Derived > &A)
```

template<typename Derived >

```
bool check row vector (const Eigen::MatrixBase< Derived > &A)
```

• template<typename Derived >

```
bool <u>_check_col_vector</u> (const Eigen::MatrixBase< Derived > &A)
```

template<typename T >

```
bool check nonzero size (const T &x)
```

- bool <u>\_check\_dims</u> (const std::vector< std::size\_t > &dims)
- template<typename Derived >

```
bool _check_dims_match_mat (const std::vector< std::size_t > &dims, const Eigen::MatrixBase< Derived > &A)
```

 $\bullet \ \ \text{template}{<} \text{typename Derived} >$ 

```
bool _check_dims_match_cvect (const std::vector< std::size_t > &dims, const Eigen::MatrixBase< Derived > &V)
```

template<typename Derived >

```
bool _check_dims_match_rvect (const std::vector< std::size_t > &dims, const Eigen::MatrixBase< Derived > &V)
```

- bool <u>\_check\_eq\_dims</u> (const std::vector< std::size\_t > &dims, std::size\_t dim)
- bool \_check\_subsys\_match\_dims (const std::vector< std::size\_t > &subsys, const std::vector< std::size\_t > &dims)
- bool \_check\_perm (const std::vector< std::size\_t > &perm)
- template<typename Derived1 , typename Derived2 >

• 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)

# 6.3.1 Detailed Description

Internal implementation details, do not use/modify these functions/classes

#### 6.3.2 Function Documentation

- 6.3.2.1 template < typename Derived > bool qpp::internal::\_check\_col\_vector ( const Eigen::MatrixBase < Derived > & A )
- 6.3.2.2 bool qpp::internal::\_check\_dims ( const std::vector < std::size\_t > & dims )

- 6.3.2.3 template<typename Derived > bool qpp::internal::\_check\_dims\_match\_cvect ( const std::vector< std::size\_t > & dims, const Eigen::MatrixBase< Derived > & V )
- 6.3.2.4 template<typename Derived > bool qpp::internal::\_check\_dims\_match\_mat ( const std::vector< std::size\_t > & dims, const Eigen::MatrixBase< Derived > & A )
- 6.3.2.5 template<typename Derived > bool qpp::internal::\_check\_dims\_match\_rvect ( const std::vector< std::size\_t > & dims, const Eigen::MatrixBase< Derived > & V )
- 6.3.2.6 bool qpp::internal::\_check\_eq\_dims ( const std::vector < std::size\_t > & dims, std::size\_t dim )
- 6.3.2.7 template<typename T > bool qpp::internal::\_check\_nonzero\_size ( const T & x )
- 6.3.2.8 bool qpp::internal::\_check\_perm ( const std::vector < std::size\_t > & perm )
- 6.3.2.9 template < typename Derived > bool qpp::internal::\_check\_row\_vector ( const Eigen::MatrixBase < Derived > & A )
- 6.3.2.10 template < typename Derived > bool qpp::internal::\_check\_square\_mat ( const Eigen::MatrixBase < Derived > & A )
- 6.3.2.11 bool qpp::internal::\_check\_subsys\_match\_dims ( const std::vector< std::size\_t > & subsys, const std::vector< std::size\_t > & dims )
- 6.3.2.12 template < typename Derived > bool qpp::internal::\_check\_vector ( const Eigen::MatrixBase < Derived > & A )

Here is the call graph for this function:

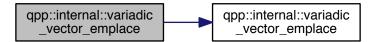


- 6.3.2.14 std::size\_t qpp::internal::\_multiidx2n ( const std::size\_t \* midx, std::size\_t numdims, const std::size\_t \* dims )
  [inline]
- 6.3.2.15 void qpp::internal::\_n2multiidx ( std::size\_t *n*, std::size\_t *numdims*, const std::size\_t \* *dims*, std::size\_t \* *result* )

  [inline]
- 6.3.2.16 template < typename T > void qpp::internal::variadic\_vector\_emplace(\_std::vector < T > & \_)

6.3.2.17 template < typename T , typename First , typename... Args > void qpp::internal::variadic\_vector\_emplace ( std::vector < T > & v, First && first, Args &&... args )

Here is the call graph for this function:





# **Chapter 7**

# **Class Documentation**

# 7.1 qpp::Codes Class Reference

const Singleton class that defines quantum error correcting codes

#include <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, std::size\_t i) const
 Returns the codeword of the specified code.

### **Private Member Functions**

• Codes ()=default

Default constructor.

#### **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( ) [private],[default]
```

Default constructor.

### 7.1.4 Member Function Documentation

7.1.4.1 ket qpp::Codes::codeword ( Type type, std::size\_t i ) const [inline]

Returns the codeword of the specified code.

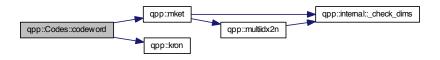
#### **Parameters**

type	Code type, defined in the enum <i>qpp::Codes::Types</i>
i	Codeword index

#### Returns

i-th codeword of the code type

Here is the call graph for this function:



### 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:

• include/classes/codes.h

# 7.2 qpp::Exception Class Reference

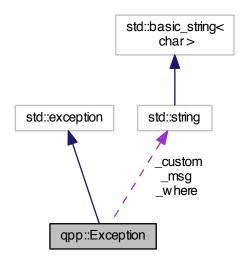
Generates custom exceptions, used when validating function parameters.

#include <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::NOT\_QUBIT\_GATE, Type::NOT\_QUBI

▼ SUBSYS,

Type::NOT\_BIPARTITE, Type::OUT\_OF\_RANGE, Type::TYPE\_MISMATCH, Type::UNDEFINED\_TYPE, Type::NO\_CODEWORD, Type::CUSTOM\_EXCEPTION }

Exception types, add more exceptions 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 noexceptoverride

Overrides std::exception::what()

#### **Private Member Functions**

• std::string \_construct\_exception\_msg ()

Constructs the exception's description from its type.

#### **Private Attributes**

- · std::string \_where
- std::string \_msg
- Type \_type
- std::string custom

#### 7.2.1 Detailed Description

Generates custom exceptions, used when validating function parameters.

Customize this class if more exceptions are needed

#### 7.2.2 Member Enumeration Documentation

**7.2.2.1** enum qpp::Exception::Type [strong]

Exception types, add more exceptions here if needed.

See also

qpp:Exception::\_construct\_exception\_msg()

#### **Enumerator**

UNKNOWN\_EXCEPTION UNKNOWN\_EXCEPTION. Unknown exception

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

MATRIX\_NOT\_SQUARE MATRIX\_NOT\_SQUARE. Eigen::Matrix is not square

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

MATRIX\_NOT\_RVECTOR MATRIX NOT RVECTOR. Eigen::Matrix is not a row vector

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

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

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

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

**MATRIX\_MISMATCH\_SUBSYS** SUBSYS\_MISMATCH\_MATRIX.

**DIMS\_NOT\_EQUAL** DIMS\_NOT\_EQUAL. std::vector<std::size\_t> representing the dimensions contains non-equal elements

**DIMS\_MISMATCH\_MATRIX** DIMS\_MISMATCH\_MATRIX. Product of the dimenisons' std::vector<std↔ ::size t> is not equal to the number of rows of Eigen::Matrix (assumed to be square)

**DIMS\_MISMATCH\_CVECTOR** DIMS\_MISMATCH\_CVECTOR. Product of the dimenisons' std::vector<std↔ ::size\_t> is not equal to the number of cols of Eigen::Matrix (assumed to be a column vector)

**DIMS\_MISMATCH\_RVECTOR** DIMS\_MISMATCH\_RVECTOR. Product of the dimenisons' std::vector<std↔ ::size\_t> is not equal to the number of cols of Eigen::Matrix (assumed to be a row vector)

**DIMS\_MISMATCH\_VECTOR** DIMS\_MISMATCH\_VECTOR. Product of the dimenisons' std::vector<std↔ ::size\_t> is not equal to the number of cols of Eigen::Matrix (assumed to be a row/column vector)

**SUBSYS\_MISMATCH\_DIMS** SUBSYS\_MISMATCH\_DIMS. std::vector<std::size\_t> representing the subsystems' labels has duplicatates, or has entries that are larger than the size of the std::vector<std::size -\_t> representing the dimensions

**PERM\_INVALID** PERM\_INVALID. Invalid std::vector<std::size\_t> permutation

NOT\_QUBIT\_GATE NOT\_QUBIT\_GATE. Eigen::Matrix is not 2 x 2

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

**NOT\_BIPARTITE** NOT\_BIPARTITE. std::vector<std::size\_t> representing the dimensions has size different from 2

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

TYPE\_MISMATCH. Types do not match (i.e. Matrix<double> vs Matrix<cplx>)

UNDEFINED\_TYPE UNDEFINED\_TYPE. Templated function not defined for this type

NO\_CODEWORD

**CUSTOM\_EXCEPTION** < NO\_CODEWORD Codeword does not exist, thrown when calling *qpp::Codes*← ::codeword() with invalid *i* CUSTOM\_EXCEPTION Custom exception, user must provide a custom message

#### 7.2.3 Constructor & Destructor Documentation

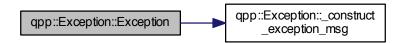
7.2.3.1 qpp::Exception::Exception ( const std::string & where, const Type & type ) [inline]

Constructs an exception.

#### **Parameters**

where	Text representing where the exception occured
type	Exception's type, see the strong enumeration qpp::Exception::Type

Here is the call graph for this function:



7.2.3.2 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**

where	Text representing where the exception occured
custom	Exception's description

Here is the call graph for this function:



#### 7.2.4 Member Function Documentation

7.2.4.1 std::string qpp::Exception::\_construct\_exception\_msg( ) [inline], [private]

Constructs the exception's description from its type.

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

#### Returns

Exception's description

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

Overrides std::exception::what()

#### Returns

**Exception**'s description

### 7.2.5 Member Data Documentation

**7.2.5.1** std::string qpp::Exception::\_custom [private]

**7.2.5.2 std::string qpp::Exception::\_msg** [private]

**7.2.5.3 Type qpp::Exception::\_type** [private]

**7.2.5.4 std::string qpp::Exception::\_where** [private]

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

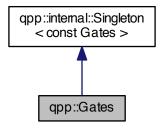
• include/classes/exception.h

# 7.3 qpp::Gates Class Reference

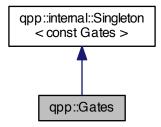
const Singleton class that implements most commonly used gates

#include <gates.h>

Inheritance diagram for qpp::Gates:



Collaboration diagram for qpp::Gates:



#### **Public Member Functions**

• cmat Rn (double theta, std::vector< double > n) const

Rotation of theta about the 3-dimensional real unit vector n.

cmat Zd (std::size t D) const

Generalized Z gate for qudits.

cmat Fd (std::size\_t D) const

Fourier transform gate for qudits.

cmat Xd (std::size\_t D) const

Generalized X gate for qudits.

• template<typename Derived = Eigen::MatrixXcd>

Derived Id (std::size\_t D) const

Identity gate.

ullet template<typename Derived >

DynMat< typename Derived::Scalar > CTRL (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size\_t > &ctrl, const std::vector< std::size\_t > &subsys, std::size\_t n, std::size\_t d=2) const

Generates the multipartite multiple-controlled-A gate in matrix form.

• template<typename Derived >

DynMat< typename Derived::Scalar > expandout (const Eigen::MatrixBase< Derived > &A, std::size\_t pos, const std::vector< std::size\_t > &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 CNOTab { cmat::ldentity(4, 4) }
      Controlled-NOT control target gate.
cmat CZ { cmat::ldentity(4, 4) }
     Controlled-Phase gate.
cmat CNOTba { cmat::Zero(4, 4) }
     Controlled-NOT target control gate.
cmat SWAP { cmat::ldentity(4, 4) }
     SWAP gate.
cmat TOF { cmat::ldentity(8, 8) }
      Toffoli gate.
cmat FRED { cmat::Identity(8, 8) }
     Fredkin gate.
```

#### **Private Member Functions**

• Gates ()
Initializes the gates.

#### **Friends**

class internal::Singleton < const Gates >

#### **Additional Inherited Members**

# 7.3.1 Detailed Description

const Singleton class that implements most commonly used gates

#### 7.3.2 Constructor & Destructor Documentation

```
7.3.2.1 qpp::Gates::Gates( ) [inline],[private]
```

Initializes the gates.

### 7.3.3 Member Function Documentation

7.3.3.1 template<typename Derived > DynMat<typename Derived::Scalar> qpp::Gates::CTRL ( const Eigen::MatrixBase< Derived > & A, const std::vector< std::size\_t > & ctrl, const std::vector< std::size\_t > & subsys, std::size\_t n, std::size\_t d = 2 ) const [inline]

Generates the multipartite multiple-controlled-A gate in matrix form.

Note

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

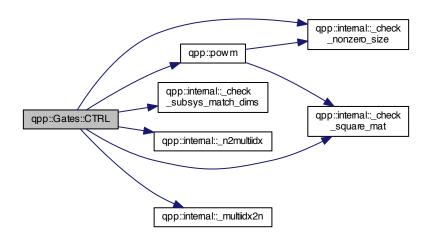
#### **Parameters**

Α	Eigen expression
ctrl	Control subsystem indexes
subsys	Subsystem indexes where the gate A is applied
n	Total number of subsystes
d	Local dimensions of all local Hilbert spaces (must all be equal)

#### Returns

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

Here is the call graph for this function:



7.3.3.2 template<typename Derived > DynMat<typename Derived::Scalar> qpp::Gates::expandout ( const Eigen::MatrixBase< Derived > & A, std::size\_t pos, const std::vector< std::size\_t > & dims ) const [inline]

Expands out.

Expands out *A* as a matrix in a multi-partite system Faster than using *qpp::kron*(I, I, ..., I, A, I, ..., I)

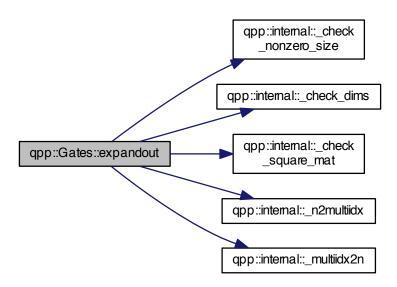
#### **Parameters**

Α	Eigen expression
pos	Position
dims	Dimensions of the multi-partite system

### Returns

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

Here is the call graph for this function:



7.3.3.3 cmat qpp::Gates::Fd ( std::size\_t 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

Here is the call graph for this function:



7.3.3.4 template < typename Derived = Eigen::MatrixXcd > Derived qpp::Gates::Id ( std::size\_t 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

7.3.3.5 cmat qpp::Gates::Rn ( double theta, std::vector< double > n ) const [inline]

Rotation of theta about the 3-dimensional real unit vector n.

#### **Parameters**

theta	Rotation angle
n	3-dimensional real unit vector

## Returns

Rotation gate

7.3.3.6 cmat qpp::Gates::Xd ( std::size\_t D ) const [inline]

Generalized X gate for qudits.

Note

Defined as  $X = \sum_{j} |j \oplus 1\rangle\langle j|$ 

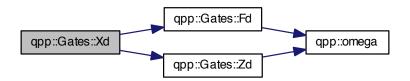
#### **Parameters**

D	Dimension of the Hilbert space

### Returns

Generalized X gate for qudits

Here is the call graph for this function:



7.3.3.7 cmat qpp::Gates::Zd ( std::size\_t 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

Here is the call graph for this function:



- 7.3.4 Friends And Related Function Documentation
- **7.3.4.1** friend class internal::Singleton < const Gates > [friend]

### 7.3.5 Member Data Documentation

```
7.3.5.1 cmat qpp::Gates::CNOTab { cmat::Identity(4, 4) }
Controlled-NOT control target gate.
7.3.5.2 cmat qpp::Gates::CNOTba { cmat::Zero(4, 4) }
Controlled-NOT target control gate.
7.3.5.3 cmat qpp::Gates::CZ { cmat::Identity(4, 4) }
Controlled-Phase gate.
7.3.5.4 cmat qpp::Gates::FRED { cmat::Identity(8, 8) }
Fredkin gate.
7.3.5.5 cmat qpp::Gates::H { cmat::Zero(2, 2) }
Hadamard gate.
7.3.5.6 cmat qpp::Gates::Id2 { cmat::Identity(2, 2) }
Identity gate.
7.3.5.7 cmat qpp::Gates::S { cmat::Zero(2, 2) }
S gate.
7.3.5.8 cmat qpp::Gates::SWAP { cmat::Identity(4, 4) }
SWAP gate.
7.3.5.9 cmat qpp::Gates::T { cmat::Zero(2, 2) }
T gate.
7.3.5.10 cmat qpp::Gates::TOF { cmat::Identity(8, 8) }
Toffoli gate.
7.3.5.11 cmat qpp::Gates::X { cmat::Zero(2, 2) }
Pauli Sigma-X gate.
7.3.5.12 cmat qpp::Gates::Y { cmat::Zero(2, 2) }
Pauli Sigma-Y gate.
```

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

Pauli Sigma-Z gate.

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

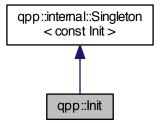
• include/classes/gates.h

# 7.4 qpp::Init Class Reference

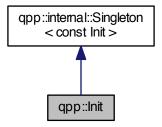
const Singleton class that performs additional initializations/cleanups

#include <init.h>

Inheritance diagram for qpp::Init:



Collaboration diagram for qpp::Init:



**Public Member Functions** 

• Init ()

Additional initializations.

#### **Private Member Functions**

```
• ∼Init ()

Cleanups.
```

#### **Friends**

Cleanups.

class internal::Singleton < const Init >

#### **Additional Inherited Members**

# 7.4.1 Detailed Description

const Singleton class that performs additional initializations/cleanups

#### 7.4.2 Constructor & Destructor Documentation

```
7.4.2.1 qpp::Init::Init( ) [inline]
Additional initializations.
7.4.2.2 qpp::Init::~Init( ) [inline], [private]
```

# 7.4.3 Friends And Related Function Documentation

```
7.4.3.1 friend class internal::Singleton < const lnit > [friend]
```

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

• include/classes/init.h

# 7.5 qpp::experimental::Qudit Class Reference

```
#include <qudit.h>
```

### **Public Member Functions**

- Qudit (const cmat &rho=States::get\_instance().pz0)
- std::size\_t measure (const cmat &U, bool destructive=false)
- std::size\_t measure (bool destructive=false)
- cmat getRho () const
- std::size\_t getD () const

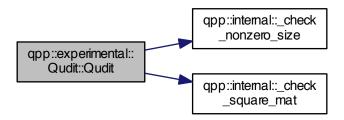
#### **Private Attributes**

- · cmat \_rho
- std::size\_t \_D

### 7.5.1 Constructor & Destructor Documentation

7.5.1.1 qpp::experimental::Qudit::Qudit ( const cmat & rho = States::get\_instance () .pz0 ) [inline]

Here is the call graph for this function:



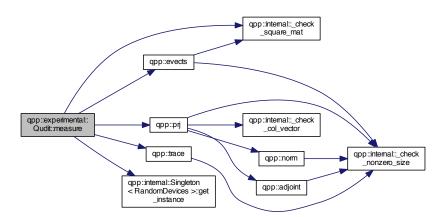
# 7.5.2 Member Function Documentation

7.5.2.1 std::size\_t qpp::experimental::Qudit::getD ( ) const [inline]

 $\textbf{7.5.2.2} \quad \textbf{cmat qpp::experimental::Qudit::getRho ( ) const} \quad \texttt{[inline]}$ 

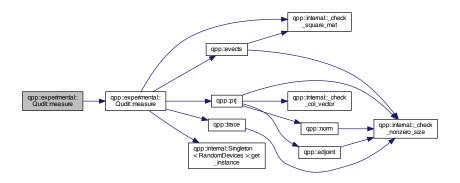
7.5.2.3 std::size\_t qpp::experimental::Qudit::measure ( const cmat & U, bool destructive = false ) [inline]

Here is the call graph for this function:



7.5.2.4 std::size\_t qpp::experimental::Qudit::measure ( bool destructive = false ) [inline]

Here is the call graph for this function:



#### 7.5.3 Member Data Documentation

**7.5.3.1** std::size\_t qpp::experimental::Qudit::\_D [private]

**7.5.3.2 cmat qpp::experimental::Qudit::\_rho** [private]

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

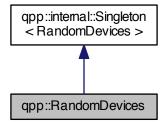
• include/experimental/classes/qudit.h

# 7.6 qpp::RandomDevices Class Reference

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

#include <randevs.h>

Inheritance diagram for qpp::RandomDevices:



Collaboration diagram for qpp::RandomDevices:



### **Public Attributes**

std::mt19937 \_rng

Mersenne twister random number generator engine.

#### **Private Member Functions**

· RandomDevices ()

Initializes and seeds the random number generators.

# **Private Attributes**

std::random\_device \_rd
 used to seed std::mt19937 \_rng

# **Friends**

class internal::Singleton < RandomDevices >

# **Additional Inherited Members**

## 7.6.1 Detailed Description

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

It 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. The class also seeds the standard std::srand C number generator, as it is used by Eigen.

#### 7.6.2 Constructor & Destructor Documentation

7.6.2.1 qpp::RandomDevices::RandomDevices() [inline], [private]

Initializes and seeds the random number generators.

#### 7.6.3 Friends And Related Function Documentation

```
7.6.3.1 friend class internal::Singleton < RandomDevices > [friend]
```

#### 7.6.4 Member Data Documentation

```
7.6.4.1 std::random_device qpp::RandomDevices::_rd [private]
used to seed std::mt19937 _rng
```

```
7.6.4.2 std::mt19937 qpp::RandomDevices::_rng
```

Mersenne twister random number generator engine.

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

· include/classes/randevs.h

# 7.7 qpp::internal::Singleton < T > Class Template Reference

Singleton policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern)

```
#include <singleton.h>
```

#### **Static Public Member Functions**

• static T & get\_instance ()

#### **Protected Member Functions**

- Singleton ()=default
- virtual ∼Singleton ()
- Singleton (const Singleton &)=delete
- Singleton & operator= (const Singleton &)=delete

## 7.7.1 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 of your class as private. To get an instance, use the static member function *qpp::internal::Singleton::get\_instance()*, which returns a 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& mySingleton = MySingleton::get_instance(); // Get an instance
```

See also

Code of *qpp::Gates*, *qpp::RandomDevices*, *qpp::States* or *qpp.h* for real world examples of usage.

- 7.7.2 Constructor & Destructor Documentation
- 7.7.2.1 template<typename T> qpp::internal::Singleton< T>::Singleton( ) [protected], [default]
- 7.7.2.2 template<typename T> virtual qpp::internal::Singleton< T>::~Singleton( ) [inline], [protected], [virtual]
- 7.7.2.3 template<typename T> qpp::internal::Singleton < T >::Singleton ( const Singleton < T > & ) [protected], [delete]
- 7.7.3 Member Function Documentation
- 7.7.3.1 template < typename T > static T& qpp::internal::Singleton < T >::get\_instance( ) [inline], [static]
- 7.7.3.2 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:

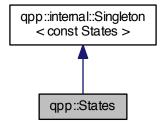
• include/classes/singleton.h

# 7.8 qpp::States Class Reference

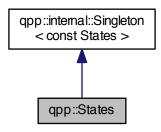
const Singleton class that implements most commonly used states

```
#include <states.h>
```

Inheritance diagram for qpp::States:



Collaboration diagram for qpp::States:



#### **Public Attributes**

```
    ket x0 { ket::Zero(2) }

      Pauli Sigma-X 0-eigenstate |+>

    ket x1 { ket::Zero(2) }

      Pauli Sigma-X 1-eigenstate |->
ket y0 { ket::Zero(2) }
      Pauli Sigma-Y 0-eigenstate.
ket y1 { ket::Zero(2) }
      Pauli Sigma-Y 1-eigenstate.

    ket z0 { ket::Zero(2) }

      Pauli Sigma-Z 0-eigenstate | 0>

    ket z1 { ket::Zero(2) }

      Pauli Sigma-Z 1-eigenstate | 1>

    cmat px0 { cmat::Zero(2, 2) }

      Projector onto the Pauli Sigma-X 0-eigenstate |+><+|.

    cmat px1 { cmat::Zero(2, 2) }

      Projector onto the Pauli Sigma-X 1-eigenstate |-><-|.

    cmat py0 { cmat::Zero(2, 2) }

      Projector onto the Pauli Sigma-Y 0-eigenstate.

    cmat py1 { cmat::Zero(2, 2) }

      Projector onto the Pauli Sigma-Y 1-eigenstate.

    cmat pz0 { cmat::Zero(2, 2) }

      Projector onto the Pauli Sigma-Z 0-eigenstate |0><0|.
cmat pz1 { cmat::Zero(2, 2) }
      Projector onto the Pauli Sigma-Z 1-eigenstate | 1><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) }
```

```
7.8 qpp::States Class Reference
          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 ()
Friends

    class internal::Singleton < const States >

Additional Inherited Members
7.8.1 Detailed Description
const Singleton class that implements most commonly used states
```

#### 7.8.2 Constructor & Destructor Documentation

```
7.8.2.1 qpp::States::States() [inline], [private]
```

### 7.8.3 Friends And Related Function Documentation

**7.8.3.1** friend class internal::Singleton < const States > [friend]

### 7.8.4 Member Data Documentation

Initialize the states

7.8.4.1 ket qpp::States::b00 { ket::Zero(4) }

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

7.8.4.2 ket qpp::States::b01 { ket::Zero(4) }

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

```
7.8.4.3 ket qpp::States::b10 { ket::Zero(4) }
Bell-10 state (following the convention in Nielsen and Chuang)
7.8.4.4 ket qpp::States::b11 { ket::Zero(4) }
Bell-11 state (following the convention in Nielsen and Chuang)
7.8.4.5 ket qpp::States::GHZ { ket::Zero(8) }
GHZ state.
7.8.4.6 cmat qpp::States::pb00 { cmat::Zero(4, 4) }
Projector onto the Bell-00 state.
7.8.4.7 cmat qpp::States::pb01 { cmat::Zero(4, 4) }
Projector onto the Bell-01 state.
7.8.4.8 cmat qpp::States::pb10 { cmat::Zero(4, 4) }
Projector onto the Bell-10 state.
7.8.4.9 cmat qpp::States::pb11 { cmat::Zero(4, 4) }
Projector onto the Bell-11 state.
7.8.4.10 cmat qpp::States::pGHZ { cmat::Zero(8, 8) }
Projector onto the GHZ state.
7.8.4.11 cmat qpp::States::pW { cmat::Zero(8, 8) }
Projector onto the W state.
7.8.4.12 cmat qpp::States::px0 { cmat::Zero(2, 2) }
Projector onto the Pauli Sigma-X 0-eigenstate |+><+|.
7.8.4.13 cmat qpp::States::px1 { cmat::Zero(2, 2) }
Projector onto the Pauli Sigma-X 1-eigenstate |-><-|.
7.8.4.14 cmat qpp::States::py0 { cmat::Zero(2, 2) }
```

Projector onto the Pauli Sigma-Y 0-eigenstate.

```
7.8.4.15 cmat qpp::States::py1 { cmat::Zero(2, 2) }
Projector onto the Pauli Sigma-Y 1-eigenstate.
7.8.4.16 cmat qpp::States::pz0 { cmat::Zero(2, 2) }
Projector onto the Pauli Sigma-Z 0-eigenstate |0><0|.
7.8.4.17 cmat qpp::States::pz1 { cmat::Zero(2, 2) }
Projector onto the Pauli Sigma-Z 1-eigenstate |1><1|.
7.8.4.18 ket qpp::States::W { ket::Zero(8) }
W state.
7.8.4.19 ket qpp::States::x0 { ket::Zero(2) }
Pauli Sigma-X 0-eigenstate |+>
7.8.4.20 ket qpp::States::x1 { ket::Zero(2) }
Pauli Sigma-X 1-eigenstate |->
7.8.4.21 ket qpp::States::y0 { ket::Zero(2) }
Pauli Sigma-Y 0-eigenstate.
7.8.4.22 ket qpp::States::y1 { ket::Zero(2) }
Pauli Sigma-Y 1-eigenstate.
7.8.4.23 ket qpp::States::z0 { ket::Zero(2) }
Pauli Sigma-Z 0-eigenstate |0>
7.8.4.24 ket qpp::States::z1 { ket::Zero(2) }
Pauli Sigma-Z 1-eigenstate |1>
The documentation for this class was generated from the following file:
```

# 7.9 qpp::Timer Class Reference

• include/classes/states.h

#### Measures time.

```
#include <timer.h>
```

#### **Public Member Functions**

• Timer ()

Constructs an instance with the current time as the starting point.

void tic ()

Resets the chronometer.

• const Timer & toc ()

Stops the chronometer.

· double seconds () const

Time passed in seconds.

#### **Protected Attributes**

```
• std::chrono::steady_clock::time_point _start
```

```
• std::chrono::steady_clock::time_point _end
```

#### **Friends**

std::ostream & operator<< (std::ostream &os, const Timer &rhs)</li>
 Overload for std::ostream operators.

### 7.9.1 Detailed Description

Measures time.

Uses a std::chrono::steady\_clock. It is not affected by wall clock changes during runtime.

#### 7.9.2 Constructor & Destructor Documentation

```
7.9.2.1 qpp::Timer::Timer() [inline]
```

Constructs an instance with the current time as the starting point.

#### 7.9.3 Member Function Documentation

```
7.9.3.1 double qpp::Timer::seconds ( ) const [inline]
```

Time passed in seconds.

#### Returns

Number of seconds that passed between the instantiation/reset and invocation of qpp::Timer::toc()

```
7.9.3.2 void qpp::Timer::tic( ) [inline]
```

Resets the chronometer.

Resets the starting/ending point to the current time

7.9.3.3 const Timer& qpp::Timer::toc( ) [inline]

Stops the chronometer.

Set the current time as the ending point

Returns

Current instance

#### 7.9.4 Friends And Related Function Documentation

7.9.4.1 std::ostream & operator << ( std::ostream & os, const Timer & rhs ) [friend]

Overload for std::ostream operators.

#### **Parameters**

os	Output stream
rhs	Timer instance

#### Returns

Writes to the output stream the number of seconds that passed between the instantiation/reset and invocation of <a href="mailto:qpp::Timer::toc()">qpp::Timer::toc()</a>.

### 7.9.5 Member Data Documentation

```
7.9.5.1 std::chrono::steady_clock::time_point qpp::Timer::_end [protected]
```

**7.9.5.2 std::chrono::steady\_clock::time\_point qpp::Timer::\_start** [protected]

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

· include/classes/timer.h

# **Chapter 8**

# **File Documentation**

# 8.1 include/classes/codes.h File Reference

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



#### Classes

class qpp::Codes

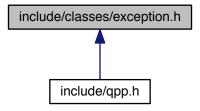
const Singleton class that defines quantum error correcting codes

# **Namespaces**

• qpp

# 8.2 include/classes/exception.h File Reference

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



# Classes

· class qpp::Exception

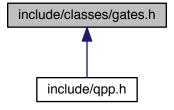
Generates custom exceptions, used when validating function parameters.

# **Namespaces**

qpp

# 8.3 include/classes/gates.h File Reference

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



# **Classes**

class qpp::Gates

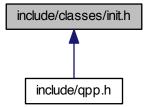
const Singleton class that implements most commonly used gates

# **Namespaces**

• qpp

# 8.4 include/classes/init.h File Reference

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



# Classes

· class qpp::Init

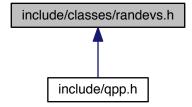
const Singleton class that performs additional initializations/cleanups

# **Namespaces**

• qpp

# 8.5 include/classes/randevs.h File Reference

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



# Classes

class qpp::RandomDevices

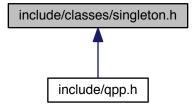
Singeleton class that manages the source of randomness in the library.

# **Namespaces**

qpp

# 8.6 include/classes/singleton.h File Reference

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



# Classes

class qpp::internal::Singleton< T >

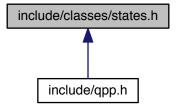
Singleton policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern)

# **Namespaces**

- qpp
- · qpp::internal

# 8.7 include/classes/states.h File Reference

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



# Classes

• class qpp::States

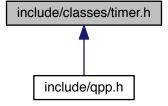
const Singleton class that implements most commonly used states

# **Namespaces**

qpp

# 8.8 include/classes/timer.h File Reference

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



# **Classes**

· class qpp::Timer

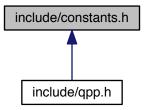
Measures time.

#### **Namespaces**

qpp

#### 8.9 include/constants.h File Reference

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



### **Namespaces**

qpp

#### **Functions**

```
    constexpr std::complex < double > qpp::operator""_i (unsigned long long int x)
```

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

constexpr std::complex< double > qpp::operator""\_i (long double x)

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

std::complex < double > qpp::omega (std::size\_t D)

D-th root of unity.

#### **Variables**

• constexpr double <a href="mailto:qpp::chop">qpp::chop</a> = 1e-10

Used in qpp::disp() and qpp::displn() for setting to zero numbers that have their absolute value smaller than qpp::ct← ::chop.

• constexpr double qpp::eps = 1e-12

Used to decide whether a number or expression in double precision is zero or not.

constexpr std::size\_t qpp::maxn = 64

Maximum number of qubits.

• constexpr double qpp::pi = 3.141592653589793238462643383279502884

π

constexpr double qpp::ee = 2.718281828459045235360287471352662497

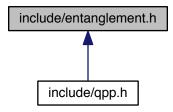
Base of natural logarithm, e.

constexpr std::size\_t qpp::infty = -1

Used to denote infinity.

# 8.10 include/entanglement.h File Reference

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



#### **Namespaces**

qpp

#### **Functions**

template<typename Derived >
 DynColVect< cplx > qpp::schmidtcoeff (const Eigen::MatrixBase< Derived > &A, const std::vector< std
 ::size\_t > &dims)

Schmidt coefficients of the bi-partite pure state A.

template<typename Derived >

 $\verb|cmat|| \textit{qpp::schmidtU}| (const| Eigen::MatrixBase < Derived > \&A, const| std::vector < std::size\_t > \&dims)|$ 

Schmidt basis on Alice's side.

• template<typename Derived >

cmat qpp::schmidtV (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size\_t > &dims)

Schmidt basis on Bob's side.

template<typename Derived >

 $\label{lem:def:DynColVect} \mbox{DynColVect} < \mbox{double} > \mbox{qpp::schmidtprob} \mbox{ (const Eigen::MatrixBase} < \mbox{Derived} > \&\mbox{A, const std::vector} < \mbox{std} \\ :: \mbox{size} \ \ t > \&\mbox{dims})$ 

Schmidt probabilities of the bi-partite pure state A.

 $\bullet \ \ \mathsf{template} \mathord{<} \mathsf{typename} \ \mathsf{Derived} >$ 

 $\label{lem:double qpp::entanglement} double \ qpp::entanglement \ (const \ Eigen::MatrixBase < Derived > \&A, \ const \ std::vector < std::size\_t > \&dims) \\ Entanglement \ of \ the \ bi-partite \ pure \ state \ A.$ 

template<typename Derived >

double <a href="mailto:qpp::gconcurrence">qpp::gconcurrence</a> (const Eigen::MatrixBase</a> Derived > &A)

G-concurrence of the bi-partite pure state A.

 $\bullet \ \ \text{template}{<} \text{typename Derived}>$ 

double qpp::negativity (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size\_t > &dims)

Negativity of the bi-partite mixed state A.

template<typename Derived >

double qpp::lognegativity (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size\_t > &dims)

Logarithmic negativity of the bi-partite mixed state A.

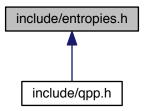
template<typename Derived >

double <a href="mailto:qpp::concurrence">qpp::concurrence</a> (const Eigen::MatrixBase</a> Derived > &A)

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

# 8.11 include/entropies.h File Reference

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



# **Namespaces**

• qpp

#### **Functions**

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

Shannon/von-Neumann entropy of the probability distribution/density matrix A.

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

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

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

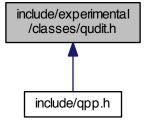
Tsallis-  $\alpha$  entropy of the probability distribution/density matrix A, for  $\alpha \geq 0$ 

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

Quantum mutual information between 2 subsystems of a composite system.

# 8.12 include/experimental/classes/qudit.h File Reference

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



#### Classes

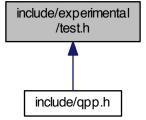
· class qpp::experimental::Qudit

# **Namespaces**

- qpp
- · qpp::experimental

# 8.13 include/experimental/test.h File Reference

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



# **Namespaces**

- qpp::experimental
- dbb

#### **Functions**

template<typename Derived1 , typename Derived2 >
 DynMat< typename Derived1::Scalar > qpp::experimental::apply (const Eigen::MatrixBase< Derived1 >
 &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< std::size\_t > &subsys, const std
 ::vector< std::size\_t > &dims)

Applies the gate A to the part subsys of a multipartite state vector or density matrix.

• template<typename Derived >

cmat qpp::experimental::channel (const Eigen::MatrixBase< Derived > &rho, const std::vector< cmat > &Ks, const std::vector< std::size\_t > &subsys, const std::vector< std::size\_t > &dims)

Applies the channel specified by the set of Kraus operators Ks to the part of the density matrix rho specified by subsys.

cmat qpp::experimental::super (const std::vector< cmat > &Ks)

Superoperator matrix representation.

• template<typename Derived >

DynMat< typename Derived::Scalar > qpp::experimental::CTRL (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size\_t > &ctrl, const std::vector< std::size\_t > &subsys, std::size\_t n, std::size\_t d=2)

Generates the multipartite multiple-controlled-A gate in matrix form.

cmat qpp::experimental::choi (const std::vector< cmat > &Ks)

Choi matrix representation.

std::vector< cmat > qpp::experimental::randkraus (std::size\_t n, std::size\_t D)

Generates a set of random Kraus operators.

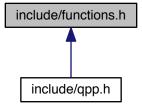
• template<typename Derived >

double <a href="mailto:qpp::experimental::renyi\_inf">qpp::experimental::renyi\_inf</a> (const Eigen::MatrixBase< Derived > &A)

Renyi- ∞ entropy (min entropy) of the probability distribution/density matrix A.

#### 8.14 include/functions.h File Reference

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



#### **Namespaces**

qpp

#### **Functions**

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

Transpose. • template<typename Derived > DynMat< typename Derived::Scalar > qpp::conjugate (const Eigen::MatrixBase< Derived > &A) Complex conjugate. template<typename Derived > DynMat< typename Derived::Scalar > qpp::adjoint (const Eigen::MatrixBase< Derived > &A) Adjoint. template<typename Derived > DynMat< typename Derived::Scalar > qpp::inverse (const Eigen::MatrixBase< Derived > &A) template<typename Derived > Derived::Scalar qpp::trace (const Eigen::MatrixBase< Derived > &A) template<typename Derived > Derived::Scalar qpp::det (const Eigen::MatrixBase< Derived > &A) Determinant. template<typename Derived > Derived::Scalar <a href="mailto:qpp::logdet">qpp::logdet</a> (const Eigen::MatrixBase</a> 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 <a href="mailto:open:prod">open:prod</a> (const Eigen::MatrixBase</a> Derived > &A) Element-wise product of A. template<typename Derived > double <a href="mailto:qpp::norm">qpp::norm</a> (const Eigen::MatrixBase< Derived > &A) Frobenius norm. template<typename Derived > DynColVect< cplx > qpp::evals (const Eigen::MatrixBase< Derived > &A) Eigenvalues. • template<typename Derived > cmat qpp::evects (const Eigen::MatrixBase< Derived > &A) Eigenvectors. template<typename Derived > DynColVect< 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 > DynColVect< 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 absolut 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)

    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 >

   DynMat< typename Derived::Scalar > qpp::powm (const Eigen::MatrixBase< Derived > &A, std::size_t n)
          Matrix power.

    template<typename Derived >

   double qpp::schatten (const Eigen::MatrixBase< Derived > &A, std::size_t p)
• template<typename OutputScalar , typename Derived >
   \label{eq:const_power_problem} \begin{tabular}{ll} DynMat &< Output Scalar > qpp::cwise (const Eigen::MatrixBase &< Derived > \&A, Output Scalar (*f)(const type-type) (const Eigen::MatrixBase) (const
   name Derived::Scalar &))
          Functor.
• template<typename T >
   DynMat< typename T::Scalar > qpp::kron (const T &head)
          Kronecker product (variadic overload)
• template<typename T , typename... Args>
   DynMat< typename T::Scalar > qpp::kron (const T &head, const Args &...tail)
          Kronecker product (variadic overload)
• template<typename Derived >
   DynMat< typename Derived::Scalar > qpp::kron (const std::vector< Derived > &As)
          Kronecker product (std::vector overload)
• template<typename Derived >
   DynMat< typename Derived::Scalar > qpp::kron (const std::initializer_list< Derived > &As)
          Kronecker product (std::initializer_list overload)

    template<typename Derived >

   DynMat< typename Derived::Scalar > qpp::kronpow (const Eigen::MatrixBase< Derived > &A, std::size t
   n)
          Kronecker power.

    template<typename Derived >

   DynMat< typename Derived::Scalar > qpp::reshape (const Eigen::MatrixBase< Derived > &A, std::size_t
   rows, std::size_t cols)
          Reshape.

    template<typename Derived1 , typename Derived2 >

   DynMat< typename Derived1::Scalar > qpp::comm (const Eigen::MatrixBase< Derived1 > &A, const
```

Eigen::MatrixBase< Derived2 > &B)

Commutator.

• template<typename Derived1 , typename Derived2 >

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

Anti-commutator.

template<typename Derived >

DynMat< typename Derived::Scalar > qpp::prj (const Eigen::MatrixBase< Derived > &V)

Projector.

• template<typename Derived >

DynMat< typename Derived::Scalar > qpp::grams (const std::vector< Derived > &Vs)

Gram-Schmidt orthogonalization (std::vector overload)

template<typename Derived >

DynMat< typename Derived::Scalar > qpp::grams (const std::initializer list< Derived > &Vs)

Gram-Schmidt orthogonalization (std::initializer list overload)

template<typename Derived >

DynMat< typename Derived::Scalar > qpp::grams (const Eigen::MatrixBase< Derived > &A)

Gram-Schmidt orthogonalization (Eigen expression (matrix) overload)

std::vector< std::size\_t > qpp::n2multiidx (std::size\_t n, const std::vector< std::size\_t > &dims)

Non-negative integer index to multi-index.

• std::size\_t qpp::multiidx2n (const std::vector< std::size\_t > &midx, const std::vector< std::size\_t > &dims)

Multi-index to non-negative integer index.

ket qpp::mket (const std::vector < std::size\_t > &mask, const std::vector < std::size\_t > &dims)

Multi-partite qudit ket (different dimensions overload)

ket qpp::mket (const std::vector< std::size\_t > &mask, std::size\_t d=2)

Multi-partite qudit ket (same dimensions overload)

cmat qpp::mprj (const std::vector< std::size\_t > &mask, const std::vector< std::size\_t > &dims)

Projector onto multi-partite qudit ket (different dimensions overload)

cmat qpp::mprj (const std::vector < std::size\_t > &mask, std::size\_t d=2)

Projector onto multi-partite qudit ket (same dimensions overload)

std::vector< std::size\_t > qpp::invperm (const std::vector< std::size\_t > &perm)

Inverse permutation.

std::vector< std::size\_t > app::compperm (const std::vector< std::size\_t > aperm, const std::vector< std
 ::size\_t > aperm, const std::vector< std
 ::size\_t

Compose permutations.

template<typename InputIterator >

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

Computes the absolut values squared of a range of complex numbers.

template<typename Derived >

std::vector< double > qpp::abssq (const Eigen::MatrixBase< Derived > &V)

Computes the absolut values squared of a column vector.

template<typename InputIterator >

auto qpp::sum (InputIterator first, InputIterator last) -> typename InputIterator::value\_type

Element-wise sum of a range.

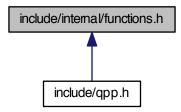
 $\bullet \ \ \text{template}{<} \text{typename InputIterator} >$ 

auto qpp::prod (InputIterator first, InputIterator last) -> typename InputIterator::value\_type

Element-wise product of a range.

#### 8.15 include/internal/functions.h File Reference

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



#### **Namespaces**

- · qpp::internal
- qpp

#### **Functions**

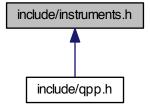
- void qpp::internal::\_n2multiidx (std::size\_t n, std::size\_t numdims, const std::size\_t \*dims, std::size\_t \*result)
- std::size\_t app::internal::\_multiidx2n (const std::size\_t \*midx, std::size\_t numdims, const std::size\_t \*dims)
- template<typename Derived >
   bool qpp::internal::\_check\_square\_mat (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\_row\_vector (const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >
   bool qpp::internal::\_check\_col\_vector (const Eigen::MatrixBase< Derived > &A)
- template<typename T >
   bool qpp::internal::\_check\_nonzero\_size (const T &x)
- bool qpp::internal::\_check\_dims (const std::vector< std::size\_t > &dims)
- template<typename Derived >
   bool qpp::internal::\_check\_dims\_match\_mat (const std::vector< std::size\_t > &dims, const Eigen::Matrix
   Base< Derived > &A)
- template<typename Derived >
   bool qpp::internal::\_check\_dims\_match\_cvect (const std::vector< std::size\_t > &dims, const Eigen::Matrix
   Base< Derived > &V)
- template<typename Derived >
   bool qpp::internal::\_check\_dims\_match\_rvect (const std::vector< std::size\_t > &dims, const Eigen::Matrix
   Base< Derived > &V)
- bool qpp::internal::\_check\_eq\_dims (const std::vector< std::size\_t > &dims, std::size\_t dim)
- bool qpp::internal::\_check\_subsys\_match\_dims (const std::vector< std::size\_t > &subsys, const std
   ::vector< std::size\_t > &dims)
- bool qpp::internal::\_check\_perm (const std::vector< std::size\_t > &perm)
- template<typename Derived1 , typename Derived2 >
   DynMat< typename Derived1::Scalar > qpp::internal::\_kron2 (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)
```

#### 8.16 include/instruments.h File Reference

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



#### **Namespaces**

• qpp

#### **Functions**

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

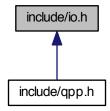
Measures the state A using the set of Kraus operators Ks (std::initializer\_list overload)

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

Measures the state A in the orthonormal basis specified by the eigenvectors of M.

#### 8.17 include/io.h File Reference

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



#### **Namespaces**

• qpp

#### **Functions**

template<typename InputIterator >
 void qpp::disp (const InputIterator &first, const InputIterator &last, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

Displays a range. Does not add a newline.

template<typename InputIterator >
 std::ostream & qpp::displn (const InputIterator &first, const InputIterator &last, const std::string &separator,
 const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

Displays a range. Adds a newline.

template<typename T >

std::ostream & qpp::disp (const T &x, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

Displays a standard container that supports std::begin, std::end and forward iteration. Does not add a newline.

template<typename T >
 std::ostream & qpp::displn (const T &x, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

Displays a standard container that supports std::begin, std::end and forward iteration. Adds a newline.

 $\bullet \ \ template {<} typename \ T >$ 

std::ostream & qpp::disp (const T \*x, const std::size\_t n, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

Displays a C-style array. Does not add a newline.

• template<typename T >

std::ostream & qpp::displn (const T \*x, const std::size\_t n, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

Displays a C-style array. Adds a newline.

template<typename Derived >

 $std::ostream \ \& \ qpp::disp \ (const \ Eigen::MatrixBase < \ Derived > \&A, \ double \ chop=qpp::chop, \ std::ostream \ \&os=std::cout)$ 

Displays an Eigen expression in matrix friendly form. Does not add a new line.

template<typename Derived >
 std::ostream & qpp::displn (const Eigen::MatrixBase< Derived > &A, double chop=qpp::chop, std::ostream &os=std::cout)

Displays an Eigen expression in matrix friendly form. Adds a newline.

std::ostream & qpp::disp (const cplx z, double chop=qpp::chop, std::ostream &os=std::cout)

Displays a number (implicitly converted to std::complex<double>) in friendly form. Does not add a new line.

std::ostream & qpp::displn (const cplx z, double chop=qpp::chop, std::ostream &os=std::cout)

Displays a number (implicitly converted to std::complex<double>) in friendly form. Adds a new line.

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.

 $\bullet \ \ \mathsf{template} \mathord{<} \mathsf{typename} \ \mathsf{Derived} >$ 

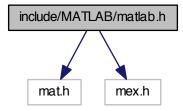
DynMat< typename Derived::Scalar > qpp::load (const std::string &fname)

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

#### 8.18 include/MATLAB/matlab.h File Reference

```
#include "mat.h"
#include "mex.h"
```

Include dependency graph for matlab.h:



# **Namespaces**

• qpp

#### **Functions**

template<typename Derived >
 Derived qpp::loadMATLABmatrix (const std::string &mat\_file, const std::string &var\_name)
 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 > &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, 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)

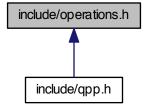
template<>

void qpp::saveMATLABmatrix (const Eigen::MatrixBase< 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)

# 8.19 include/operations.h File Reference

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



#### Namespaces

• qpp

#### **Functions**

template<typename Derived1, typename Derived2 >
 DynMat< typename Derived1::Scalar > qpp::applyCTRL (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< std::size\_t > &ctrl, const std::vector< std::size\_t > &subsys, std::size\_t n, std::size\_t d=2)

Applies the controlled-gate A to the part subsys of a multipartite state vector or density matrix.

template<typename Derived1, typename Derived2 >
 DynMat< typename Derived1::Scalar > qpp::apply (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< std::size t > &subsys, std::size t n, std::size t d=2)

Applies the gate A to the part subsys of a multipartite state vector or density matrix.

template<typename Derived >
 cmat qpp::channel (const Eigen::MatrixBase< Derived > &rho, const std::vector< cmat > &Ks)

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

template<typename Derived >
 cmat qpp::channel (const Eigen::MatrixBase< Derived > &rho, const std::vector< cmat > &Ks, const std
 ::vector< std::size\_t > &subsys, std::size\_t n, std::size\_t d=2)

Applies the channel specified by the set of Kraus operators Ks to the part of the density matrix rho specified by subsys.

cmat qpp::super (const std::vector< cmat > &Ks)

Superoperator matrix representation.

cmat qpp::choi (const std::vector< cmat > &Ks)

Choi matrix representation.

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

Extracts orthogonal Kraus operators from Choi matrix.

• template<typename Derived >

Partial trace.

template<typename Derived >

```
DynMat< typename Derived::Scalar > qpp::ptrace2 (const Eigen::MatrixBase< Derived > &A, const std⇔ ::vector< std::size_t > &dims)
```

Partial trace.

• template<typename Derived >

```
\label{lem:def:DynMat} \mbox{DynMat} < \mbox{typename Derived::Scalar} > \mbox{qpp::ptrace (const Eigen::MatrixBase} < \mbox{Derived} > \&\mbox{A, const std} :::vector < \mbox{std::size\_t} > \&\mbox{subsys, const std::vector} < \mbox{std::size\_t} > \&\mbox{dims})
```

Partial trace.

• template<typename Derived >

Partial transpose.

ullet template<typename Derived >

```
DynMat< typename Derived::Scalar > qpp::syspermute (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &perm, const std::vector< std::size_t > &dims)
```

System permutation.

# 8.20 include/qpp.h File Reference

```
#include <algorithm>
#include <chrono>
#include <cmath>
#include <complex>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <exception>
#include <fstream>
#include <functional>
#include <initializer_list>
#include <iomanip>
#include <iostream>
#include <iterator>
#include <limits>
#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 "constants.h"
#include "types.h"
#include "classes/exception.h"
#include "classes/singleton.h"
#include "classes/states.h"
#include "classes/randevs.h"
#include "internal/functions.h"
#include "classes/init.h"
#include "functions.h"
#include "classes/gates.h"
#include "classes/codes.h"
#include "operations.h"
#include "entropies.h"
#include "io.h"
#include "entanglement.h"
#include "instruments.h"
#include "random.h"
#include "classes/timer.h"
#include "experimental/test.h"
#include "experimental/classes/qudit.h"
Include dependency graph for qpp.h:
```

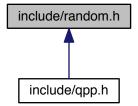
#### **Namespaces**

qpp

#### **Variables**

#### 8.21 include/random.h File Reference

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



#### **Namespaces**

qpp

#### **Functions**

```
    template<typename Derived >
        Derived qpp::rand (std::size_t rows, std::size_t cols, double a=0, double b=1)
            Generates a random matrix with entries uniformly distributed in the interval [a, b)
    template<>
            dmat qpp::rand (std::size_t rows, std::size_t 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 (std::size_t rows, std::size_t 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)
    double qpp::rand (double a=0, double b=1)
            Generates a random real number uniformly distributed in the interval [a, b)
```

int qpp::randint (int a=std::numeric\_limits< int >::min(), int b=std::numeric\_limits< int >::max())

Generates a random integer (int) uniformly distributed in the interval [a, b].

• template<typename Derived >

Derived <a href="mailto:qpp::randn">qpp::randn</a> (std::size\_t rows, std::size\_t cols, double mean=0, double sigma=1)

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

template<>

dmat qpp::randn (std::size\_t rows, std::size\_t cols, double mean, double sigma)

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

template<>

cmat qpp::randn (std::size\_t rows, std::size\_t cols, double mean, double sigma)

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

• double <a href="mailto:qpp::randn">qpp::randn</a> (double mean=0, double sigma=1)

Generates a random real number (double) normally distributed in N(mean, sigma)

cmat qpp::randU (std::size\_t D)

Generates a random unitary matrix.

cmat qpp::randV (std::size\_t Din, std::size\_t Dout)

Generates a random isometry matrix.

• std::vector< cmat > qpp::randkraus (std::size t N, std::size t D)

Generates a set of random Kraus operators.

cmat qpp::randH (std::size\_t D)

Generates a random Hermitian matrix.

ket qpp::randket (std::size\_t D)

Generates a random normalized ket (pure state vector)

cmat qpp::randrho (std::size t D)

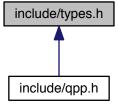
Generates a random density matrix.

std::vector< std::size\_t > qpp::randperm (std::size\_t n)

Generates a random uniformly distributed permutation.

# 8.22 include/types.h File Reference

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



#### **Namespaces**

qpp

# **Typedefs**

```
using qpp::cplx = std::complex < double >
      Complex number in double precision.
template<typename Scalar >
  using <a href="mailto:qpp::DynMat">qpp::DynMat</a> = Eigen::Matrix< Scalar, Eigen::Dynamic, Eigen::Dynamic >
     Dynamic Eigen matrix over the field specified by Scalar.
• template<typename Scalar >
  using qpp::DynColVect = Eigen::Matrix < Scalar, Eigen::Dynamic, 1 >
     Dynamic Eigen column vector over the field specified by Scalar.
\bullet \ \ \text{template}{<} \text{typename Scalar} >
  using qpp::DynRowVect = Eigen::Matrix< Scalar, 1, Eigen::Dynamic >
     Dynamic Eigen row vector over the field specified by Scalar.
using qpp::ket = DynColVect< cplx >
      Complex (double precision) dynamic Eigen column vector.
using qpp::bra = DynRowVect< cplx >
      Complex (double precision) dynamic Eigen row vector.
using qpp::cmat = DynMat< cplx >
      Complex (double precision) dynamic Eigen matrix.
using qpp::dmat = DynMat< double >
     Real (double precision) dynamic Eigen matrix.
```

# 8.23 mainpage.dox File Reference

# Index

absm	qpp::Exception, 109
qpp, 22	DIMS_MISMATCH_VECTOR
abssq	qpp::Exception, 109
qpp, 23	DIMS_NOT_EQUAL
adjoint	qpp::Exception, 109
qpp, 24	det
anticomm	
	qpp, 36
qpp, 24	disp
apply	qpp, 36–38
qpp, 25	displn
	qpp, 39–41
bra	dmat
qpp, 21	qpp, <mark>21</mark>
CUSTOM_EXCEPTION	ee
qpp::Exception, 110	qpp, <mark>92</mark>
channel	entanglement
qpp, 27, 28	qpp, 42
choi	eps
qpp, 29	qpp, 92
choi2kraus	evals
	qpp, 43
qpp, 30	evects
chop	
qpp, 92	qpp, 44
cmat	expm
qpp, 21	qpp, 44
codes	EN/E OUDIT
qpp, 92	FIVE_QUBIT
comm	qpp::Codes, 106
qpp, 31	funm
compperm	qpp, <del>45</del>
qpp, 31	
concurrence	gconcurrence
qpp, 33	qpp, <del>45</del>
	grams
conjugate	qpp, 46, 47
qpp, 33	gt
cosm	qpp, <mark>92</mark>
qpp, 35	
cplx	hevals
qpp, 21	qpp, 48
cwise	hevects
qpp, 35	qpp, 48
	11 1 /
DIMS_INVALID	infty
qpp::Exception, 109	qpp, 92
DIMS_MISMATCH_CVECTOR	init
qpp::Exception, 109	qpp, 93
DIMS_MISMATCH_MATRIX	inverse
qpp::Exception, 109 DIMS MISMATCH RVECTOR	qpp, 49 invperm

INDEX 157

App. 49		
ket	qpp, 49	
kron	ket	
qpp, 50, 51         omega           kronpow         qpp, 62           qpp, 52         PERM_INVALID           opp, 52         pi           logdet         qpp, 93           qpp, 54         powm           logme qpp, 54         prj           lognegativity         qpp, 63           qpp, 55         prod           MATRIX_MISMATCH_SUBSYS         prace           qpp, 64         ptrace           qpp, 65         ptrace           qpp, 65         ptrace           qpp, 66         ptrace1           qpp, 66         ptrace2           qpp, 67         ptrace1           qpp, 68         ptrace2           qpp, 67         ptrace2           qpp:Exception, 109         qpp, 67           MATRIX_NOT_SQUARE         ptrace2           qpp, 67         qpp, 68           MATRIX_NOT_SQUARE_OR_CVECTOR         qpp, 69           qpp:Exception, 109         qpp, 69           MATRIX_NOT_SQUARE_OR_VECTOR         absm, 22           qpp:Exception, 109         adjoint, 24           qpp:Exception, 109         adjoint, 24           qpp:Exception, 109         adjoint, 24           qpp:Exception, 109		
Rronpow   App, 52   PERM_INVALID		
PERM_INVALID		•
load		<b>4ρρ</b> , <b>02</b>
logdet	466, 02	PERM_INVALID
logdet         qpp, 54           logm         qpp, 63           powm         qpp, 63           qpp, 54         prj           lognegativity         qpp, 63           qpp, 55         prod           MATRIX_MISMATCH_SUBSYS         ptrace           qpp:Exception, 109         qpp, 65           MATRIX_NOT_CVECTOR         ptrace1           qpp:Exception, 109         qpp, 66           MATRIX_NOT_RVECTOR         ptrace2           qpp:Exception, 109         qpp, 67           MATRIX_NOT_SQUARE         ptraspose           qpp:Exception, 109         qpp, 68           MATRIX_NOT_SQUARE_OR_CVECTOR         qpp, 69           qpp:Exception, 109         qpp, 69           MATRIX_NOT_SQUARE_OR_VECTOR         qpp, 69           qpp:Exception, 109         adjoint, 24           adpp:Exception, 109         adjoint, 24           app, 50         anticomm, 24           app, 50         app, 50           measure         channel, 27, 28           qpp, 58         choi, 29           mpr         codes, 92           comm, 31         conp. 92           cmt, 21         comp. 92           cmt, 21         comp. 92 <td></td> <td>qpp::Exception, 109</td>		qpp::Exception, 109
logm		
logm qpp, 54 prj qpp, 63 prj qpp, 63 qpp, 55 prod qpp, 64 ptrace qpp, 65 prod qpp, 64 ptrace qpp, 65 qpp, 65 ptrace qpp, 65 ptrace qpp, 66 ptrace1 qpp, 66 ptrace2 qpp, 67 ptrace3 qpp, 67 ptrace4 qpp, 66 ptrace4 qpp, 66 ptrace5 qpp; Exception, 109 qpp, 67 ptrace5 qpp; Exception, 109 qpp, 68 ptrace5 qpp; Exception, 109 qmutualinfo qpp, 68 ptrace5 qpp; Exception, 109 qpp, 69 qpp, 69 qpp, 69 qpp, 69 qpp; Exception, 109 qpp, 13 absm, 22 qpp, 69 qpp; Exception, 109 qpp, 13 absm, 22 qpp, 69 qpp; Exception, 109 absm, 22 qpp, 25 ptrace5 qpp; Exception, 109 anticomm, 24 apply, 25 ptrace5 qpp, 56, 57 qpp, 58 ptrace5 qpp, 58 ptrace5 qpp, 58 ptrace5 qpp, 58 qpp, 59 qpp, 59 qpp, 59 qpp, 59 qpp, 59 qpp, 59 qpp, 60 qpp; Exception, 100 qpp; Exception, 100 qpp; 50 qpp, 60 qpp; 50 q	-	
qpp, 54         prj           lognegativity         qpp, 63           qpp, 55         prod           dpp, 64         pp, 64           MATRIX_MISMATCH_SUBSYS         ptrace           qpp:Exception, 109         qpp, 65           MATRIX_NOT_CVECTOR         ptrace1           qpp:Exception, 109         qpp, 66           MATRIX_NOT_RVECTOR         ptrace2           qpp:Exception, 109         qpp, 67           MATRIX_NOT_SQUARE         ptraspose           qpp:Exception, 109         qpp, 68           MATRIX_NOT_SQUARE_OR_CVECTOR         qpp, 69           qpp:Exception, 109         qpp, 69           MATRIX_NOT_SQUARE_OR_VECTOR         absm, 22           qpp:Exception, 109         absm, 22           MATRIX_NOT_VECTOR         adjoint, 24           qpp, 93         bra, 21           measure         channel, 27, 28           qpp, 56, 57         channel, 27, 28           mkt         choi, 29           pprj         cmat, 21           qpp, 60         comm, 31           concurrence, 33         conjugate, 33           npp, 60         comperm, 31           non_cobes, 106         cwise, 35           NO_CODEWORD		•
lognegativity         qpp, 63           qpp, 55         prod           MATRIX_MISMATCH_SUBSYS         ptrace           qpp::Exception, 109         qpp, 65           MATRIX_NOT_CVECTOR         ptrace1           qpp::Exception, 109         qpp, 66           MATRIX_NOT_RVECTOR         ptrace2           qpp::Exception, 109         qpp, 67           MATRIX_NOT_SQUARE         ptranspose           qpp::Exception, 109         qpp, 68           MATRIX_NOT_SQUARE_OR_CVECTOR         qpp, 69           qpp::Exception, 109         qpp, 13           MATRIX_NOT_SQUARE_OR_VECTOR         absm, 22           qpp::Exception, 109         absm, 22           MATRIX_NOT_VECTOR         adjoint, 24           qpp::Exception, 109         anticomm, 24           maxn         apply, 25           pap; 3         bra, 21           measure         channel, 27, 28           qpp, 58         choi, 29           mpri         cmat, 21           qpp, 59         codes, 92           multiidx2n         comm, 31           qpp, 60         comm, 31           NO_EDEWORD         det, 36           qpp::Exception, 110         disp, 36–38	-	
qpp, 55         prod           MATRIX_MISMATCH_SUBSYS         ptrace           qpp:Exception, 109         qpp, 65           MATRIX_NOT_CVECTOR         ptrace1           qpp:Exception, 109         qpp, 66           MATRIX_NOT_RVECTOR         ptrace2           qpp;Exception, 109         qpp, 67           MATRIX_NOT_SQUARE         ptranspose           qpp:Exception, 109         qpp, 68           MATRIX_NOT_SQUARE_OR_CVECTOR         qpp, 69           qpp:Exception, 109         qpp, 69           MATRIX_NOT_SQUARE_OR_VECTOR         qpp, 69           qpp:Exception, 109         absq, 23           MATRIX_NOT_VECTOR         adjoint, 24           qpp:Exception, 109         anticomm, 24           maxn         apply, 25           dpp, 3         bra, 21           channel, 27, 28         choi, 29           mket         choizkraus, 30           dpp, 58         choi, 29           multiidx2n         comm, 31           qpp, 60         comm, 31           conpperm, 31         concurrence, 33           comm, 31         conjugate, 33           cosm, 35         conjugate, 33           cosm, 35         conjugate, 33		
MATRIX_MISMATCH_SUBSYS         ptrace           qpp::Exception, 109         qpp, 65           MATRIX_NOT_CVECTOR         ptrace1           qpp; (6)         qpp, 66           MATRIX_NOT_RVECTOR         ptrace2           qpp::Exception, 109         qpp, 67           MATRIX_NOT_SQUARE         ptranspose           qpp::Exception, 109         qpp, 68           MATRIX_NOT_SQUARE_OR_CVECTOR         qpp, 69           qpp::Exception, 109         qpp, 69           MATRIX_NOT_SQUARE_OR_VECTOR         qpp, 69           qpp::Exception, 109         absm, 22           absm, 22         absm, 22           qpp::Exception, 109         actionmm, 24           maxn         apply, 25           pp, 33         bra, 21           measure         choi, 29           qpp, 56, 57         choi, 29           mket         dpp, 58           mprj         cmat, 21           qpp, 59         codes, 92           multiidx2n         compperm, 31           qpp, 60         compperm, 31           n2multiidx         conjugate, 33           qpp, 60         compperm, 31           NO_CODEWORD         det, 36           qpp:Exception, 110		
qpp:Exception, 109         qpp, 65           MATRIX_NOT_CVECTOR         ptrace1           qpp:Exception, 109         qpp, 66           MATRIX_NOT_RVECTOR         ptrace2           qpp:Exception, 109         qpp, 67           MATRIX_NOT_SQUARE         qpp, 68           qpp:Exception, 109         qpp, 68           MATRIX_NOT_SQUARE_OR_CVECTOR         qpp, 69           qpp:Exception, 109         qpp, 69           MATRIX_NOT_SQUARE_OR_VECTOR         qpp, 69           qpp:Exception, 109         absm, 22           dpp:Exception, 109         abssq, 23           MATRIX_NOT_VECTOR         adjoint, 24           qpp:Exception, 109         anticomm, 24           maxn         apply, 25           bra, 21         channel, 27, 28           choi, 29         choizkraus, 30           measure         choizkraus, 30           qpp, 58         choi, 29           mpri         cmat, 21           qpp, 60         comm, 31           compperm, 31         concurrence, 33           n2multiidx         conjugate, 33           qpp, 60         cosm, 35           NINE_QUBIT_SHOR         cpix, 21           qpp:Exception, 110         det, 36		qpp, 64
MATRIX_NOT_CVECTOR         ptrace1           qpp:Exception, 109         qpp, 66           MATRIX_NOT_RVECTOR         ptrace2           qpp:Exception, 109         qpp, 67           MATRIX_NOT_SQUARE         ptranspose           qpp:Exception, 109         qpp, 68           MATRIX_NOT_SQUARE_OR_CVECTOR         qpp, 69           qpp:Exception, 109         qpp, 69           MATRIX_NOT_SQUARE_OR_VECTOR         qpp, 69           qpp:Exception, 109         absm, 22           MATRIX_NOT_VECTOR         adjoint, 24           qpp:Exception, 109         anticomm, 24           maxn         apply, 25           pp, 93         bra, 21           measure         choi.2 29           qpp, 56, 57         choi.2 29           mket         choi.2 29           qpp, 58         chop, 92           mpri         codes, 92           qpp, 60         comm, 31           npp, 60         compperm, 31           n2multiidx         conpperm, 31           qpp:Codes, 106         cosm, 35           NO_CODEWORD         det, 36           qpp:Exception, 110         dispn, 39-41           NOT_BIPARTITE         qpp:Exception, 110         ee, 92		•
qpp::Exception, 109         qpp, 66           MATRIX_NOT_RVECTOR         ptrace2           qpp::Exception, 109         qpp, 67           MATRIX_NOT_SQUARE         ptranspose           qpp::Exception, 109         qpp, 68           MATRIX_NOT_SQUARE_OR_CVECTOR         qpp, 69           qpp::Exception, 109         qpp, 69           MATRIX_NOT_SQUARE_OR_VECTOR         qpp, 13           qpp::Exception, 109         absm, 22           dpp::Exception, 109         adjoint, 24           qpp, 93         adjoint, 24           qpp, 93         anticomm, 24           maxn         apply, 25           dpp, 58         choi, 29           mori         choi, 29           mket         choiZkraus, 30           dpp, 58         choi, 29           mpri         cmat, 21           qpp, 59         codes, 92           multiidx2n         comm, 31           qpp, 60         compperm, 31           n2multiidx         conjugate, 33           qpp, 60         cosm, 35           NO_COBEWORD         det, 36           qpp::Exception, 110         disp, 36–38           NOT_BIPARTITE         displn, 39–41           qpp::Exception, 110		
MATRIX_NOT_RVECTOR         ptrace2           qpp::Exception, 109         qpp, 67           MATRIX_NOT_SQUARE         ptranspose           qpp::Exception, 109         qpp, 68           MATRIX_NOT_SQUARE_OR_CVECTOR         qpp, 69           qpp::Exception, 109         qmutualinfo           MATRIX_NOT_SQUARE_OR_RVECTOR         qpp, 69           qpp::Exception, 109         abssq, 23           MATRIX_NOT_VECTOR         adjoint, 24           qpp::Exception, 109         anticomm, 24           maxn         apply, 25           pp, 93         bra, 21           measure         choi, 29           qpp, 56, 57         choi2kraus, 30           ropp, 58         choi, 29           multidx2n         comm, 31           qpp, 60         comm, 31           concurrence, 33         conjugate, 33           qpp, 60         compperm, 31           NINE_QUBIT_SHOR         cpix, 21           qpp::Codes, 106         cwise, 35           NO_CODEWORD         det, 36           qpp::Exception, 110         disp, 36–38           NOT_BIPARTITE         displn, 39–41           qpp::Exception, 110         eps, 92           norticity         evels, 43 <td></td> <td>·</td>		·
qpp::Exception, 109         qpp, 67           MATRIX_NOT_SQUARE qpp::Exception, 109         qpp, 68           MATRIX_NOT_SQUARE_OR_CVECTOR qpp::Exception, 109         qmutualinfo qpp, 69           MATRIX_NOT_SQUARE_OR_RVECTOR qpp::Exception, 109         qpp, 13           MATRIX_NOT_SQUARE_OR_VECTOR qpp::Exception, 109         absm, 22 abssq, 23           MATRIX_NOT_VECTOR qpp::Exception, 109         adjoint, 24 anticomm, 24 apply, 25 bra, 21 channel, 27, 28 choi, 29 choi, 29           maxn qpp, 56, 57         channel, 27, 28 choi, 29 choi, 29 cmat, 21 codes, 92 cmat, 21 qpp, 59 codes, 92 cmm, 31 compperm, 31 concurrence, 33 qpp, 60         comm, 31 compperm, 31 concurrence, 33 cosm, 35 compperm, 31 concurrence, 33 cosm, 35 cosm, 21 det, 36 disp, 36–38 displn, 39–41 disp, 36–38 displn, 39–41 dmat, 21 epp::Exception, 110 dmat, 21 epp::Exception, 110 dmat, 21 epp::Exception, 110 central general, 42 epp, 92 epp::Exception, 110 negativity evects, 44		
MATRIX_NOT_SQUARE qpp::Exception, 109         qpp, 68           MATRIX_NOT_SQUARE_OR_CVECTOR qpp::Exception, 109         qmutualinfo           MATRIX_NOT_SQUARE_OR_RVECTOR qpp::Exception, 109         qpp, 69           MATRIX_NOT_SQUARE_OR_VECTOR qpp::Exception, 109         absm, 22           MATRIX_NOT_VECTOR qpp::Exception, 109         anticomm, 24           maxn qpp, 93         bra, 21           measure qpp, 56, 57         channel, 27, 28           dpp, 58         cho; 29           mprj qpp, 59         codes, 92           multiidx2n qpp, 60         comm, 31           n2multiidx qpp, 60         compperm, 31           NINE_QUBIT_SHOR qpp::Codes, 106         cpix, 21           NO_CODEWORD qpp::Exception, 110         det, 36           NOT_BIPARTITE qpp::Exception, 110         disp, 36–38           NOT_QUBIT_GATE qpp::Exception, 110         ee, 92           NOT_QUBIT_GATE qpp::Exception, 110         evals, 43           NOT_QUBIT_SUBSYS qpp::Exception, 110         evals, 43           negativity         evets, 44		•
MATRIX_NOT_SQUARE_OR_CVECTOR         qpp::Exception, 109         qmutualinfo           MATRIX_NOT_SQUARE_OR_RVECTOR         qpp, 69         qpp, 69           qpp::Exception, 109         qpp, 13         absm, 22           qpp::Exception, 109         abssq, 23         adjoint, 24           qpp::Exception, 109         anticomm, 24         anticomm, 24           maxn         apply, 25         bra, 21           measure         channel, 27, 28         choi, 29           qpp, 56, 57         choi, 29         choizkraus, 30           dpp, 58         chop, 92         cmat, 21           qpp, 59         codes, 92           multiidx2n         comm, 31         concurrence, 33           n2multiidx         conjugate, 33         conm, 31           qpp, 60         cosm, 35         coliquate, 33           NINE_QUBIT_SHOR         cplx, 21         compperm, 31           qpp::Codes, 106         cosm, 35           NO_CODEWORD         det, 36         disp, 36–38           Qpp::Exception, 110         disp, 36–38           NOT_QUBIT_GATE         ee, 92         eps, 92           qpp::Exception, 110         entanglement, 42           NOT_QUBIT_SUBSYS         eps, 92           qpp:Exception, 110 <td>MATRIX_NOT_SQUARE</td> <td></td>	MATRIX_NOT_SQUARE	
qpp::Exception, 109         qmutualinfo           MATRIX_NOT_SQUARE_OR_RVECTOR         qpp, 69           qpp::Exception, 109         qpp, 13           MATRIX_NOT_SQUARE_OR_VECTOR         absm, 22           qpp::Exception, 109         adjoint, 24           maxn         apply, 25           dpp, 93         bra, 21           measure         channel, 27, 28           dpp, 56, 57         choi, 29           mket         choi2kraus, 30           dpp, 58         chop, 92           mprj         cmat, 21           dpp, 59         codes, 92           multiidx2n         comm, 31           dpp, 60         compperm, 31           n2multiidx         conjugate, 33           dpp, 60         cosm, 35           NINE_QUBIT_SHOR         cplx, 21           dpp::Codes, 106         cwise, 35           NO_CODEWORD         det, 36           dpp::Exception, 110         displn, 39-41           NOT_BIPARTITE         displn, 39-41           dpp::Exception, 110         eps, 92           qpp:Exception, 110         entanglement, 42           NOT_QUBIT_SUBSYS         eps, 92           qpp:Exception, 110         evals, 43		qpp, 68
MATRIX_NOT_SQUARE_OR_RVECTOR         qpp, 69           qpp::Exception, 109         qpp, 13           MATRIX_NOT_SQUARE_OR_VECTOR         absm, 22           qpp::Exception, 109         adjoint, 24           qpp::Exception, 109         anticomm, 24           maxn         apply, 25           qpp, 93         bra, 21           measure         channel, 27, 28           qpp, 56, 57         choi, 29           mket         choi2kraus, 30           qpp, 58         chop, 92           mprj         cmat, 21           qpp, 59         codes, 92           multiidx2n         comm, 31           qpp, 60         compperm, 31           n2multiidx         conjugate, 33           qpp, 60         cosm, 35           NINE_QUBIT_SHOR         cplx, 21           qpp::Codes, 106         cwise, 35           NO_CODEWORD         det, 36           qpp::Exception, 110         displn, 39–41           NOT_BIPARTITE         displn, 39–41           qpp::Exception, 110         eps, 92           qpp::Exception, 110         entanglement, 42           NOT_QUBIT_SUBSYS         eps, 92           qpp::Exception, 110         evals, 43           <		ament alimfa
qpp::Exception, 109         qpp, 13           MATRIX_NOT_SQUARE_OR_VECTOR         absm, 22           qpp::Exception, 109         adjoint, 24           MATRIX_NOT_VECTOR         adjoint, 24           qpp::Exception, 109         anticomm, 24           maxn         apply, 25           dpp, 93         bra, 21           measure         channel, 27, 28           dpp, 56, 57         choi, 29           mket         choi2kraus, 30           dpp, 58         chop, 92           mprj         cmat, 21           dpp, 59         codes, 92           multiidxn         comm, 31           dpp, 60         compperm, 31           concurrence, 33         conjugate, 33           dpp, 60         cosm, 35           NINE_QUBIT_SHOR         cplx, 21           dpp::Codes, 106         cwise, 35           NO_CODEWORD         det, 36           dpp::Exception, 110         dispn, 36–38           NOT_BIPARTITE         displn, 39–41           dpp::Exception, 110         ee, 92           npp::Exception, 110         entanglement, 42           NOT_QUBIT_SUBSYS         eps, 92           qpp::Exception, 110         evals, 43           ne		·
MATRIX_NOT_SQUARE_OR_VECTOR         absm, 22           qpp::Exception, 109         abssq, 23           MATRIX_NOT_VECTOR         adjoint, 24           qpp::Exception, 109         anticomm, 24           maxn         apply, 25           qpp, 93         bra, 21           measure         channel, 27, 28           qpp, 56, 57         choi, 29           mket         choizkraus, 30           qpp, 58         chop, 92           mprj         cmat, 21           qpp, 59         codes, 92           multiidx2n         comm, 31           qpp, 60         compperm, 31           concurrence, 33         conjugate, 33           qpp, 60         cosm, 35           NINE_QUBIT_SHOR         cplx, 21           qpp::Codes, 106         cwise, 35           NO_CODEWORD         det, 36           qpp::Exception, 110         disp, 36–38           NOT_BIPARTITE         displn, 39–41           qpp::Exception, 110         ee, 92           qpp::Exception, 110         entanglement, 42           NOT_QUBIT_SUBSYS         eps, 92           qpp::Exception, 110         evals, 43           negativity         evects, 44		
qpp::Exception, 109       abssq, 23         MATRIX_NOT_VECTOR       adjoint, 24         qpp::Exception, 109       anticomm, 24         maxn       apply, 25         qpp, 93       bra, 21         measure       channel, 27, 28         qpp, 56, 57       choi, 29         mket       choi2kraus, 30         qpp, 58       chop, 92         mprj       cmat, 21         qpp, 59       codes, 92         multiidx2n       comm, 31         qpp, 60       comperm, 31         n2multiidx       conjugate, 33         qpp, 60       cosm, 35         NINE_QUBIT_SHOR       cplx, 21         qpp::Codes, 106       cwise, 35         NO_CODEWORD       det, 36         qpp::Exception, 110       disp, 36–38         NOT_BIPARTITE       displn, 39–41         qpp::Exception, 110       entanglement, 42         NOT_QUBIT_GATE       ee, 92         qpp::Exception, 110       entanglement, 42         NOT_QUBIT_SUBSYS       eps, 92         qpp::Exception, 110       evals, 43         negativity       evects, 44		
MATRIX_NOT_VECTOR       adjoint, 24         qpp::Exception, 109       anticomm, 24         maxn       apply, 25         pp, 93       bra, 21         measure       channel, 27, 28         qpp, 56, 57       choi, 29         mket       choi/2kraus, 30         qpp, 58       chop, 92         mprj       cmat, 21         qpp, 59       codes, 92         multiidx2n       compperm, 31         qpp, 60       compperm, 31         n2multiidx       conjugate, 33         qpp, 60       cosm, 35         NINE_QUBIT_SHOR       cplx, 21         qpp::Codes, 106       cwise, 35         NO_CODEWORD       det, 36         qpp::Exception, 110       disp, 36–38         NOT_BIPARTITE       displn, 39–41         qpp::Exception, 110       ee, 92         qpp::Exception, 110       entanglement, 42         NOT_QUBIT_SUBSYS       eps, 92         qpp::Exception, 110       evals, 43         negativity       evects, 44		
maxn       apply, 25         qpp, 93       bra, 21         measure       channel, 27, 28         qpp, 56, 57       choi, 29         mket       choi2kraus, 30         qpp, 58       chop, 92         mprj       cmat, 21         qpp, 59       codes, 92         multiidx2n       comm, 31         qpp, 60       compperm, 31         NINE_QUBIT_SHOR       cplx, 21         qpp::Codes, 106       cwise, 35         NO_CODEWORD       det, 36         qpp::Exception, 110       disp, 36–38         NOT_BIPARTITE       displn, 39–41         qpp::Exception, 110       dmat, 21         NOT_QUBIT_GATE       ee, 92         qpp::Exception, 110       entanglement, 42         NOT_QUBIT_SUBSYS       eps, 92         qpp::Exception, 110       evals, 43         negativity       evects, 44		adjoint, 24
qpp, 93       bra, 21         measure       channel, 27, 28         qpp, 56, 57       choi, 29         mket       choi2kraus, 30         qpp, 58       chop, 92         mprj       cmat, 21         qpp, 59       codes, 92         multiidx2n       compperm, 31         qpp, 60       compperm, 31         n2multiidx       conjugate, 33         qpp, 60       cosm, 35         NINE_QUBIT_SHOR       cplx, 21         qpp::Codes, 106       cwise, 35         NO_CODEWORD       det, 36         qpp::Exception, 110       displn, 39–41         NOT_BIPARTITE       displn, 39–41         qpp::Exception, 110       dmat, 21         NOT_QUBIT_GATE       ee, 92         qpp::Exception, 110       entanglement, 42         NOT_QUBIT_SUBSYS       eps, 92         qpp::Exception, 110       evals, 43         negativity       evects, 44	qpp::Exception, 109	
measure channel, 27, 28		
qpp, 56, 57       choi, 29         mket       choi2kraus, 30         qpp, 58       chop, 92         mprj       cmat, 21         qpp, 59       codes, 92         multiidx2n       comm, 31         qpp, 60       compperm, 31         n2multiidx       conjugate, 33         qpp, 60       cosm, 35         NINE_QUBIT_SHOR       cplx, 21         qpp::Codes, 106       cwise, 35         NO_CODEWORD       det, 36         qpp::Exception, 110       disp, 36–38         NOT_BIPARTITE       displn, 39–41         qpp::Exception, 110       dmat, 21         NOT_QUBIT_GATE       ee, 92         qpp::Exception, 110       entanglement, 42         NOT_QUBIT_SUBSYS       eps, 92         qpp::Exception, 110       evals, 43         negativity       evets, 44		
mket       choizkraus, 30         qpp, 58       chop, 92         mprj       cmat, 21         qpp, 59       codes, 92         multiidx2n       comm, 31         qpp, 60       concurrence, 33         n2multiidx       conjugate, 33         qpp, 60       cosm, 35         NINE_QUBIT_SHOR       cplx, 21         qpp::Codes, 106       cwise, 35         NO_CODEWORD       det, 36         qpp::Exception, 110       dispn, 36–38         NOT_BIPARTITE       displn, 39–41         qpp::Exception, 110       dmat, 21         NOT_QUBIT_GATE       ee, 92         qpp::Exception, 110       entanglement, 42         NOT_QUBIT_SUBSYS       eps, 92         qpp::Exception, 110       evals, 43         negativity       evects, 44		
qpp, 58       chop, 92         mprj       cmat, 21         qpp, 59       codes, 92         multiidx2n       comm, 31         qpp, 60       compperm, 31         n2multiidx       conjugate, 33         qpp, 60       cosm, 35         NINE_QUBIT_SHOR       cplx, 21         qpp::Codes, 106       cwise, 35         NO_CODEWORD       det, 36         qpp::Exception, 110       disp, 36–38         NOT_BIPARTITE       displn, 39–41         qpp::Exception, 110       dmat, 21         NOT_QUBIT_GATE       ee, 92         qpp::Exception, 110       entanglement, 42         NOT_QUBIT_SUBSYS       eps, 92         qpp::Exception, 110       evals, 43         negativity       evects, 44		
mprj       cmat, 21         qpp, 59       codes, 92         multiidx2n       comm, 31         qpp, 60       compperm, 31         n2multiidx       conjugate, 33         qpp, 60       cosm, 35         NINE_QUBIT_SHOR       cplx, 21         qpp::Codes, 106       cwise, 35         NO_CODEWORD       det, 36         qpp::Exception, 110       displ, 36–38         NOT_BIPARTITE       displn, 39–41         qpp::Exception, 110       dmat, 21         NOT_QUBIT_GATE       ee, 92         qpp::Exception, 110       entanglement, 42         NOT_QUBIT_SUBSYS       eps, 92         qpp::Exception, 110       evals, 43         negativity       evects, 44		
multiidx2n       comm, 31         qpp, 60       compperm, 31         n2multiidx       concurrence, 33         qpp, 60       cosm, 35         NINE_QUBIT_SHOR       cplx, 21         qpp::Codes, 106       cwise, 35         NO_CODEWORD       det, 36         qpp::Exception, 110       disp, 36–38         NOT_BIPARTITE       displn, 39–41         qpp::Exception, 110       dmat, 21         NOT_QUBIT_GATE       ee, 92         qpp::Exception, 110       entanglement, 42         NOT_QUBIT_SUBSYS       eps, 92         qpp::Exception, 110       evals, 43         negativity       evects, 44		cmat, 21
qpp, 60       compperm, 31         n2multiidx       concurrence, 33         qpp, 60       cosm, 35         NINE_QUBIT_SHOR       cplx, 21         qpp::Codes, 106       cwise, 35         NO_CODEWORD       det, 36         qpp::Exception, 110       disp, 36–38         NOT_BIPARTITE       displn, 39–41         qpp::Exception, 110       dmat, 21         NOT_QUBIT_GATE       ee, 92         qpp::Exception, 110       entanglement, 42         NOT_QUBIT_SUBSYS       eps, 92         qpp::Exception, 110       evals, 43         negativity       evects, 44	qpp, 59	
n2multiidx conjugate, 33 qpp, 60 cosm, 35 NINE_QUBIT_SHOR cplx, 21 qpp::Codes, 106 cwise, 35 NO_CODEWORD det, 36 qpp::Exception, 110 disp, 36–38 NOT_BIPARTITE displn, 39–41 qpp::Exception, 110 dmat, 21 NOT_QUBIT_GATE ee, 92 qpp::Exception, 110 entanglement, 42 NOT_QUBIT_SUBSYS eps, 92 qpp::Exception, 110 evals, 43 negativity evects, 44		· ·
n2multiidx       conjugate, 33         qpp, 60       cosm, 35         NINE_QUBIT_SHOR       cplx, 21         qpp::Codes, 106       cwise, 35         NO_CODEWORD       det, 36         qpp::Exception, 110       displn, 39–41         NOT_BIPARTITE       displn, 39–41         qpp::Exception, 110       dmat, 21         NOT_QUBIT_GATE       ee, 92         qpp::Exception, 110       entanglement, 42         NOT_QUBIT_SUBSYS       eps, 92         qpp::Exception, 110       evals, 43         negativity       evects, 44	qpp, 60	
qpp, 60       cosm, 35         NINE_QUBIT_SHOR       cplx, 21         qpp::Codes, 106       cwise, 35         NO_CODEWORD       det, 36         qpp::Exception, 110       displn, 39–41         NOT_BIPARTITE       displn, 39–41         qpp::Exception, 110       dmat, 21         NOT_QUBIT_GATE       ee, 92         qpp::Exception, 110       entanglement, 42         NOT_QUBIT_SUBSYS       eps, 92         qpp::Exception, 110       evals, 43         negativity       evects, 44	n2multiidx	
NINE_QUBIT_SHOR       cplx, 21         qpp::Codes, 106       cwise, 35         NO_CODEWORD       det, 36         qpp::Exception, 110       disp, 36–38         NOT_BIPARTITE       displn, 39–41         qpp::Exception, 110       dmat, 21         NOT_QUBIT_GATE       ee, 92         qpp::Exception, 110       entanglement, 42         NOT_QUBIT_SUBSYS       eps, 92         qpp::Exception, 110       evals, 43         negativity       evects, 44		
NO_CODEWORD         det, 36           qpp::Exception, 110         disp, 36–38           NOT_BIPARTITE         displn, 39–41           qpp::Exception, 110         dmat, 21           NOT_QUBIT_GATE         ee, 92           qpp::Exception, 110         entanglement, 42           NOT_QUBIT_SUBSYS         eps, 92           qpp::Exception, 110         evals, 43           negativity         evects, 44		
qpp::Exception, 110       disp, 36–38         NOT_BIPARTITE       displn, 39–41         qpp::Exception, 110       dmat, 21         NOT_QUBIT_GATE       ee, 92         qpp::Exception, 110       entanglement, 42         NOT_QUBIT_SUBSYS       eps, 92         qpp::Exception, 110       evals, 43         negativity       evects, 44	qpp::Codes, 106	cwise, 35
NOT_BIPARTITE         displn, 39-41           qpp::Exception, 110         dmat, 21           NOT_QUBIT_GATE         ee, 92           qpp::Exception, 110         entanglement, 42           NOT_QUBIT_SUBSYS         eps, 92           qpp::Exception, 110         evals, 43           negativity         evects, 44	<del>-</del>	
qpp::Exception, 110 dmat, 21  NOT_QUBIT_GATE ee, 92 qpp::Exception, 110 entanglement, 42  NOT_QUBIT_SUBSYS eps, 92 qpp::Exception, 110 evals, 43 negativity evects, 44		•
NOT_QUBIT_GATE ee, 92 qpp::Exception, 110 entanglement, 42 NOT_QUBIT_SUBSYS eps, 92 qpp::Exception, 110 evals, 43 negativity evects, 44	<del>-</del>	•
qpp::Exception, 110 entanglement, 42  NOT_QUBIT_SUBSYS eps, 92 qpp::Exception, 110 evals, 43 negativity evects, 44		,
NOT_QUBIT_SUBSYS eps, 92 qpp::Exception, 110 evals, 43 negativity evects, 44		*
negativity evects, 44		
	qpp::Exception, 110	
qpp, 61 expm, 44		
	dbb' ₽ l	expm, 44

158 INDEX

funm, 45	trace, 90
gconcurrence, 45	transpose, 91
grams, 46, 47	tsallis, 91
gt, 92	qpp::Codes
hevals, 48	FIVE_QUBIT, 106
hevects, 48	NINE_QUBIT_SHOR, 106
infty, 92	SEVEN_QUBIT_STEANE, 106
init, 93	qpp::Exception
inverse, 49	CUSTOM_EXCEPTION, 110
invperm, 49	DIMS_INVALID, 109
ket, 22	DIMS_MISMATCH_CVECTOR, 109
kron, 50, 51	DIMS_MISMATCH_MATRIX, 109
kronpow, 52	DIMS_MISMATCH_RVECTOR, 109
load, 52	DIMS_MISMATCH_VECTOR, 109
logdet, 54	DIMS_NOT_EQUAL, 109
logm, 54	MATRIX_MISMATCH_SUBSYS, 109
lognegativity, 55	MATRIX_NOT_CVECTOR, 109
maxn, 93	MATRIX_NOT_RVECTOR, 109
measure, 56, 57	MATRIX_NOT_SQUARE, 109
mket, 58	MATRIX_NOT_SQUARE_OR_CVECTOR, 109
mprj, 59	MATRIX_NOT_SQUARE_OR_RVECTOR, 109
multiidx2n, 60	MATRIX_NOT_SQUARE_OR_VECTOR, 109
n2multiidx, 60	MATRIX_NOT_VECTOR, 109
negativity, 61	NO_CODEWORD, 110
norm, 62	NOT_BIPARTITE, 110
omega, 62	NOT_QUBIT_GATE, 110
pi, 93	NOT_QUBIT_SUBSYS, 110
powm, 63	OUT_OF_RANGE, 110
prj, 63	PERM_INVALID, 109
prod, 64	SUBSYS_MISMATCH_DIMS, 109
ptrace, 65	TYPE_MISMATCH, 110
ptrace1, 66	UNDEFINED_TYPE, 110
ptrace2, 67	UNKNOWN_EXCEPTION, 109
ptranspose, 68	ZERO_SIZE, 109
qmutualinfo, 69	rand
rand, 70, 71	qpp, 70, 71
randint, 72	randint
randket, 73	qpp, 72
randkraus, 73	randket
randn, 74, 75	qpp, 73
randperm, 76	randkraus
randrho, 76	qpp, 73
rdevs, 93	randn
renyi, 77	qpp, 74, 75
reshape, 78	randperm
save, 78	qpp, 76
schatten, 80	randrho
schmidtcoeff, 80	qpp, 76
schmidtprob, 81	rdevs
shannon, 84	qpp, 93
sinm, 85	renyi
spectralpowm, 85	qpp, 77
sqrtm, 86	reshape
st, 93	qpp, 78
sum, 86, 87	חרוי
super, 87	SEVEN_QUBIT_STEANE
svals, 88	qpp::Codes, 106
syspermute, 89	SUBSYS_MISMATCH_DIMS
• •	

```
qpp::Exception, 109
save
    qpp, 78
schatten
    qpp, 80
schmidtcoeff
    qpp, 80
schmidtprob
    qpp, 81
shannon
    qpp, 84
sinm
    qpp, 85
spectralpowm
    qpp, 85
sqrtm
    qpp, 86
st
    qpp, 93
sum
    qpp, 86, 87
super
    qpp, 87
svals
    qpp, 88
syspermute
    qpp, <mark>89</mark>
TYPE_MISMATCH
    qpp::Exception, 110
trace
    qpp, 90
transpose
    qpp, 91
tsallis
    qpp, <mark>91</mark>
UNDEFINED_TYPE
    qpp::Exception, 110
UNKNOWN_EXCEPTION
    qpp::Exception, 109
ZERO_SIZE
    qpp::Exception, 109
```