

quantum++
0.1

Generated by Doxygen 1.8.7

Tue Nov 11 2014 22:38:57

Contents

1	Quantum++ - A C++11 quantum computing library	1
2	Namespace Index	5
2.1	Namespace List	5
3	Hierarchical Index	7
3.1	Class Hierarchy	7
4	Class Index	9
4.1	Class List	9
5	File Index	11
5.1	File List	11
6	Namespace Documentation	13
6.1	qpp Namespace Reference	13
6.1.1	Typedef Documentation	21
6.1.1.1	bra	21
6.1.1.2	cmat	22
6.1.1.3	cplx	22
6.1.1.4	dmat	22
6.1.1.5	DynColVect	22
6.1.1.6	DynMat	22
6.1.1.7	DynRowVect	22
6.1.1.8	ket	22
6.1.2	Function Documentation	22
6.1.2.1	absm	22
6.1.2.2	abssq	23
6.1.2.3	abssq	23
6.1.2.4	adjoint	24
6.1.2.5	anticomm	24
6.1.2.6	apply	25
6.1.2.7	apply	26

6.1.2.8	<code>applyCTRL</code>	27
6.1.2.9	<code>applyCTRL</code>	28
6.1.2.10	<code>channel</code>	29
6.1.2.11	<code>channel</code>	30
6.1.2.12	<code>channel</code>	31
6.1.2.13	<code>choi</code>	32
6.1.2.14	<code>choi2kraus</code>	33
6.1.2.15	<code>comm</code>	33
6.1.2.16	<code>compperm</code>	34
6.1.2.17	<code>concurrence</code>	34
6.1.2.18	<code>conjugate</code>	35
6.1.2.19	<code>contfrac2x</code>	36
6.1.2.20	<code>contfrac2x</code>	37
6.1.2.21	<code>cosm</code>	37
6.1.2.22	<code>cwise</code>	37
6.1.2.23	<code>det</code>	38
6.1.2.24	<code>disp</code>	38
6.1.2.25	<code>disp</code>	39
6.1.2.26	<code>disp</code>	39
6.1.2.27	<code>disp</code>	39
6.1.2.28	<code>disp</code>	40
6.1.2.29	<code>entanglement</code>	41
6.1.2.30	<code>evals</code>	42
6.1.2.31	<code>evecs</code>	42
6.1.2.32	<code>expm</code>	43
6.1.2.33	<code>funm</code>	44
6.1.2.34	<code>gcd</code>	44
6.1.2.35	<code>gcd</code>	45
6.1.2.36	<code>gconcurrence</code>	45
6.1.2.37	<code>grams</code>	46
6.1.2.38	<code>grams</code>	46
6.1.2.39	<code>grams</code>	47
6.1.2.40	<code>hevals</code>	47
6.1.2.41	<code>hevecs</code>	48
6.1.2.42	<code>inverse</code>	49
6.1.2.43	<code>invperm</code>	50
6.1.2.44	<code>kron</code>	50
6.1.2.45	<code>kron</code>	51
6.1.2.46	<code>kron</code>	51
6.1.2.47	<code>kron</code>	52

6.1.2.48	kronpow	52
6.1.2.49	lcm	53
6.1.2.50	lcm	53
6.1.2.51	load	54
6.1.2.52	loadMATLABmatrix	54
6.1.2.53	loadMATLABmatrix	54
6.1.2.54	loadMATLABmatrix	55
6.1.2.55	logdet	55
6.1.2.56	logm	56
6.1.2.57	lognegativity	56
6.1.2.58	measure	57
6.1.2.59	measure	58
6.1.2.60	measure	59
6.1.2.61	mket	60
6.1.2.62	mket	61
6.1.2.63	mprj	61
6.1.2.64	mprj	62
6.1.2.65	multiidx2n	62
6.1.2.66	n2multiidx	63
6.1.2.67	negativity	64
6.1.2.68	norm	65
6.1.2.69	omega	66
6.1.2.70	operator""_i	66
6.1.2.71	operator""_i	66
6.1.2.72	powm	66
6.1.2.73	prj	67
6.1.2.74	prod	68
6.1.2.75	prod	68
6.1.2.76	ptrace	69
6.1.2.77	ptrace1	70
6.1.2.78	ptrace2	71
6.1.2.79	ptranspose	72
6.1.2.80	qmutualinfo	73
6.1.2.81	rand	74
6.1.2.82	rand	74
6.1.2.83	rand	75
6.1.2.84	rand	75
6.1.2.85	randH	76
6.1.2.86	randint	76
6.1.2.87	randket	77

6.1.2.88	randkraus	77
6.1.2.89	randn	78
6.1.2.90	randn	78
6.1.2.91	randn	79
6.1.2.92	randn	79
6.1.2.93	randperm	80
6.1.2.94	randrho	80
6.1.2.95	randU	81
6.1.2.96	randV	81
6.1.2.97	renyi	82
6.1.2.98	reshape	82
6.1.2.99	save	83
6.1.2.100	saveMATLABmatrix	83
6.1.2.101	saveMATLABmatrix	83
6.1.2.102	saveMATLABmatrix	84
6.1.2.103	schatten	84
6.1.2.104	schmidtcoeff	85
6.1.2.105	schmidtprob	86
6.1.2.106	schmidtU	87
6.1.2.107	schmidtV	88
6.1.2.108	shannon	89
6.1.2.109	sinm	90
6.1.2.110	spectralpowm	90
6.1.2.111	sqrtn	91
6.1.2.112	sum	91
6.1.2.113	sum	92
6.1.2.114	super	92
6.1.2.115	svals	93
6.1.2.116	svdU	93
6.1.2.117	svdV	94
6.1.2.118	syspermute	94
6.1.2.119	trace	95
6.1.2.120	transpose	96
6.1.2.121	tsallis	96
6.1.2.122	x2contfrac	97
6.1.3	Variable Documentation	97
6.1.3.1	chop	97
6.1.3.2	codes	97
6.1.3.3	ee	97
6.1.3.4	eps	98

6.1.3.5	gt	98
6.1.3.6	infty	98
6.1.3.7	init	98
6.1.3.8	maxn	98
6.1.3.9	pi	98
6.1.3.10	rdevs	98
6.1.3.11	st	98
6.2	qpp::experimental Namespace Reference	98
6.2.1	Detailed Description	100
6.2.2	Function Documentation	100
6.2.2.1	apply	100
6.2.2.2	apply_old	101
6.2.2.3	applyCTRL	102
6.2.2.4	channel	103
6.2.2.5	channel_old	104
6.2.2.6	choi	105
6.2.2.7	CTRL	106
6.2.2.8	disp	106
6.2.2.9	disp	107
6.2.2.10	disp	107
6.2.2.11	disp	108
6.2.2.12	disp	108
6.2.2.13	displn	109
6.2.2.14	displn	109
6.2.2.15	displn	110
6.2.2.16	displn	111
6.2.2.17	displn	111
6.2.2.18	randkraus	112
6.2.2.19	renyi_inf	113
6.2.2.20	super	114
6.3	qpp::internal Namespace Reference	115
6.3.1	Detailed Description	115
6.3.2	Function Documentation	116
6.3.2.1	_check_col_vector	116
6.3.2.2	_check_dims	116
6.3.2.3	_check_dims_match_cvect	116
6.3.2.4	_check_dims_match_mat	116
6.3.2.5	_check_dims_match_rvect	116
6.3.2.6	_check_eq_dims	116
6.3.2.7	_check_nonzero_size	116

6.3.2.8	_check_perm	116
6.3.2.9	_check_row_vector	116
6.3.2.10	_check_square_mat	116
6.3.2.11	_check_subsys_match_dims	116
6.3.2.12	_check_vector	116
6.3.2.13	_kron2	116
6.3.2.14	_multiidx2n	116
6.3.2.15	_n2multiidx	116
6.3.2.16	variadic_vector_emplace	116
6.3.2.17	variadic_vector_emplace	117
7	Class Documentation	119
7.1	qpp::Codes Class Reference	119
7.1.1	Detailed Description	120
7.1.2	Member Enumeration Documentation	120
7.1.2.1	Type	120
7.1.3	Constructor & Destructor Documentation	120
7.1.3.1	Codes	120
7.1.4	Member Function Documentation	120
7.1.4.1	codeword	120
7.1.5	Friends And Related Function Documentation	121
7.1.5.1	internal::Singleton< const Codes >	121
7.2	qpp::Exception Class Reference	121
7.2.1	Detailed Description	123
7.2.2	Member Enumeration Documentation	123
7.2.2.1	Type	123
7.2.3	Constructor & Destructor Documentation	124
7.2.3.1	Exception	124
7.2.3.2	Exception	124
7.2.4	Member Function Documentation	125
7.2.4.1	_construct_exception_msg	125
7.2.4.2	what	125
7.2.5	Member Data Documentation	125
7.2.5.1	_custom	125
7.2.5.2	_msg	125
7.2.5.3	_type	125
7.2.5.4	_where	125
7.3	qpp::Gates Class Reference	125
7.3.1	Detailed Description	127
7.3.2	Constructor & Destructor Documentation	127

7.3.2.1	Gates	127
7.3.3	Member Function Documentation	128
7.3.3.1	CTRL	128
7.3.3.2	expandout	128
7.3.3.3	Fd	129
7.3.3.4	Id	130
7.3.3.5	Rn	130
7.3.3.6	Xd	130
7.3.3.7	Zd	131
7.3.4	Friends And Related Function Documentation	131
7.3.4.1	internal::Singleton< const Gates >	131
7.3.5	Member Data Documentation	131
7.3.5.1	CNOTab	132
7.3.5.2	CNOTba	132
7.3.5.3	CZ	132
7.3.5.4	FRED	132
7.3.5.5	H	132
7.3.5.6	Id2	132
7.3.5.7	S	132
7.3.5.8	SWAP	132
7.3.5.9	T	132
7.3.5.10	TOF	132
7.3.5.11	X	132
7.3.5.12	Y	132
7.3.5.13	Z	133
7.4	qpp::Init Class Reference	133
7.4.1	Detailed Description	134
7.4.2	Constructor & Destructor Documentation	134
7.4.2.1	Init	134
7.4.2.2	~Init	134
7.4.3	Friends And Related Function Documentation	134
7.4.3.1	internal::Singleton< const Init >	134
7.5	qpp::internal::IOManipEigen Class Reference	134
7.5.1	Constructor & Destructor Documentation	135
7.5.1.1	IOManipEigen	135
7.5.1.2	IOManipEigen	135
7.5.2	Friends And Related Function Documentation	135
7.5.2.1	operator<<	135
7.5.3	Member Data Documentation	135
7.5.3.1	_A	135

7.5.3.2	<code>_chop</code>	135
7.6	<code>qpp::internal::IOManipPointer< PointerType ></code> Class Template Reference	135
7.6.1	Constructor & Destructor Documentation	136
7.6.1.1	<code>IOManipPointer</code>	136
7.6.1.2	<code>IOManipPointer</code>	137
7.6.2	Member Function Documentation	137
7.6.2.1	<code>operator=</code>	137
7.6.3	Friends And Related Function Documentation	137
7.6.3.1	<code>operator<<</code>	137
7.6.4	Member Data Documentation	137
7.6.4.1	<code>_end</code>	137
7.6.4.2	<code>_n</code>	137
7.6.4.3	<code>_p</code>	137
7.6.4.4	<code>_separator</code>	137
7.6.4.5	<code>_start</code>	137
7.7	<code>qpp::internal::IOManipRange< InputIterator ></code> Class Template Reference	137
7.7.1	Constructor & Destructor Documentation	138
7.7.1.1	<code>IOManipRange</code>	138
7.7.2	Friends And Related Function Documentation	138
7.7.2.1	<code>operator<<</code>	139
7.7.3	Member Data Documentation	139
7.7.3.1	<code>_end</code>	139
7.7.3.2	<code>_first</code>	139
7.7.3.3	<code>_last</code>	139
7.7.3.4	<code>_separator</code>	139
7.7.3.5	<code>_start</code>	139
7.8	<code>qpp::experimental::Qudit</code> Class Reference	139
7.8.1	Constructor & Destructor Documentation	139
7.8.1.1	<code>Qudit</code>	140
7.8.2	Member Function Documentation	140
7.8.2.1	<code>getD</code>	140
7.8.2.2	<code>getRho</code>	140
7.8.2.3	<code>measure</code>	140
7.8.2.4	<code>measure</code>	141
7.8.3	Member Data Documentation	141
7.8.3.1	<code>_D</code>	141
7.8.3.2	<code>_rho</code>	141
7.9	<code>qpp::RandomDevices</code> Class Reference	141
7.9.1	Detailed Description	142
7.9.2	Constructor & Destructor Documentation	142

7.9.2.1	RandomDevices	142
7.9.3	Friends And Related Function Documentation	143
7.9.3.1	internal::Singleton< RandomDevices >	143
7.9.4	Member Data Documentation	143
7.9.4.1	_rd	143
7.9.4.2	_rng	143
7.10	qpp::internal::Singleton< T > Class Template Reference	143
7.10.1	Detailed Description	143
7.10.2	Constructor & Destructor Documentation	144
7.10.2.1	Singleton	144
7.10.2.2	~Singleton	144
7.10.2.3	Singleton	144
7.10.3	Member Function Documentation	144
7.10.3.1	get_instance	144
7.10.3.2	operator=	144
7.11	qpp::States Class Reference	144
7.11.1	Detailed Description	146
7.11.2	Constructor & Destructor Documentation	146
7.11.2.1	States	146
7.11.3	Friends And Related Function Documentation	146
7.11.3.1	internal::Singleton< const States >	146
7.11.4	Member Data Documentation	146
7.11.4.1	b00	146
7.11.4.2	b01	146
7.11.4.3	b10	147
7.11.4.4	b11	147
7.11.4.5	GHZ	147
7.11.4.6	pb00	147
7.11.4.7	pb01	147
7.11.4.8	pb10	147
7.11.4.9	pb11	147
7.11.4.10	pGHZ	147
7.11.4.11	pW	147
7.11.4.12	px0	147
7.11.4.13	px1	147
7.11.4.14	py0	147
7.11.4.15	py1	148
7.11.4.16	pz0	148
7.11.4.17	pz1	148
7.11.4.18	W	148

7.11.4.19	x0	148
7.11.4.20	x1	148
7.11.4.21	y0	148
7.11.4.22	y1	148
7.11.4.23	z0	148
7.11.4.24	z1	148
7.12	qpp::Timer Class Reference	148
7.12.1	Detailed Description	149
7.12.2	Constructor & Destructor Documentation	149
7.12.2.1	Timer	149
7.12.3	Member Function Documentation	149
7.12.3.1	seconds	149
7.12.3.2	tic	149
7.12.3.3	toc	150
7.12.4	Friends And Related Function Documentation	150
7.12.4.1	operator<<	150
7.12.5	Member Data Documentation	150
7.12.5.1	_end	150
7.12.5.2	_start	150
8	File Documentation	151
8.1	include/classes/codes.h File Reference	151
8.2	include/classes/exception.h File Reference	152
8.3	include/classes/gates.h File Reference	152
8.4	include/classes/init.h File Reference	153
8.5	include/classes/random_devices.h File Reference	153
8.6	include/classes/singleton.h File Reference	154
8.7	include/classes/states.h File Reference	155
8.8	include/classes/timer.h File Reference	155
8.9	include/constants.h File Reference	156
8.10	include/entanglement.h File Reference	157
8.11	include/entropies.h File Reference	158
8.12	include/experimental/classes/qudit.h File Reference	159
8.13	include/experimental/test.h File Reference	159
8.14	include/functions.h File Reference	161
8.15	include/internal/functions.h File Reference	165
8.16	include/input_output.h File Reference	166
8.17	include/instruments.h File Reference	167
8.18	include/internal/classes/iomanip.h File Reference	168
8.19	include/MATLAB/matlab.h File Reference	168

8.20	include/number_theory.h File Reference	169
8.21	include/operations.h File Reference	170
8.22	include/qpp.h File Reference	172
8.23	include/random.h File Reference	173
8.24	include/types.h File Reference	174
8.25	mainpage.dox File Reference	175
Index		176

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 "qpp.h"

// #include "MATLAB/matlab.h" // support for MATLAB

using namespace qpp;

int main()
{
    // Qudit Teleportation
    {
        std::size_t D = 3; // size of the system
        std::cout << std::endl << "**** Qudit Teleportation, D = " << D << " ****" << std::endl;
        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);
        mes_AB /= std::sqrt((double) D);
        // circuit that measures in the qudit Bell basis
        cmat Bell_aA = adjoint(gt.CTRL(gt.Xd(D), {0}, {1}, 2, D) *
            kron(gt.Fd(D), gt.Id(D)));
        ket psi_a = randket(D); // random state as input on a
        std::cout << ">> Initial state:" << std::endl;
        std::cout << disp(psi_a) << std::endl;
        ket input_aAB = kron(psi_a, mes_AB); // joint input state aAB
        // output before measurement
        ket output_aAB = apply(input_aAB, Bell_aA, {0, 1}, D);
        auto measured_aA = measure(ptrace2(prj(output_aAB), {D * D, D}),
            gt.Id(D * D)); // measure on aA
        std::discrete_distribution<std::size_t> dd(measured_aA.first.begin(), measured_aA.first.end());
        std::cout << ">> Alice's measurement probabilities: ";
        std::cout << disp(measured_aA.first, " ") << std::endl;
        std::size_t m = dd(rdevs._rng); // sample
        auto midx = n2multiidx(m, {D, D});
        std::cout << ">> Alice's measurement result: ";
        std::cout << disp(midx, " ") << std::endl;
        // conditional result on B before correction
        ket output_m_aAB = apply(output_aAB, prj(mket(midx, D)), {0, 1}, 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}, D);
    }
}
```

```

    cmat rho_B = ptrace1(prj(output_aAB), {D * D, D});
    std::cout << ">> Bob's density operator: " << std::endl;
    std::cout << disp(rho_B) << std::endl;
    std::cout << ">> Norm difference: " << norm(rho_B - prj(psi_a)) << std::endl; //
    verification
}

// Qudit Dense Coding
{
    std::size_t D = 3; // size of the system
    std::cout << std::endl << "**** Qudit Dense Coding, D = " << D << " ****" << std::endl;
    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);
    mes_AB /= std::sqrt((double) D);
    // circuit that measures in the qudit Bell basis
    cmat Bell_AB = adjoint(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: ";
    std::cout << disp(midx, " ") << std::endl;
    // 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}, D);
    // Bob measures the joint system in the qudit Bell basis
    psi_AB = apply(psi_AB, Bell_AB, {0, 1}, D);
    auto measured = measure(psi_AB, gt.Id(D * D));
    std::cout << ">> Bob's measurement probabilities: ";
    std::cout << disp(measured.first, " ") << std::endl;
    // 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: ";
    std::cout << disp(n2multiidx(m_B, {D, D}), " ") << std::endl;
}

// 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;
    std::vector<std::size_t> dims(n, 2); // local dimensions
    std::size_t N = std::pow(2, n); // number of elements in the database
    std::cout << ">> Database size: " << N << 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 << " -> ";
    std::cout << disp(n2multiidx(marked, dims), " ") << std::endl;
    ket psi = mket(n2multiidx(0, dims)); // computational |0>^{\otimes n}
    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++)
    {
        psi(marked) = -psi(marked); // apply the oracle first, no aliasing
        psi = (G * psi).eval(); // then the diffusion operator, no aliasing
    }
    // we now measure the state in the computational basis
    auto measured = measure(psi, gt.Id(N));
    std::cout << ">> Probability of the marked state: " << measured.first[marked] << std::endl;
    std::cout << ">> Probability of all results: ";
    std::cout << disp(measured.first, " ") << std::endl;
    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: ";
    else
        std::cout << ">> Not there yet... we obtained: ";
    std::cout << result << " -> ";
    std::cout << disp(n2multiidx(result, dims), " ") << std::endl;
    // 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;
}

```



```

    cmat rho = 0.2 * st.pb00 + 0.8 * st.pb11;
    std::cout << ">> rho: " << std::endl;
    std::cout << disp(rho) << std::endl;
    std::cout << ">> Concurrence of rho: " << concurrence(rho) << std::endl;
    std::cout << ">> Negativity of rho: " << negativity(rho, {2, 2}) << std::endl;
    std::cout << ">> Logarithmic negativity of rho: " << lognegativity(rho, {2, 2}) <<
std::endl;
    ket psi = 0.8 * mket({0, 0}) + 0.6 * mket({1, 1});
    // apply some local random unitaries
    psi = kron(randU(2), randU(2)) * psi;
    std::cout << ">> psi: " << std::endl;
    std::cout << disp(psi) << std::endl;
    std::cout << ">> Entanglement of psi: " << entanglement(psi, {2, 2}) << std::endl;
    std::cout << ">> Concurrence of psi: " << concurrence(prj(psi)) << std::endl;
    std::cout << ">> G-Concurrence of psi: " << gconcurrence(psi) << std::endl;
    std::cout << ">> Schmidt coefficients of psi: " << std::endl;
    std::cout << disp(schmidtcoeff(psi, {2, 2})) << std::endl;
    std::cout << ">> Schmidt probabilities of psi: " << std::endl;
    std::cout << disp(schmidtprob(psi, {2, 2})) << std::endl;
    cmat U = schmidtU(psi, {2, 2});
    cmat V = schmidtV(psi, {2, 2});
    std::cout << ">> Schmidt vectors on Alice's side: " << std::endl;
    std::cout << disp(U) << std::endl;
    std::cout << ">> Schmidt vectors on Bob's side: " << std::endl;
    std::cout << disp(V) << std::endl;
    std::cout << ">> State psi in the Schmidt basis: " << std::endl;
    std::cout << disp(adjoint(kron(U, V)) * psi) << std::endl;
    // 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;
    std::cout << disp(psi_from_schmidt) << std::endl;
    std::cout << ">> Norm difference: " << norm(psi - psi_from_schmidt) << 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 << ">> [[5, 1, 3]] Five qubit code. ";
    std::cout << "Checking codeword orthogonality." << std::endl;
    std::cout << ">> |<0L|1L>| = ";
    std::cout << disp(adjoint(a0) * a1) << std::endl;
    std::cout << ">> [[7, 1, 3]] Seven qubit Steane code. ";
    std::cout << "Checking codeword orthogonality." << std::endl;
    std::cout << ">> |<0L|1L>| = ";
    std::cout << disp(adjoint(b0) * b1) << std::endl;
    std::cout << ">> [[9, 1, 3]] Nine qubit Shor code. ";
    std::cout << "Checking codeword orthogonality." << std::endl;
    std::cout << ">> |<0L|1L>| = ";
    std::cout << disp(adjoint(c0) * c1) << std::endl;
}

// Timing tests
{
    std::cout << std::endl << "**** Timing tests ****" << std::endl;

    std::size_t n = 12; // number of qubits
    std::size_t N = std::pow(2, n);
    std::vector<std::size_t> dims(n, 2); // local dimensions
    std::cout << ">> n = " << n << " qubits, matrix size " << N << " x " << N << "." << std::endl;
    cmat randcmat = cmat::Random(N, N);

    // ptrace
    std::cout << std::endl << "**** qpp::ptrace() timing ****" << std::endl;
    std::vector<std::size_t> subsys_ptrace = {0};
    std::cout << ">> Subsystem(s): ";
    std::cout << disp(subsys_ptrace, ", ") << std::endl;
    Timer t;
    ptrace(randcmat, subsys_ptrace, dims);
    std::cout << ">> It took " << t.toc() << " seconds." << std::endl;

    // ptranspose
    std::cout << std::endl << "**** qpp::ptranspose() timing ****" << std::endl;
    std::vector<std::size_t> subsys_ptranspose; // partially transpose n-1 subsystems
    for (std::size_t i = 0; i < n - 1; i++)
        subsys_ptranspose.push_back(i);

```

```
std::cout << ">> Subsystem(s): ";
std::cout << disp(subsys_ptranspose, " ", " ") << std::endl;
t.tic();
ptranspose(randcmat, subsys_ptranspose, dims);
std::cout << ">> It took " << t.toc() << " seconds." << std::endl;

// syspermute
std::cout << std::endl << "**** qpp::syspermute() timing ****" << std::endl;
std::vector<std::size_t> perm; // left-shift all subsystems by 1
for (std::size_t i = 0; i < n; i++)
    perm.push_back((i + 1) % n);
std::cout << ">> Subsystem(s): ";
std::cout << disp(perm, " ", " ") << std::endl;
t.tic();
syspermute(randcmat, perm, dims);
std::cout << ">> It took " << t.toc() << " seconds." << std::endl;
}
}
```

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

qpp	13
qpp::experimental	98
qpp::internal	115

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

std::exception	
qpp::Exception	121
qpp::internal::IOManipEigen	134
qpp::internal::IOManipPointer< PointerType >	135
qpp::internal::IOManipRange< InputIterator >	137
qpp::experimental::Qudit	139
qpp::internal::Singleton< T >	143
qpp::internal::Singleton< const Codes >	143
qpp::Codes	119
qpp::internal::Singleton< const Gates >	143
qpp::Gates	125
qpp::internal::Singleton< const Init >	143
qpp::Init	133
qpp::internal::Singleton< const States >	143
qpp::States	144
qpp::internal::Singleton< RandomDevices >	143
qpp::RandomDevices	141
qpp::Timer	148

Chapter 4

Class Index

4.1 Class List

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

qpp::Codes	Const Singleton class that defines quantum error correcting codes	119
qpp::Exception	Generates custom exceptions, used when validating function parameters	121
qpp::Gates	Const Singleton class that implements most commonly used gates	125
qpp::Init	Const Singleton class that performs additional initializations/cleanups	133
qpp::internal::IOManipEigen	134
qpp::internal::IOManipPointer< PointerType >	135
qpp::internal::IOManipRange< InputIterator >	137
qpp::experimental::Qudit	139
qpp::RandomDevices	Singleton class that manages the source of randomness in the library	141
qpp::internal::Singleton< T >	Singleton policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern)	143
qpp::States	Const Singleton class that implements most commonly used states	144
qpp::Timer	Measures time	148

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

include/	constants.h	156
include/	entanglement.h	157
include/	entropies.h	158
include/	functions.h	161
include/	input_output.h	166
include/	instruments.h	167
include/	number_theory.h	169
include/	operations.h	170
include/	qpp.h	172
include/	random.h	173
include/	types.h	174
include/classes/	codes.h	151
include/classes/	exception.h	152
include/classes/	gates.h	152
include/classes/	init.h	153
include/classes/	random_devices.h	153
include/classes/	singleton.h	154
include/classes/	states.h	155
include/classes/	timer.h	155
include/experimental/	test.h	159
include/experimental/classes/	qudit.h	159
include/internal/	functions.h	165
include/internal/classes/	iomanip.h	168
include/MATLAB/	matlab.h	168

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](#)
Singleton 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.
- template<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 >`
`DynColVect< double > schmidtprob (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &dims)`
Schmidt probabilities of the bi-partite pure state A.
- `template<typename Derived >`
`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- α 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- α entropy of the probability distribution/density matrix A , for $\alpha \geq 0$
- Quantum mutual information between 2 subsystems of a composite system.*

 - `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)`
- Transpose.*

 - `template<typename Derived >`
`DynMat< typename Derived::Scalar > transpose (const Eigen::MatrixBase< Derived > &A)`
- Complex conjugate.*

 - `template<typename Derived >`
`DynMat< typename Derived::Scalar > conjugate (const Eigen::MatrixBase< Derived > &A)`
- Adjoint.*

 - `template<typename Derived >`
`DynMat< typename Derived::Scalar > adjoint (const Eigen::MatrixBase< Derived > &A)`
- Inverse.*

 - `template<typename Derived >`
`DynMat< typename Derived::Scalar > inverse (const Eigen::MatrixBase< Derived > &A)`
- Trace.*

 - `template<typename Derived >`
`Derived::Scalar trace (const Eigen::MatrixBase< Derived > &A)`
- Determinant.*

 - `template<typename Derived >`
`Derived::Scalar det (const Eigen::MatrixBase< Derived > &A)`
- Logarithm of the determinant.*

 - `template<typename Derived >`
`Derived::Scalar logdet (const Eigen::MatrixBase< Derived > &A)`
- Element-wise sum of A .*

 - `template<typename Derived >`
`Derived::Scalar sum (const Eigen::MatrixBase< Derived > &A)`
- Element-wise product of A .*

 - `template<typename Derived >`
`Derived::Scalar prod (const Eigen::MatrixBase< Derived > &A)`
- Frobenius norm.*

 - `template<typename Derived >`
`double norm (const Eigen::MatrixBase< Derived > &A)`
- Eigenvalues.*

 - `template<typename Derived >`
`DynColVect< cplx > evals (const Eigen::MatrixBase< Derived > &A)`
- Eigenvectors.*

 - `template<typename Derived >`
`cmat evects (const Eigen::MatrixBase< Derived > &A)`
- Hermitian eigenvalues.*

 - `template<typename Derived >`
`DynColVect< double > hevals (const Eigen::MatrixBase< Derived > &A)`
- Hermitian eigenvectors.*

 - `template<typename Derived >`
`cmat hevects (const Eigen::MatrixBase< Derived > &A)`
- Eigenvalues.*

 - `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)

Matrix sin.

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

Matrix cos.

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

Matrix power.

- `template<typename Derived >`
`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)

Schatten norm.

- `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)
- template<typename InputIterator >
std::vector< double > **abssq** (InputIterator first, InputIterator last)
Computes the absolut values squared of a range of complex numbers.
- template<typename Derived >
std::vector< double > **abssq** (const Eigen::MatrixBase< Derived > &V)
Computes the absolut values squared of a column vector.
- template<typename InputIterator >
auto **sum** (InputIterator first, InputIterator last) -> typename InputIterator::value_type
Element-wise sum of a range.
- template<typename InputIterator >
auto **prod** (InputIterator first, InputIterator last) -> typename InputIterator::value_type
Element-wise product of a range.
- template<typename Derived >
internal::IOManipEigen disp (const Eigen::MatrixBase< Derived > &A, double **chop**=qpp::chop)

- Eigen expression ostream manipulator.*

 - [internal::IOManipEigen disp](#) (cplx z, double chop=[qpp::chop](#))

Complex number ostream manipulator.
- template<typename InputIterator >

[internal::IOManipRange](#)

< InputIterator > [disp](#) (const InputIterator &first, const InputIterator &last, const std::string &separator, const std::string &start="[", const std::string &end="]")

Range ostream manipulator.
- template<typename Container >

[internal::IOManipRange](#)

< typename

Container::const_iterator > [disp](#) (const Container &c, const std::string &separator, const std::string &start="[", const std::string &end="]")

Standard container ostream manipulator. The container must support std::begin(), std::end() and forward iteration.
- template<typename PointerType >

[internal::IOManipPointer](#)

< PointerType > [disp](#) (const PointerType *p, std::size_t n, const std::string &separator, const std::string &start="[", const std::string &end="]")

C-style pointer ostream manipulator.
- template<typename Derived >

void [save](#) (const Eigen::MatrixBase< Derived > &A, const std::string &fname)

Saves Eigen expression to a binary file (internal format) in double precision.
- template<typename Derived >

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

Loads Eigen matrix from a binary file (internal format) in double precision.
- 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 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](#))
- `std::vector< int > x2confrac (double x, std::size_t n, std::size_t cut=1e5)`
Simple continued fraction expansion.
- `double confrac2x (const std::vector< int > &cf, std::size_t n)`
Real representation of a simple continued fraction.
- `double confrac2x (const std::vector< int > &cf)`
Real representation of a simple continued fraction.
- `std::size_t gcd (std::size_t m, std::size_t n)`
Greatest common divisor of two non-negative integers.
- `std::size_t gcd (const std::vector< std::size_t > &ns)`
Greatest common divisor of a list of non-negative integers.
- `std::size_t lcm (std::size_t m, std::size_t n)`
Least common multiple of two positive integers.
- `std::size_t lcm (const std::vector< std::size_t > &ns)`
Least common multiple of a list of positive integers.
- `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 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, const std::vector< std::size_t > &dims)`
Applies the controlled-gate A to the part subsys of a multi-partite state vector or density matrix.
- `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 d=2)`
Applies the controlled-gate A to the part subsys of a multi-partite state vector or density matrix.
- `template<typename Derived1 , typename Derived2 >`
`DynMat< typename Derived1::Scalar > 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 multi-partite state vector or density matrix.
- `template<typename Derived1 , typename Derived2 >`
`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 d=2)`
Applies the gate A to the part subsys of a multi-partite state vector or density matrix.
- `template<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, 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.
- `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 d=2)`

Applies the channel specified by the set of Kraus operators K_s to the part of the density matrix ρ 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 $[a, b]$.

- `template<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 $N(\text{mean}, \text{sigma})$

- `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 $N(\text{mean}, \text{sigma})$, 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 $N(\text{mean}, \text{sigma})$, specialization for complex matrices (`qpp::cmat`)

- double `randn` (double mean=0, double sigma=1)
Generates a random real number (double) normally distributed in $N(\text{mean}, \text{sigma})$
- `cmat randU` (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
 π
- 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, 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.

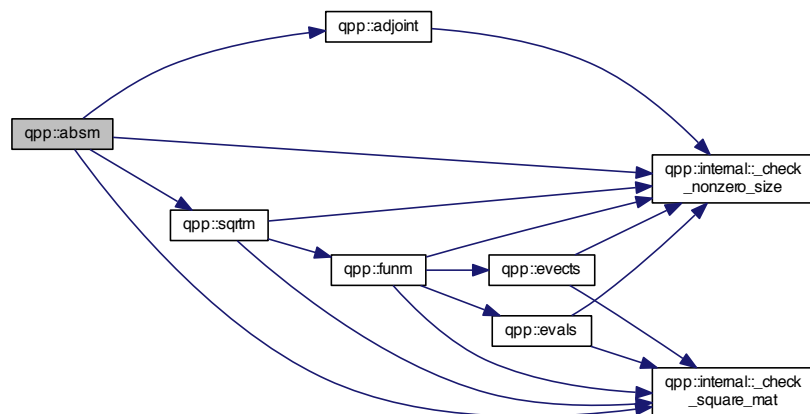
Parameters

A	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

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

Returns

Real vector consisting of the range'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.

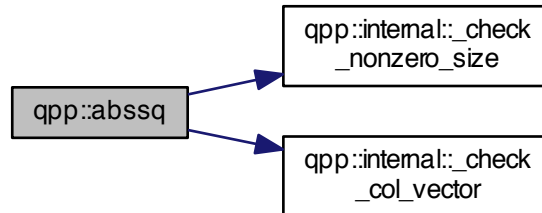
Parameters

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.

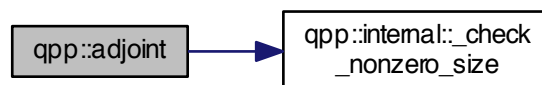
Parameters

<code>A</code>	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

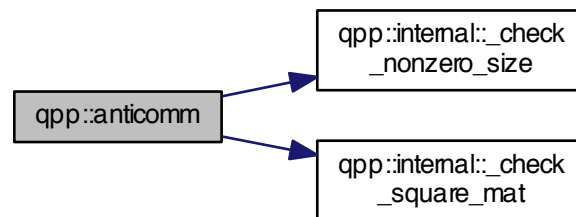
Parameters

A	Eigen expression
B	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, const std::vector< std::size_t > & dims)`

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

Note

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

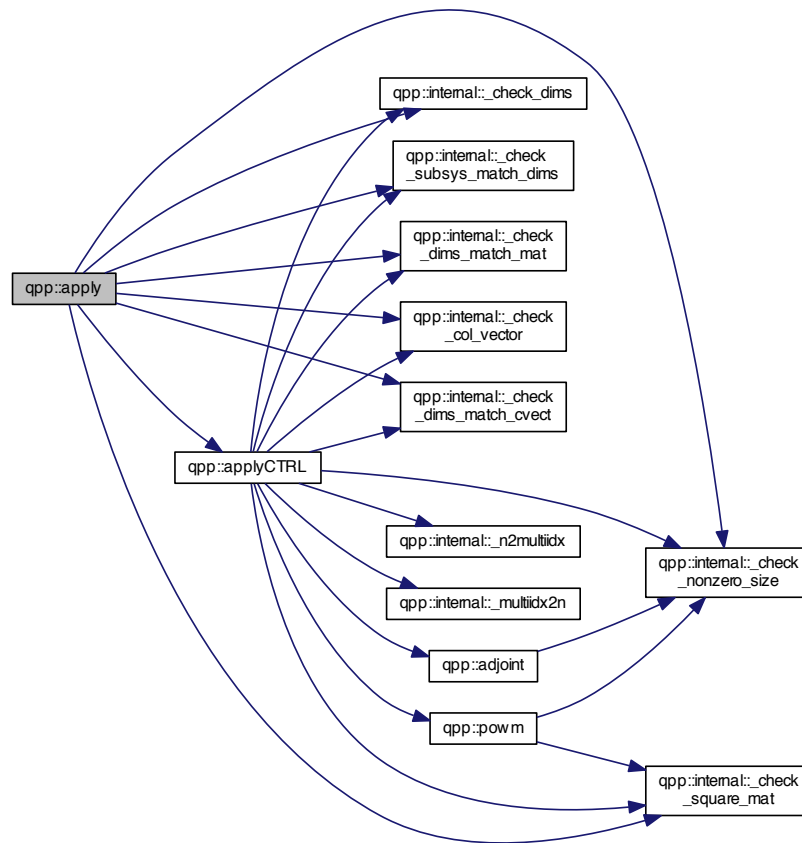
Parameters

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

Returns

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

Here is the call graph for this function:



```

6.1.2.7 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 d = 2 )

```

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

Note

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

Parameters

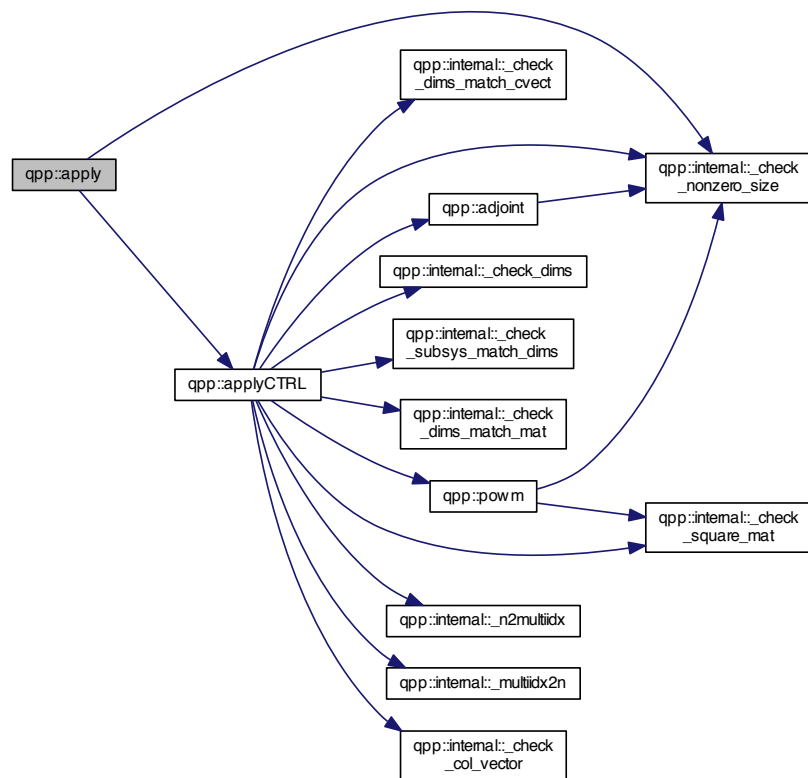
<i>state</i>	Eigen expression
<i>A</i>	Eigen expression
<i>subsys</i>	Subsystem indexes where the gate A is applied

<i>d</i>	Local dimensions of all local Hilbert spaces (must all be equal)
----------	--

Returns

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

Here is the call graph for this function:



6.1.2.8 `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, const std::vector< std::size_t > & dims)`

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

Note

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

Parameters

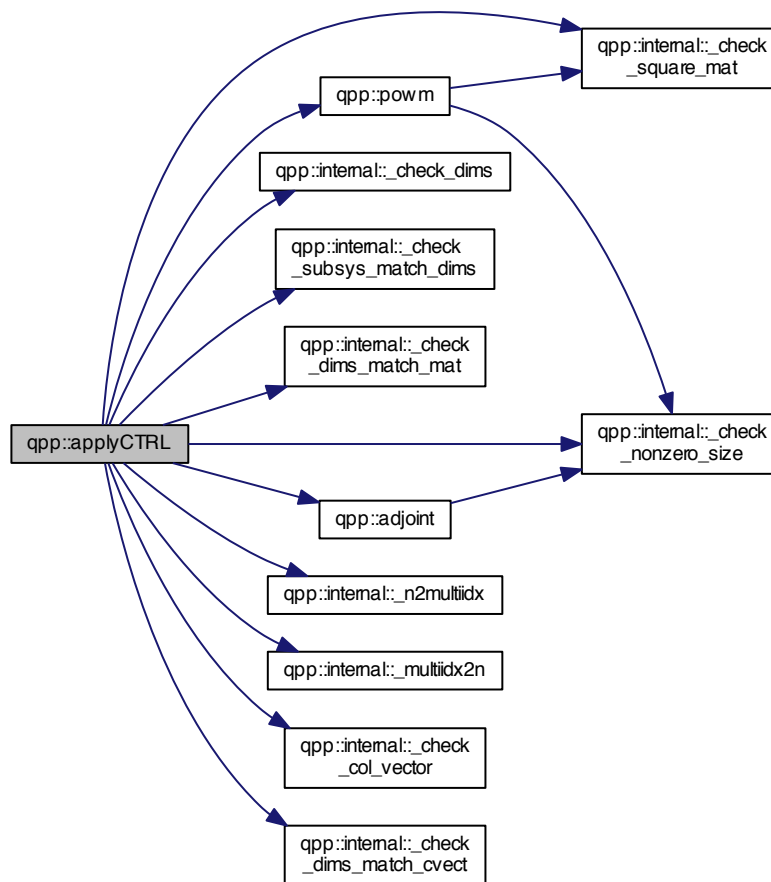
<i>state</i>	Eigen expression
--------------	------------------

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

Returns

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

Here is the call graph for this function:



6.1.2.9 `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 multi-partite state vector or density matrix.

Note

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

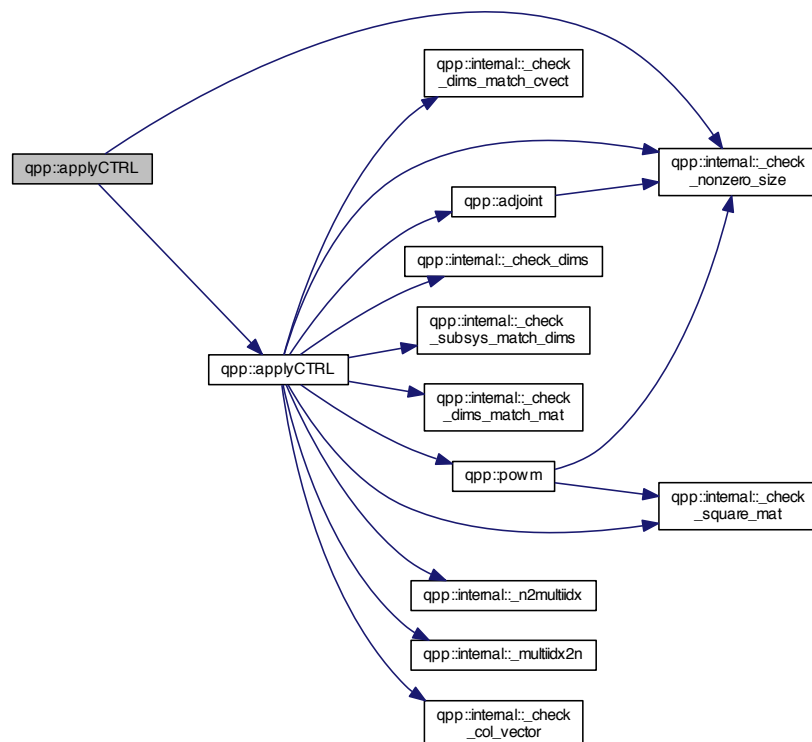
Parameters

<i>state</i>	Eigen expression
<i>A</i>	Eigen expression
<i>ctrl</i>	Control subsystem indexes
<i>subsys</i>	Subsystem indexes where the gate <i>A</i> is applied
<i>d</i>	Local dimensions of all local Hilbert spaces (must all be equal)

Returns

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

Here is the call graph for this function:



6.1.2.10 `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*.

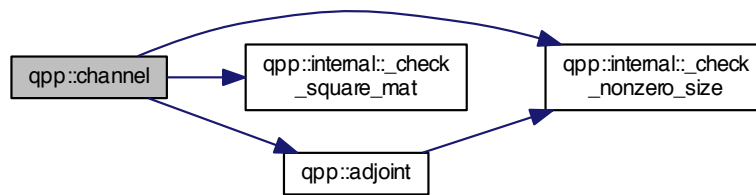
Parameters

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

Returns

Output density matrix after the action of the channel

Here is the call graph for this function:



6.1.2.11 `template<typename Derived> cmat qpp::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*.

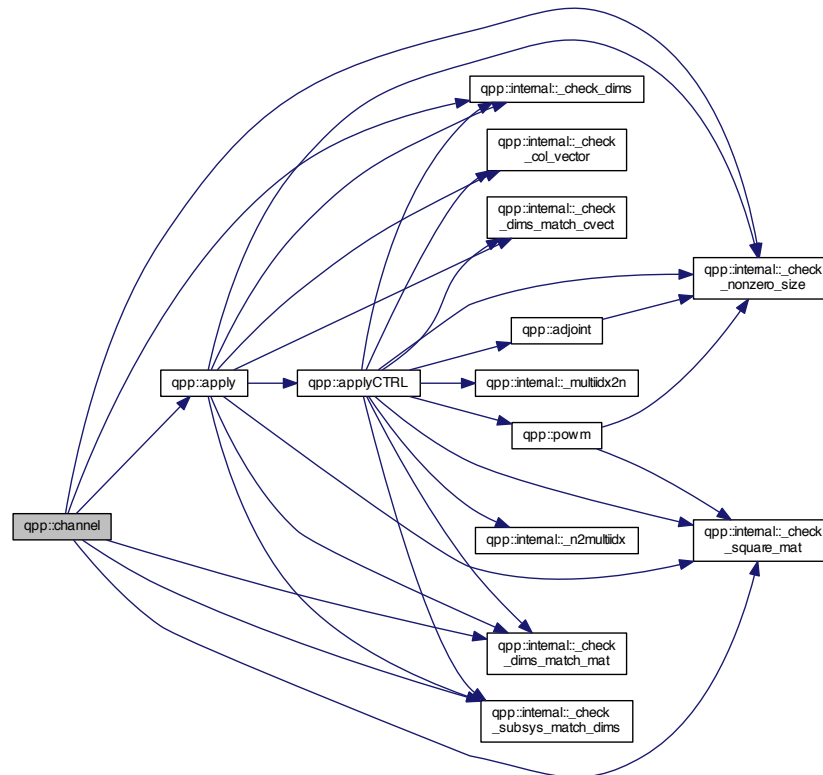
Parameters

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

Returns

Output density matrix after the action of the channel

Here is the call graph for this function:



6.1.2.12 `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 d = 2)`

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

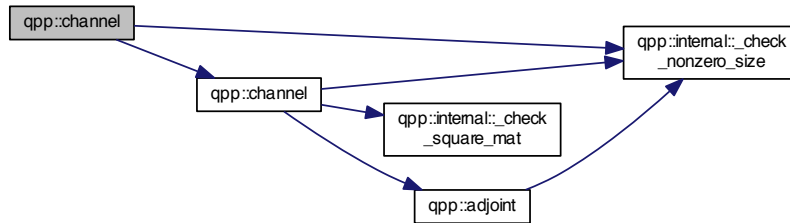
Parameters

<i>rho</i>	Eigen expression
<i>Ks</i>	Set of Kraus operators
<i>subsys</i>	Subsystems' indexes where the Kraus operators <i>Ks</i> are applied
<i>d</i>	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:



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

Note

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

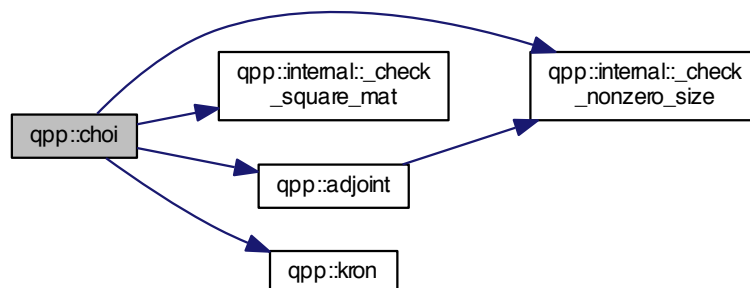
Parameters

K_s	Set of Kraus operators
-------	------------------------

Returns

Choi matrix representation

Here is the call graph for this function:



6.1.2.14 `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 $i \neq j$

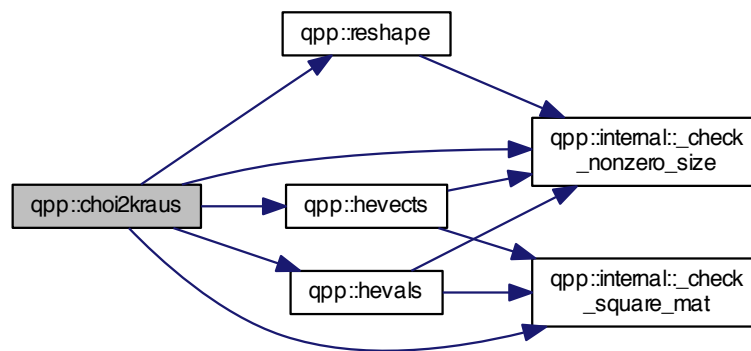
Parameters

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

Returns

Set of Kraus operators

Here is the call graph for this function:

6.1.2.15 `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

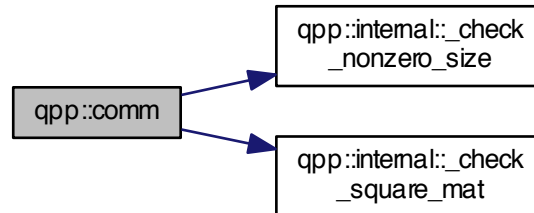
Parameters

A	Eigen expression
B	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:



6.1.2.16 `std::vector<std::size_t> qpp::compperm (const std::vector< std::size_t > & perm, const std::vector< std::size_t > & sigma)`

Compose permutations.

Parameters

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

Returns

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

Here is the call graph for this function:



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

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

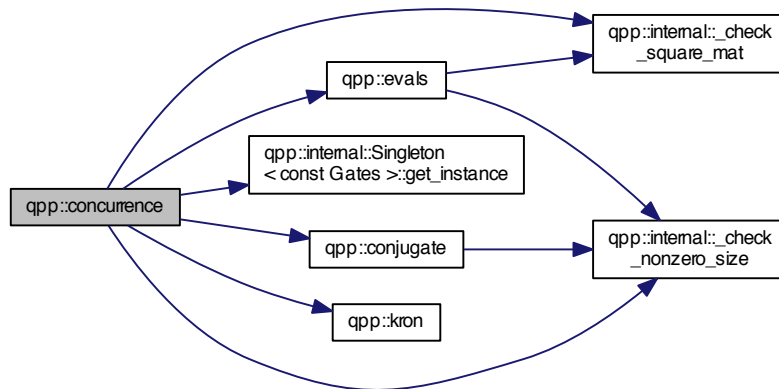
Parameters

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

Returns

Wootters concurrence

Here is the call graph for this function:



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

Complex conjugate.

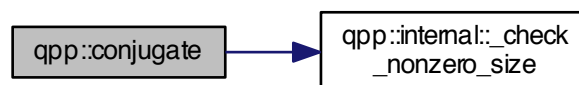
Parameters

A	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:



6.1.2.19 `double qpp::contfrac2x (const std::vector< int > & cf, std::size_t n)`

Real representation of a simple continued fraction.

Parameters

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

Returns

Real representation of the simple continued fraction

6.1.2.20 `double qpp::contfrac2x (const std::vector< int > & cf)`

Real representation of a simple continued fraction.

Parameters

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

Returns

Real representation of the simple continued fraction

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

Matrix cos.

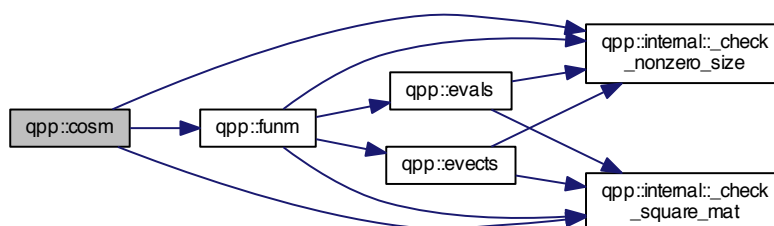
Parameters

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

Returns

Matrix cosine of *A*

Here is the call graph for this function:

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

Functor.

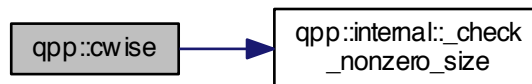
Parameters

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

Returns

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

Here is the call graph for this function:



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

Determinant.

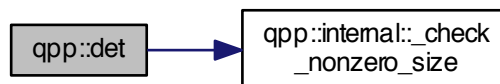
Parameters

A	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:



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

Eigen expression ostream manipulator.

Parameters

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

Returns

Instance of `qpp::internal::internal::LOManipEigen`

6.1.2.25 `internal::LOManipEigen qpp::disp (cplx z, double chop = qpp::chop)`

Complex number ostream manipulator.

Parameters

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

Returns

Instance of `qpp::internal::internal::LOManipEigen`

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

Range ostream manipulator.

Parameters

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

Returns

Instance of `qpp::internal::internal::LOManipRange`

6.1.2.27 `template<typename Container > internal::LOManipRange<typename Container::const_iterator> qpp::disp (const Container & c, const std::string & separator, const std::string & start = " [", const std::string & end = "] ")`

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

Parameters

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

Returns

Instance of `qpp::internal::internal::LOManipRange`

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

C-style pointer ostream manipulator.

Parameters

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

Returns

Instance of `qpp::internal::internal::IOManipPointer`

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\(\)](#)

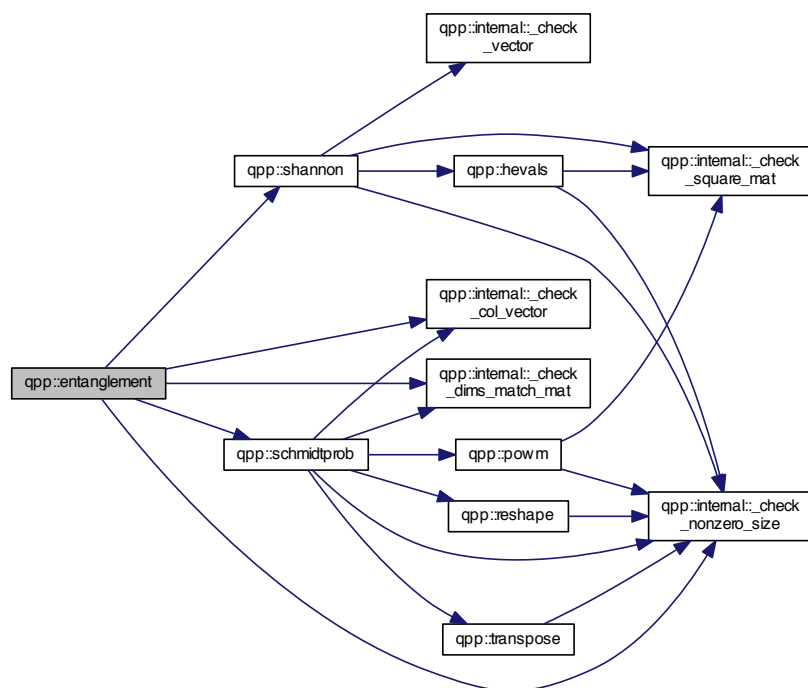
Parameters

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

Returns

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.

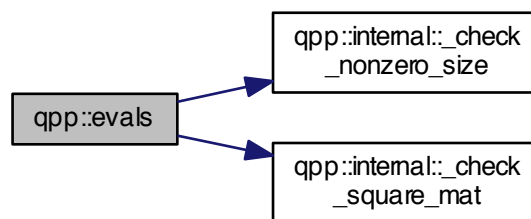
Parameters

<code>A</code>	Eigen expression
----------------	------------------

Returns

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::evecs (const Eigen::MatrixBase< Derived > & A)`

Eigenvectors.

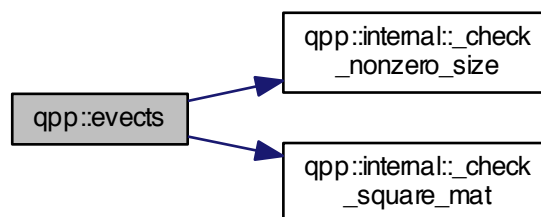
Parameters

<code>A</code>	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.

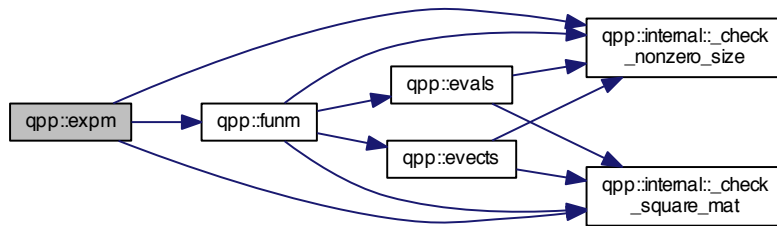
Parameters

A	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 $f(A)$

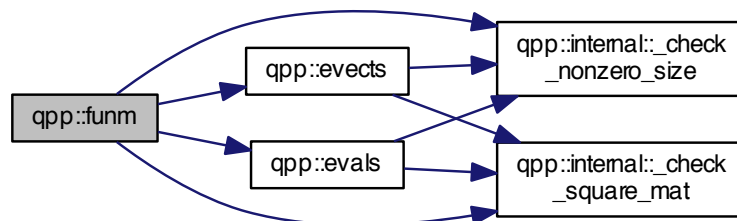
Parameters

A	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 `std::size_t qpp::gcd (std::size_t m, std::size_t n)`

Greatest common divisor of two non-negative integers.

Parameters

m	Non-negative integer
n	Non-negative integer

Returns

Greatest common divisor of m and n

6.1.2.35 `std::size_t qpp::gcd (const std::vector< std::size_t > & ns)`

Greatest common divisor of a list of non-negative integers.

Parameters

ns	List of non-negative integers
------	-------------------------------

Returns

Greatest common divisor of all numbers in ns

Here is the call graph for this function:

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

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

Note

Both local dimensions must be equal

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

See also

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

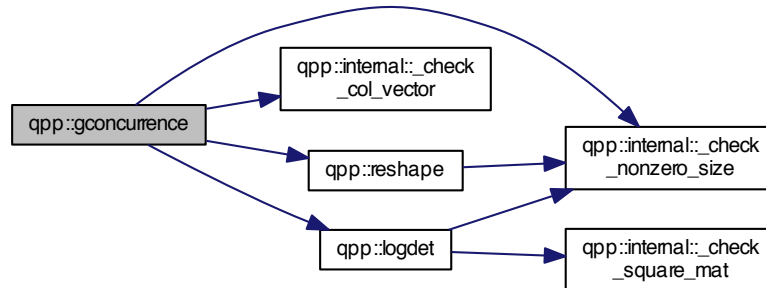
Parameters

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

Returns

G-concurrence

Here is the call graph for this function:



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

Gram-Schmidt orthogonalization (std::vector overload)

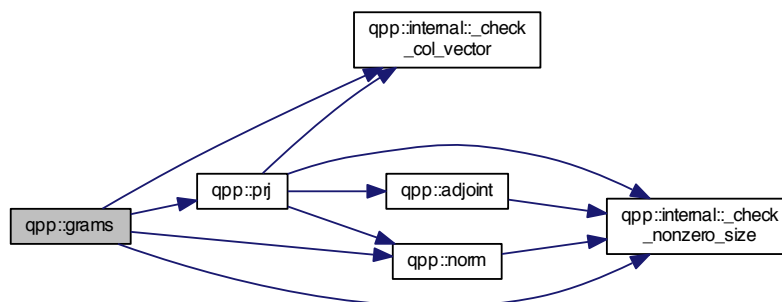
Parameters

<code>Vs</code>	std::vector 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:



6.1.2.38 `template<typename Derived> DynMat<typename Derived::Scalar> qpp::grams (const std::initializer_list< Derived> & Vs)`

Gram-Schmidt orthogonalization (std::initializer_list overload)

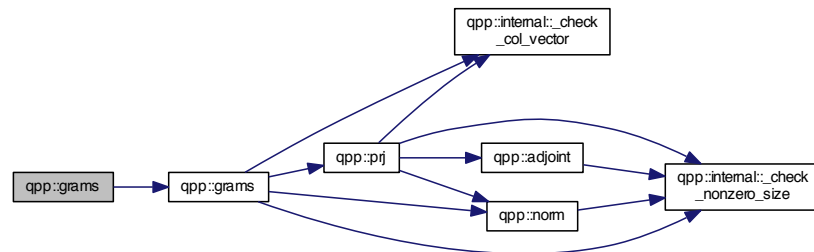
Parameters

<i>Vs</i>	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:



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

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

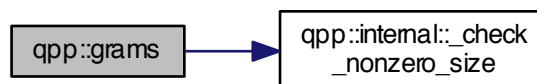
Parameters

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

Returns

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

Here is the call graph for this function:



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

Hermitian eigenvalues.

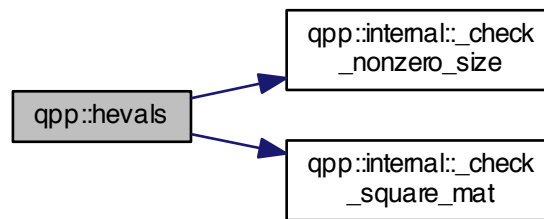
Parameters

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

Returns

Eigenvalues of Hermitian A , as a real dynamic column vector

Here is the call graph for this function:



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

Hermitian eigenvectors.

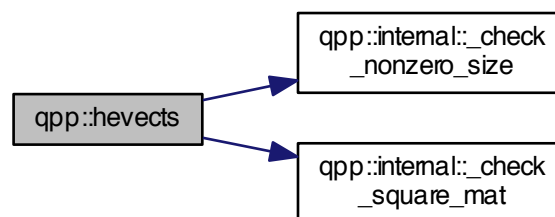
Parameters

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

Returns

Eigenvectors of Hermitian A , as columns of a complex matrix

Here is the call graph for this function:



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

Inverse.

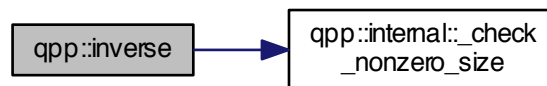
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.43 `std::vector<std::size_t> qpp::invperm (const std::vector< std::size_t > & perm)`

Inverse permutation.

Parameters

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

Returns

Inverse of the permutation *perm*

Here is the call graph for this function:



6.1.2.44 `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

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

Returns

Its argument *head*

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

Kronecker product (variadic overload)

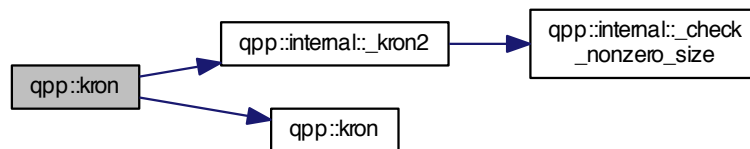
Parameters

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

Returns

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

Here is the call graph for this function:



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

Kronecker product (std::vector overload)

Parameters

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

Returns

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

Here is the call graph for this function:



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

Kronecker product (std::initializer_list overload)

Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.48 `template<typename Derived> DynMat<typename Derived::Scalar> qpp::kronpow (const Eigen::MatrixBase< Derived> & A, std::size_t n)`

Kronecker power.

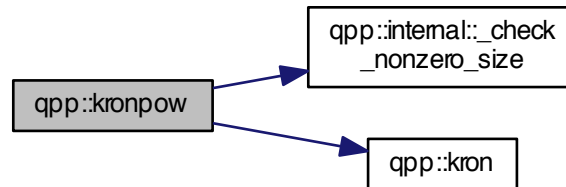
Parameters

A	Eigen expression
n	Non-negative integer

Returns

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

Here is the call graph for this function:



6.1.2.49 `std::size_t qpp::lcm (std::size_t m, std::size_t n)`

Least common multiple of two positive integers.

Parameters

m	Positive integer
n	Positive integer

Returns

Least common multiple of m and n

Here is the call graph for this function:



6.1.2.50 `std::size_t qpp::lcm (const std::vector< std::size_t > & ns)`

Least common multiple of a list of positive integers.

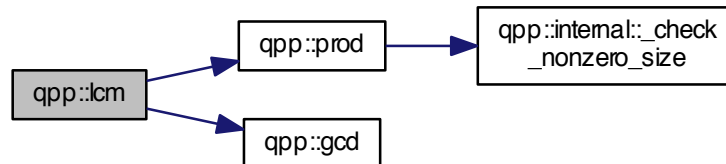
Parameters

ns	List of positive integers
------	---------------------------

Returns

Least common multiple of all numbers in *ns*

Here is the call graph for this function:



6.1.2.51 `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

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

Parameters

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

6.1.2.52 `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)

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

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

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

Example:

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

Note

If *var_name* is a complex matrix, only the real part is loaded

Parameters

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

Returns

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

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

Loads an Eigen dynamic matrix from a MATLAB .mat file, specialization for complex matrices ([qpp::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");
```

Parameters

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

Returns

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

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

Logarithm of the determinant.

Especially useful when the determinant overflows/underflows

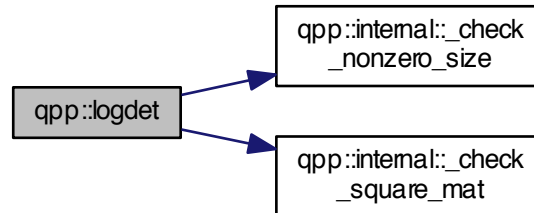
Parameters

<i>A</i>	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:



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

Matrix logarithm.

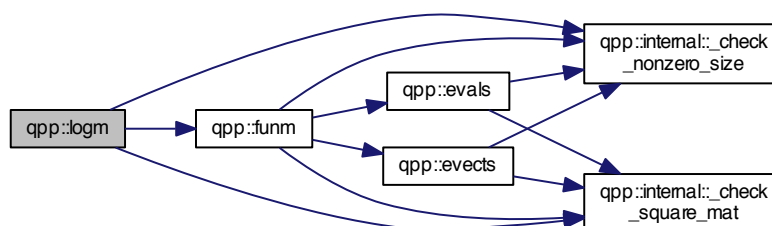
Parameters

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

Returns

Matrix logarithm of A

Here is the call graph for this function:



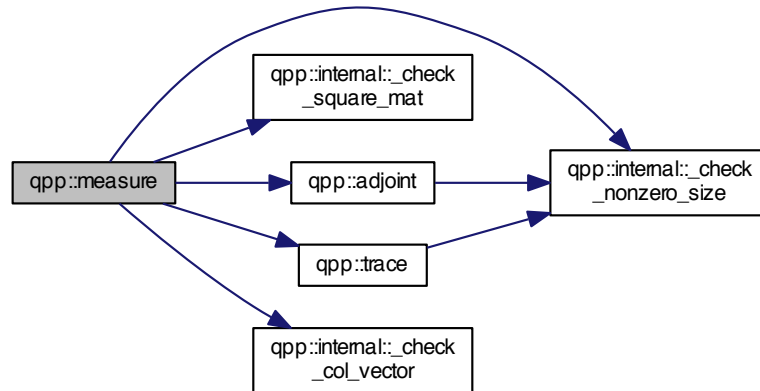
6.1.2.57 `template<typename Derived> double qpp::lognegativity (const Eigen::MatrixBase< Derived> & A, const std::vector< std::size_t> & dims)`

Logarithmic negativity of the bi-partite mixed state A .

Returns

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

Here is the call graph for this function:



6.1.2.59 `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)

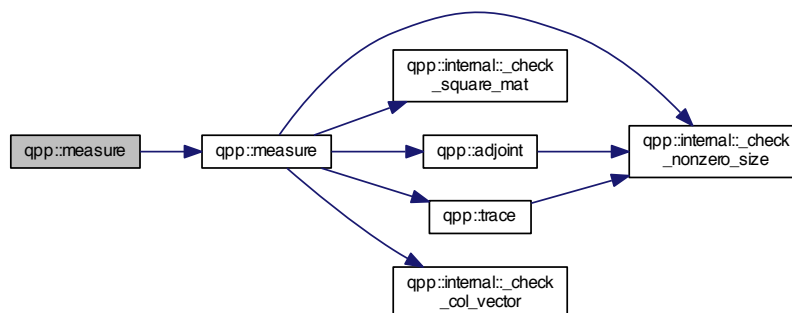
Parameters

A	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:



6.1.2.60 `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 .

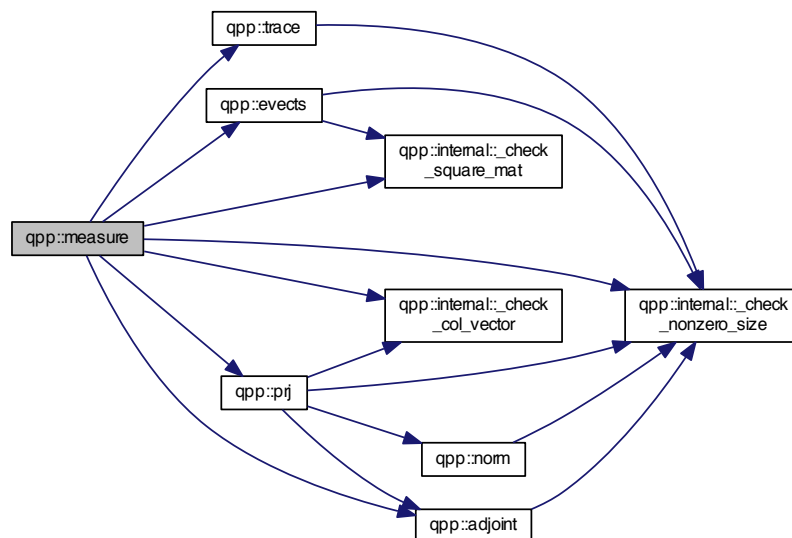
Parameters

A	Eigen expression
M	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:



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

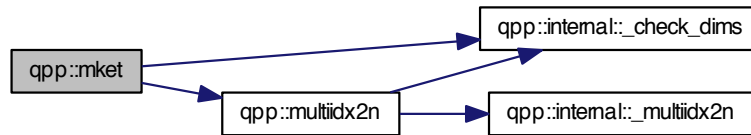
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.62 `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

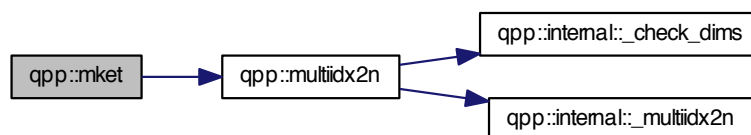
Parameters

<i>mask</i>	<code>std::vector</code> of non-negative integers
<i>d</i>	Subsystems' dimension

Returns

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

Here is the call graph for this function:



6.1.2.63 `cmat qpp::mproj (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 $dims$

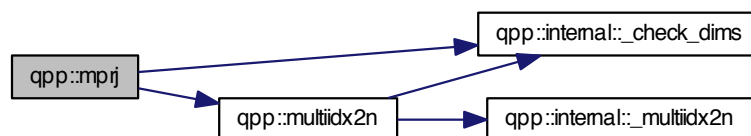
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.64 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

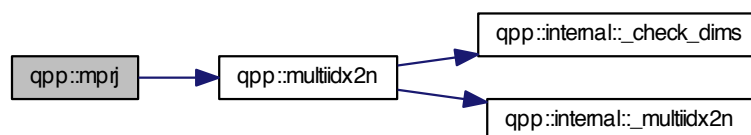
Parameters

<i>mask</i>	std::vector of non-negative integers
<i>d</i>	Subsystems' dimension

Returns

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

Here is the call graph for this function:



6.1.2.65 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.

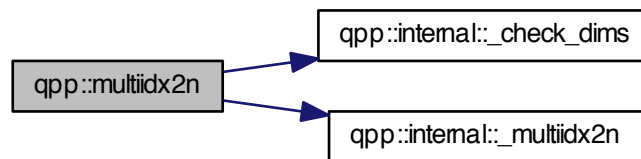
Parameters

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

Returns

Non-negative integer index

Here is the call graph for this function:



6.1.2.66 `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.

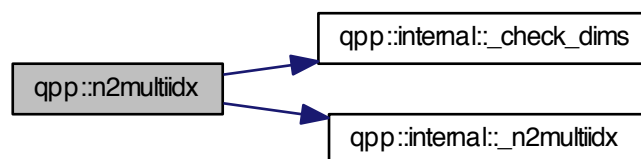
Parameters

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

Returns

Multi-index of the same size as *dims*

Here is the call graph for this function:



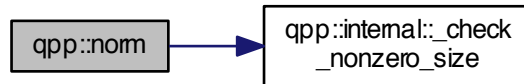
6.1.2.67 `template<typename Derived > double qpp::negativity (const Eigen::MatrixBase< Derived > & A, const
std::vector< std::size_t > & dims)`

Negativity of the bi-partite mixed state A .

Returns

Frobenius norm of A , as a real number

Here is the call graph for this function:



6.1.2.69 `std::complex<double> qpp::omega (std::size_t D)`

D-th root of unity.

Parameters

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

Returns

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

6.1.2.70 `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>
```

6.1.2.71 `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>
```

6.1.2.72 `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 $A^0 = I$

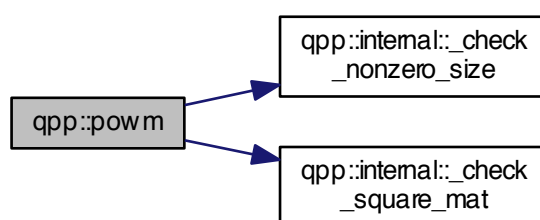
Parameters

A	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:



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

Projector.

Normalized projector onto state vector

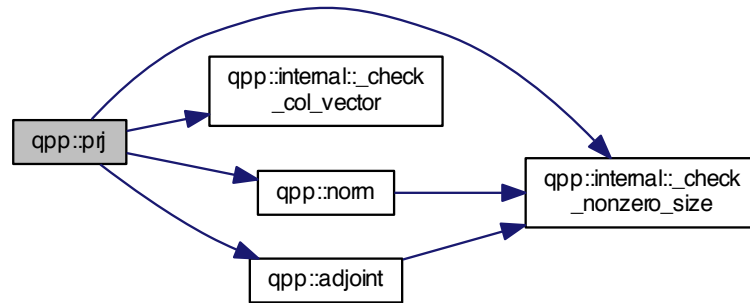
Parameters

V	Eigen expression
-----	------------------

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:



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

Element-wise product of A .

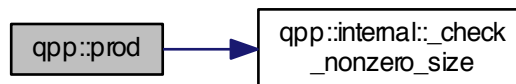
Parameters

A	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:



6.1.2.75 `template<typename InputIterator> auto qpp::prod (InputIterator first, InputIterator last) -> typename InputIterator::value_type`

Element-wise product of a range.

Parameters

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

Returns

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

6.1.2.76 `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

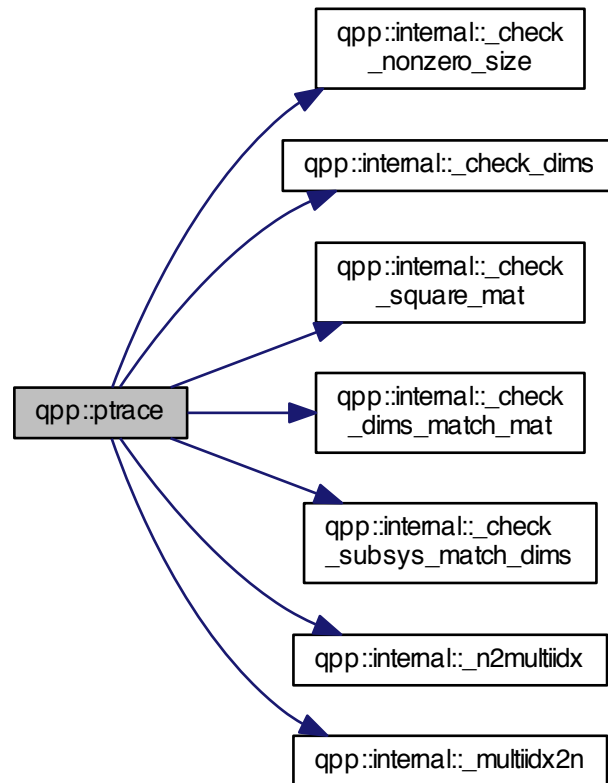
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.77 `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

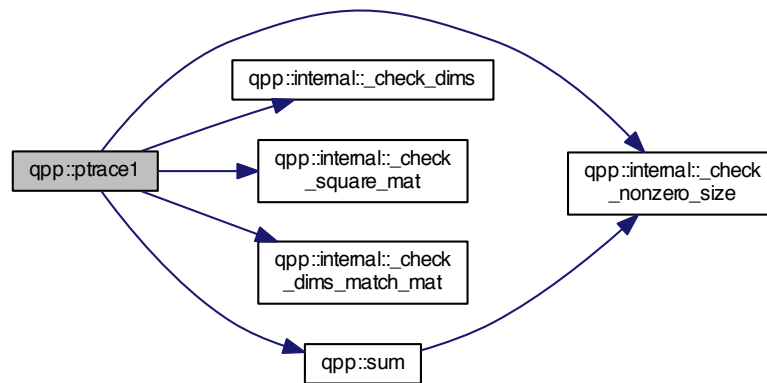
Parameters

<i>A</i>	Eigen expression
<i>dims</i>	Dimensions of the bi-partite system (must be a <code>std::vector</code> with 2 elements)

Returns

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

Here is the call graph for this function:



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

Partial trace.

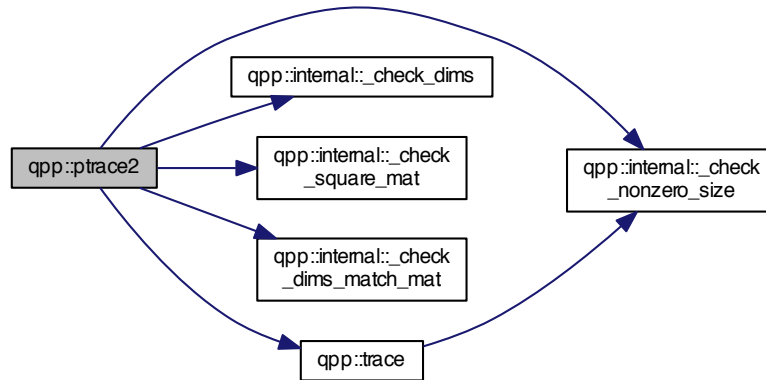
Parameters

A	Eigen expression
$dims$	Dimensions of the bi-partite system (must be a <code>std::vector</code> 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:



6.1.2.79 `template<typename Derived> DynMat<typename Derived::Scalar> qpp::ptranspose (const Eigen::MatrixBase<Derived> & A, const std::vector< std::size_t > & subsys, const std::vector< std::size_t > & dims)`

Partial transpose.

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

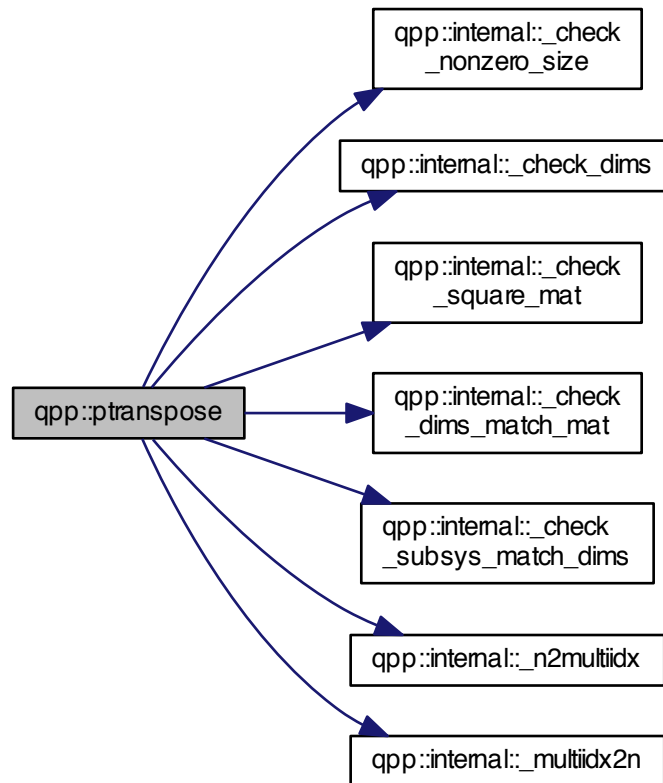
Parameters

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

Returns

Partial transpose $(\cdot)^{T_{subsys}}$ over the subsystems *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:



6.1.2.80 `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.

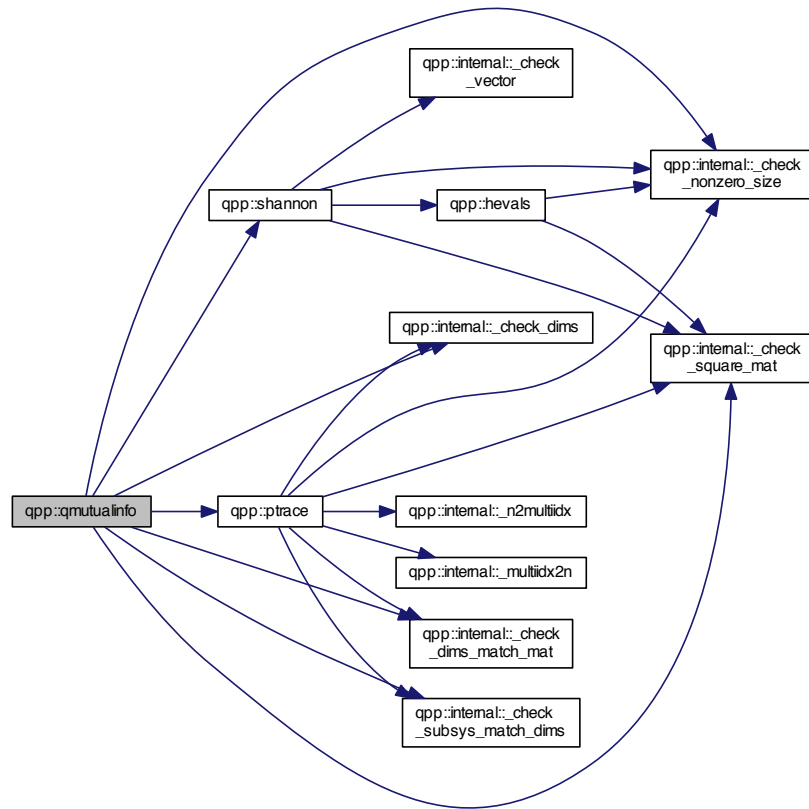
Parameters

<i>A</i>	Eigen expression
<i>subsysA</i>	Indexes of the first subsystem
<i>subsysB</i>	Indexes of the second subsystem
<i>dims</i>	Dimensions of the multi-partite system

Returns

Mutual information between the 2 subsystems

Here is the call graph for this function:



6.1.2.81 `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`

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

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

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

Example:

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


Parameters

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

Returns

Random real matrix

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

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

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

Example:

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

Parameters

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

Returns

Random complex matrix

Here is the call graph for this function:



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

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

Parameters

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

Returns

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

Here is the call graph for this function:

**6.1.2.85** `cmat qpp::randH (std::size_t D)`

Generates a random Hermitian matrix.

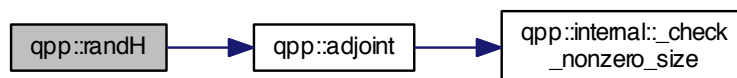
Parameters

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

Returns

Random Hermitian matrix

Here is the call graph for this function:

**6.1.2.86** `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].

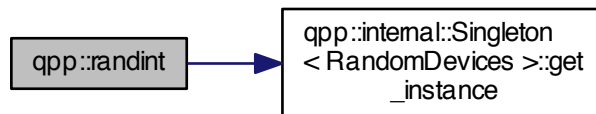
Parameters

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

Returns

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

Here is the call graph for this function:

6.1.2.87 `ket qpp::randket (std::size_t D)`

Generates a random normalized ket (pure state vector)

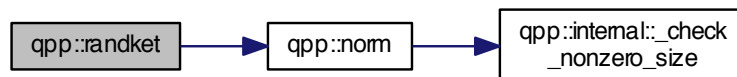
Parameters

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

Returns

Random normalized ket

Here is the call graph for this function:

6.1.2.88 `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 $\sum_i K_i^\dagger K_i = I$

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:



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

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

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

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

Generates a random real matrix with entries normally distributed in $N(\text{mean}, \text{sigma})$, specialization for double matrices ([`qpp::dmat`](#))

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

Example:

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

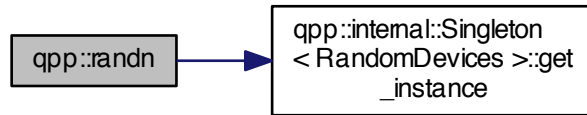
Parameters

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

Returns

Random real matrix

Here is the call graph for this function:



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

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

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

Example:

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

Parameters

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

Returns

Random complex matrix

Here is the call graph for this function:



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

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

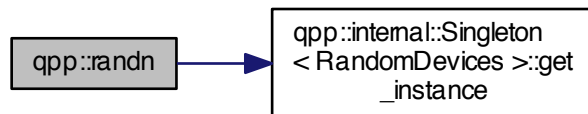
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.93 `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

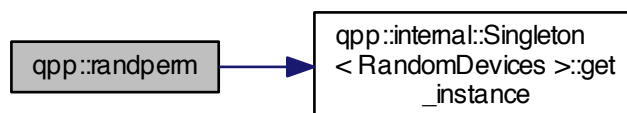
Parameters

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

Returns

Random permutation of size *n*

Here is the call graph for this function:



6.1.2.94 `cmat qpp::randrho (std::size_t D)`

Generates a random density matrix.

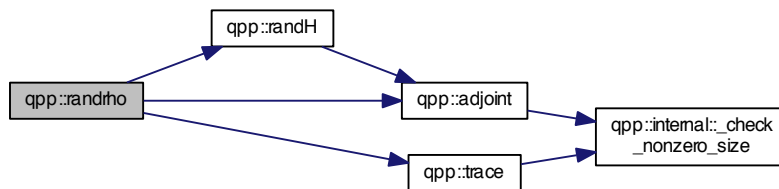
Parameters

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

Returns

Random density matrix

Here is the call graph for this function:

6.1.2.95 `cmat qpp::randU (std::size_t D)`

Generates a random unitary matrix.

Parameters

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

Returns

Random unitary

6.1.2.96 `cmat qpp::randV (std::size_t Din, std::size_t Dout)`

Generates a random isometry matrix.

Parameters

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

Returns

Random isometry matrix

Here is the call graph for this function:



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

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

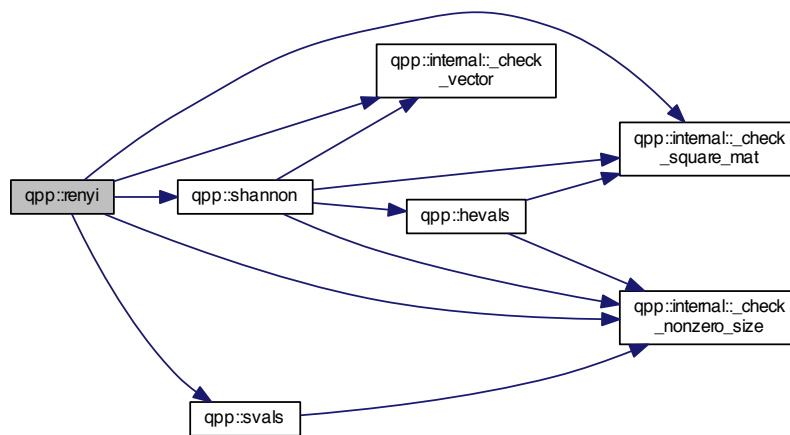
Parameters

A	Eigen expression, representing a probability distribution (real dynamic column vector) or a density matrix (complex dynamic matrix)
$alpha$	Non-negative real number, use qpp::infty for $\alpha = \infty$

Returns

Renyi- α entropy, with the logarithm in base 2

Here is the call graph for this function:



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

Reshape.

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

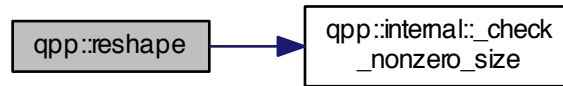
Parameters

A	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:



6.1.2.99 `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

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

6.1.2.100 `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.

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)

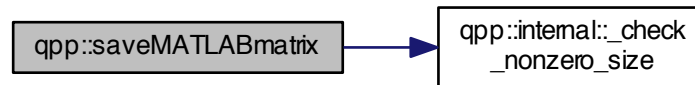
6.1.2.101 `template<> void qpp::saveMATLABmatrix (const Eigen::MatrixBase< dmat> & A, const std::string & mat_file, const std::string & var_name, const std::string & mode) [inline]`

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

Parameters

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

Here is the call graph for this function:



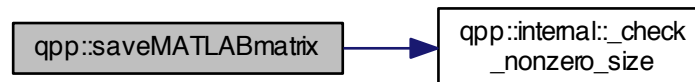
6.1.2.102 `template<> void qpp::saveMATLABmatrix (const Eigen::MatrixBase< cmat > & A, const std::string & mat_file, const std::string & var_name, const std::string & mode) [inline]`

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

Parameters

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

Here is the call graph for this function:



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

Schatten norm.

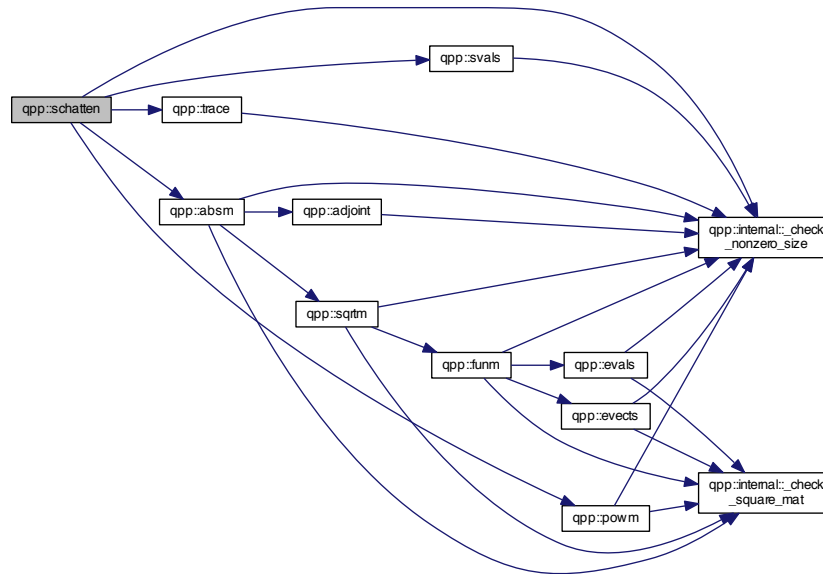
Parameters

<i>A</i>	Eigen expression
<i>p</i>	Integer, greater or equal to 1

Returns

Schatten- p norm of A , as a real number

Here is the call graph for this function:



6.1.2.104 `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\(\)](#)

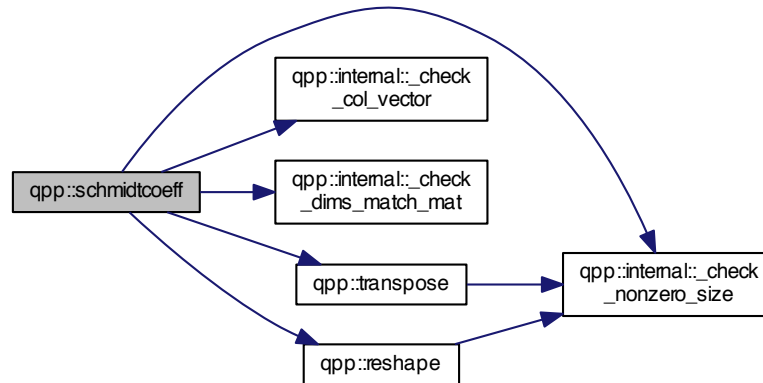
Parameters

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

Returns

Schmidt coefficients of A , as a complex dynamic column vector

Here is the call graph for this function:



6.1.2.105 `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\(\)](#)

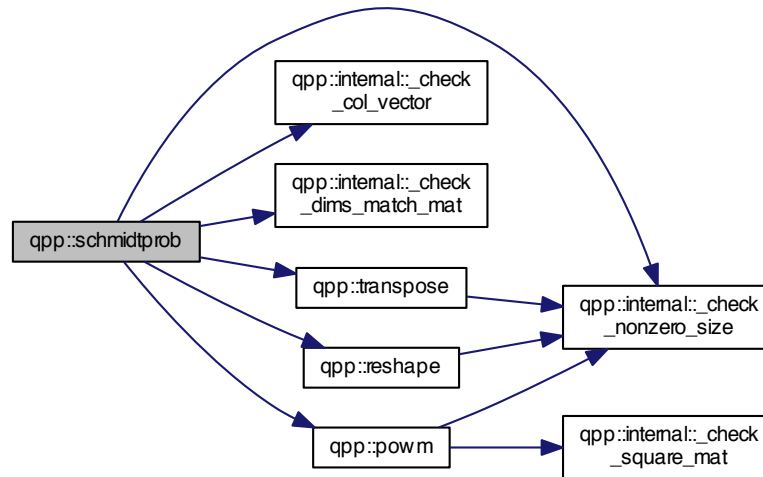
Parameters

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

Returns

Schmidt probabilities of A , as a real dynamic column vector

Here is the call graph for this function:



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

Schmidt basis on Alice's side.

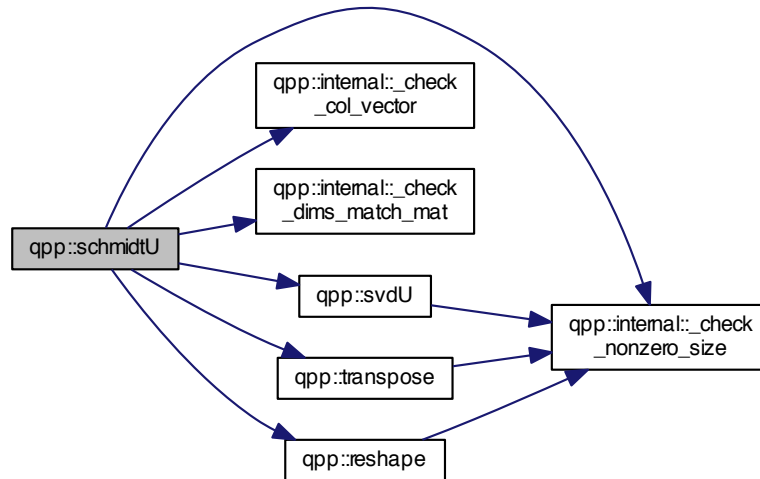
Parameters

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

Returns

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

Here is the call graph for this function:



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

Schmidt basis on Bob's side.

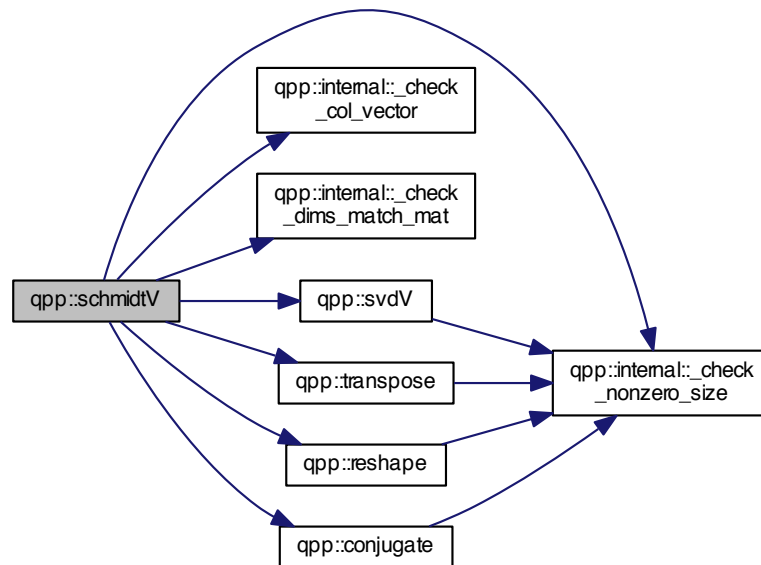
Parameters

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

Returns

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

Here is the call graph for this function:



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

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

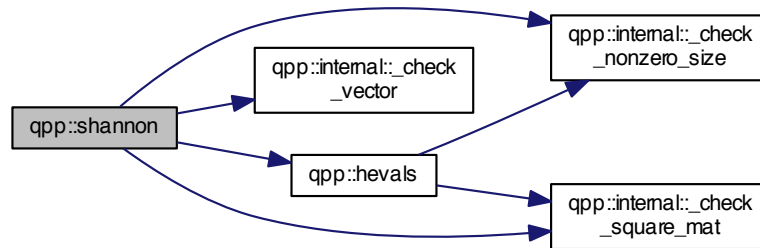
Parameters

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

Returns

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

Here is the call graph for this function:



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

Matrix sin.

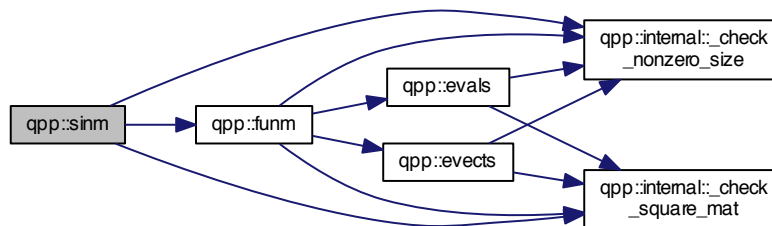
Parameters

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

Returns

Matrix sine of A

Here is the call graph for this function:



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

Matrix power.

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

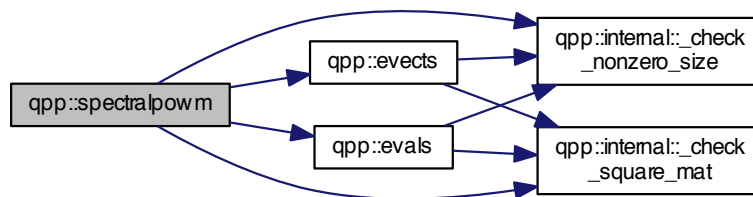
Parameters

A	Eigen expression
z	Complex number

Returns

Matrix power A^z

Here is the call graph for this function:



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

Matrix square root.

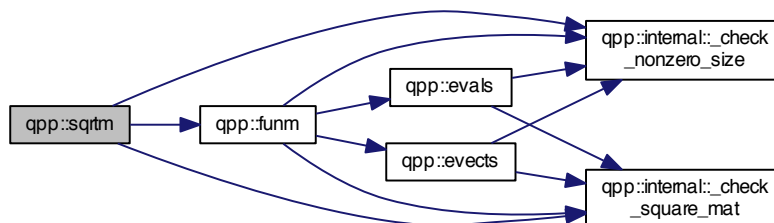
Parameters

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

Returns

Matrix square root of A

Here is the call graph for this function:



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

Element-wise sum of A .

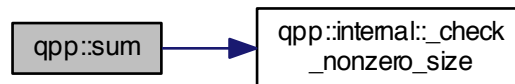
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.113 `template<typename InputIterator > auto qpp::sum (InputIterator first, InputIterator last) -> typename InputIterator::value_type`

Element-wise sum of a range.

Parameters

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

Returns

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

6.1.2.114 `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.

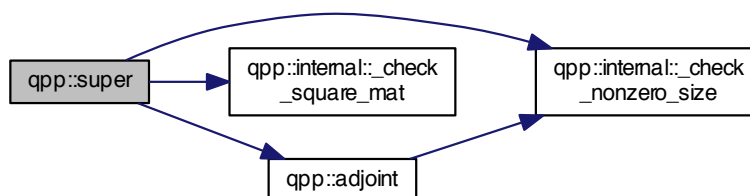
Parameters

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

Returns

Superoperator matrix representation

Here is the call graph for this function:



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

Singular values.

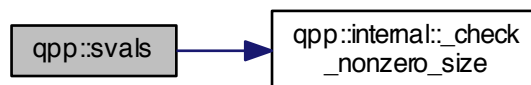
Parameters

<code>A</code>	Eigen expression
----------------	------------------

Returns

Singular values of A , as a real dynamic column vector

Here is the call graph for this function:



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

Left singular vectors.

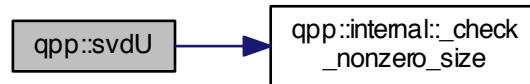
Parameters

<code>A</code>	Eigen expression
----------------	------------------

Returns

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

Here is the call graph for this function:



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

Right singular vectors.

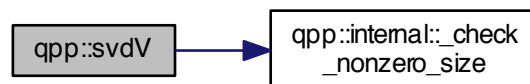
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.118 `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

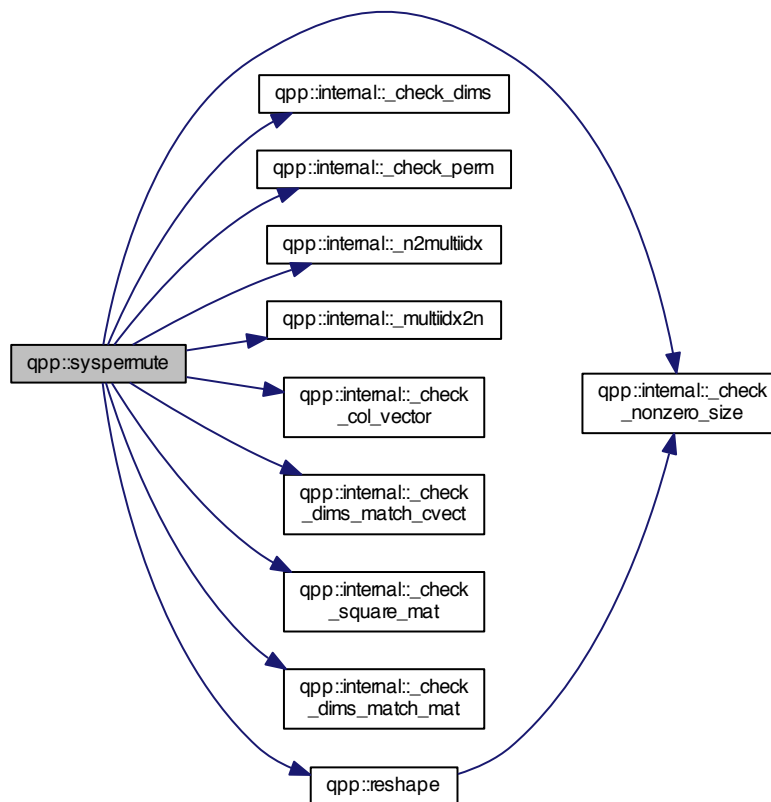
Parameters

<i>A</i>	Eigen expression
<i>perm</i>	Permutation
<i>dims</i>	Dimensions of the multi-partite system

Returns

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

Here is the call graph for this function:



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

Trace.

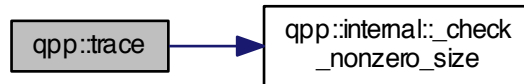
Parameters

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

Returns

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

Here is the call graph for this function:



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

Transpose.

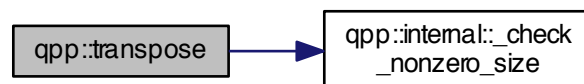
Parameters

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

Returns

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

Here is the call graph for this function:



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

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

.

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

Parameters

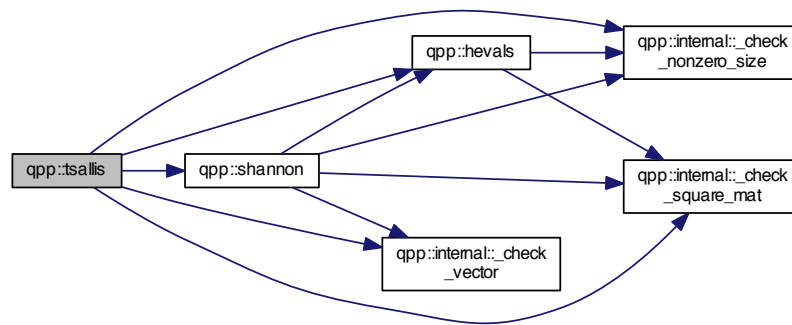
A	Eigen expression, representing a probability distribution (real dynamic column vector) or a density matrix (complex dynamic matrix)
-----	---

<i>alpha</i>	Non-negative real number
--------------	--------------------------

Returns

Renyi- α entropy, with the logarithm in base 2

Here is the call graph for this function:



6.1.2.122 `std::vector<int> qpp::x2contfrac (double x, std::size_t n, std::size_t cut = 1e5)`

Simple continued fraction expansion.

Parameters

<i>x</i>	Real number
<i>n</i>	Number of terms in the expansion
<i>cut</i>	Stop the expansion when the next term is greater than <i>cut</i>

Returns

Integer vector containing the simple continued fraction expansion of *x*. If there are *m* less than *n* terms in the expansion, a shorter vector with *m* components is returned.

6.1.3 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
```

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::infy = -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)

6.1.3.9 constexpr double qpp::pi = 3.141592653589793238462643383279502884

π

6.1.3.10 RandomDevices& qpp::rdevs = RandomDevices::get_instance()

[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

- class [Qudit](#)

Functions

- `template<typename Derived1 , typename Derived2 >`
`DynMat< typename Derived1::Scalar > apply_old` (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 multi-partite state vector or density matrix.
- `template<typename Derived >`
`cmat channel_old` (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 multi-partite 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.
- `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.
- `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 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, const std::vector< std::size_t > &dims)`
Applies the controlled-gate A to the part subsys of a multi-partite state vector or density matrix.
- `template<typename Derived1 , typename Derived2 >`
`DynMat< typename Derived1::Scalar > 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 multi-partite 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.

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 multi-partite state vector or density matrix.

Note

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

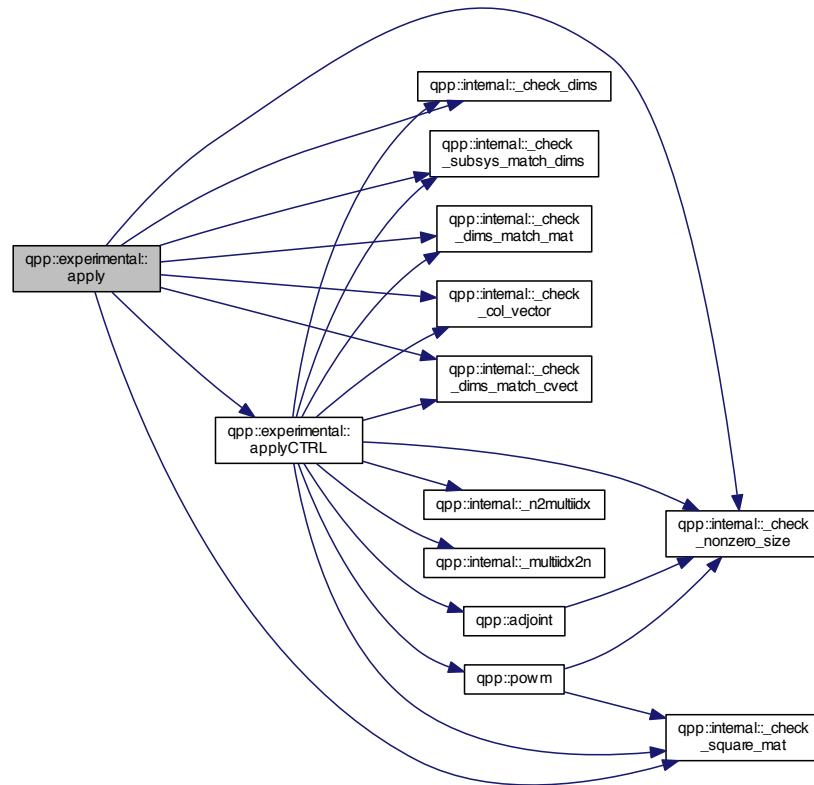
Parameters

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

Returns

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

Here is the call graph for this function:



```

6.2.2.2 template<typename Derived1 , typename Derived2 > DynMat<typename Derived1::Scalar>
qpp::experimental::apply_old ( 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 multi-partite state vector or density matrix.

Note

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

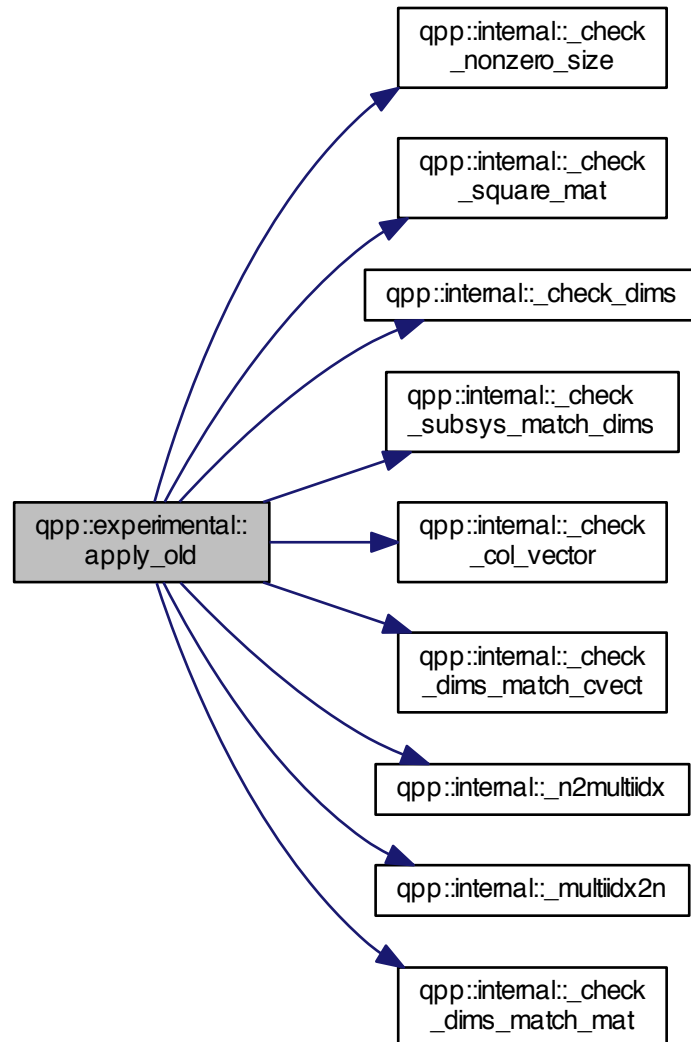
Parameters

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

Returns

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

Here is the call graph for this function:



```

6.2.2.3 template<typename Derived1 , typename Derived2 > DynMat<typename Derived1::Scalar>
qpp::experimental::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, const std::vector<
std::size_t > & dims )

```

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

Note

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

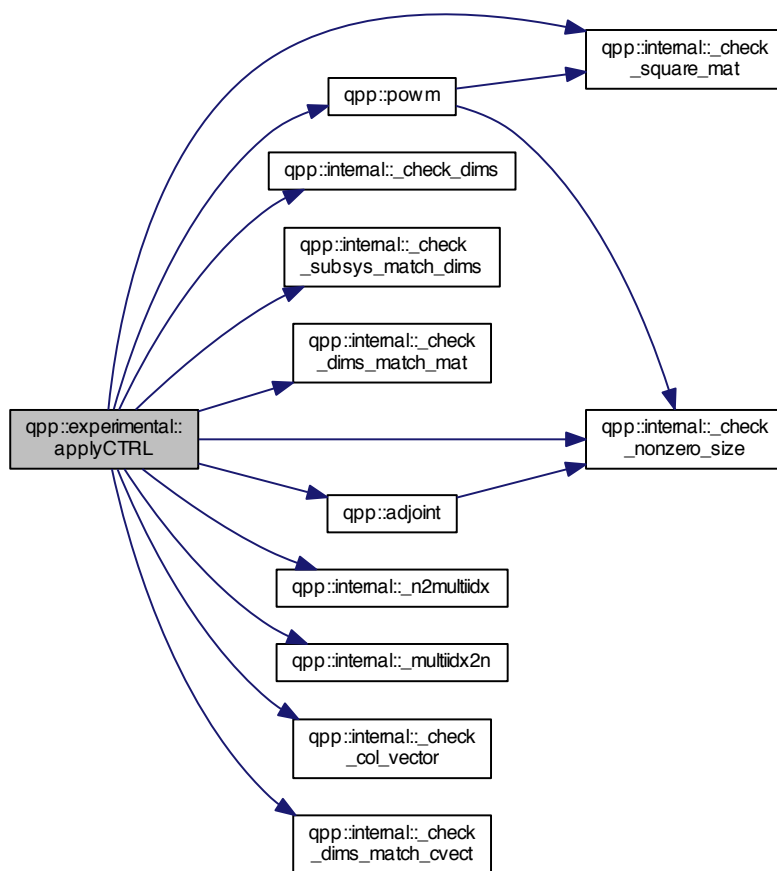
Parameters

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

Returns

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

Here is the call graph for this function:



6.2.2.4 `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*.

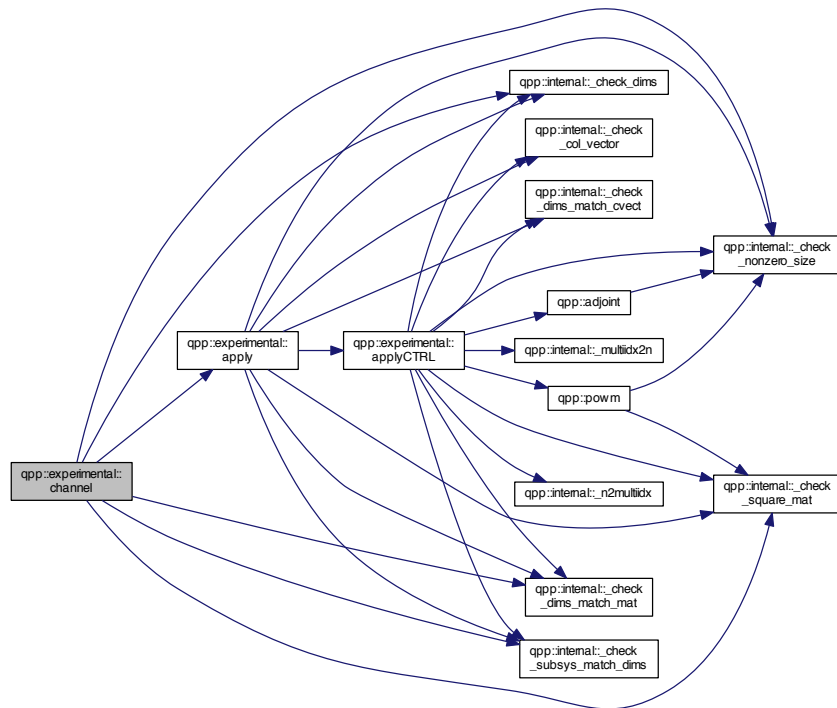
Parameters

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

Returns

Output density matrix after the action of the channel

Here is the call graph for this function:



```
6.2.2.5 template<typename Derived> cmat qpp::experimental::channel_old ( 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*.

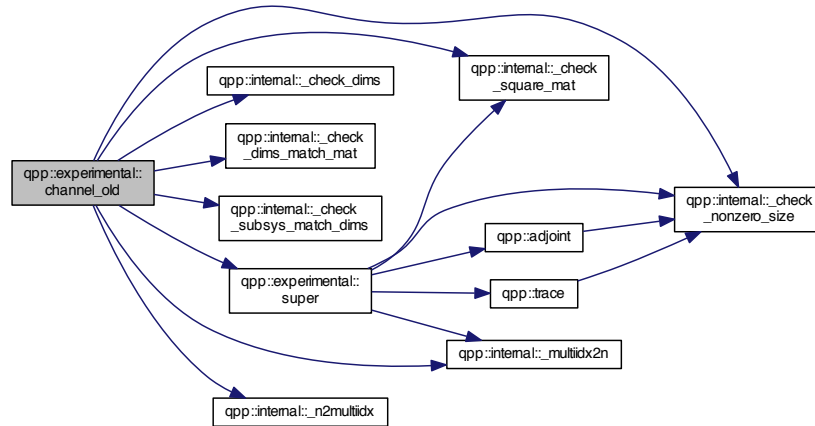
Parameters

<i>rho</i>	Eigen expression
<i>Ks</i>	Set of Kraus operators
<i>subsys</i>	Subsystems' indexes
<i>dims</i>	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:



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

Note

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

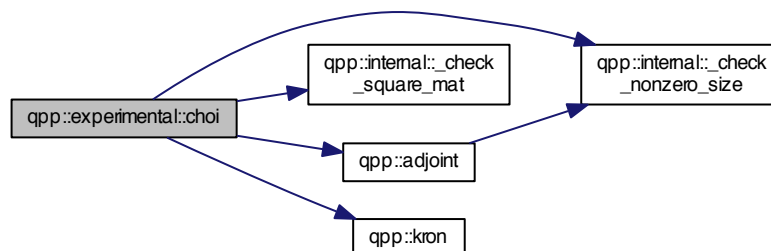
Parameters

K_s	Set of Kraus operators
-------	------------------------

Returns

Choi matrix representation

Here is the call graph for this function:



6.2.2.7 `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 multi-partite multiple-controlled- A gate in matrix form.

Note

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

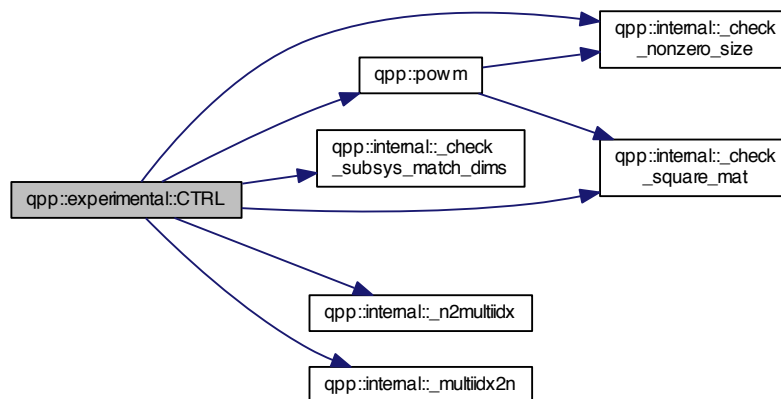
Parameters

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



6.2.2.8 `template<typename InputIterator > void qpp::experimental::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::experimental::displn\(\)](#)

Parameters

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

Returns

Output stream

6.2.2.9 `template<typename T> std::ostream& qpp::experimental::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::experimental::displn\(\)](#)

Parameters

<i>x</i>	Container
<i>separator</i>	Separator
<i>start</i>	Left marking
<i>end</i>	Right marking
<i>os</i>	Output stream

Returns

Output stream

Here is the call graph for this function:



6.2.2.10 `template<typename T> std::ostream& qpp::experimental::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::experimental::displn\(\)](#)

Parameters

<i>x</i>	Pointer to the first element
<i>n</i>	Number of elements to be displayed
<i>separator</i>	Separator
<i>start</i>	Left marking
<i>end</i>	Right marking
<i>os</i>	Output stream

Returns

Output stream

6.2.2.11 `template<typename Derived> std::ostream& qpp::experimental::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::experimental::displn\(\)](#)

Parameters

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

Returns

Output stream

6.2.2.12 `std::ostream& qpp::experimental::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::experimental::displn\(\)](#)

Parameters

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

Returns

Output stream

Here is the call graph for this function:



6.2.2.13 `template<typename InputIterator > std::ostream& qpp::experimental::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::experimental::disp\(\)](#)

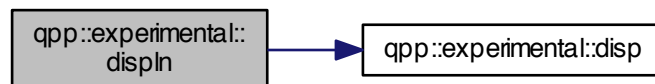
Parameters

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

Returns

Output stream

Here is the call graph for this function:



6.2.2.14 `template<typename T > std::ostream& qpp::experimental::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::experimental::disp\(\)](#)

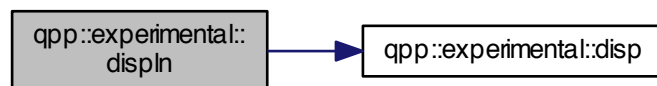
Parameters

<i>x</i>	Container
<i>separator</i>	Separator
<i>start</i>	Left marking
<i>end</i>	Right marking
<i>os</i>	Output stream

Returns

Output stream

Here is the call graph for this function:



6.2.2.15 `template<typename T> std::ostream& qpp::experimental::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::experimental::disp\(\)](#)

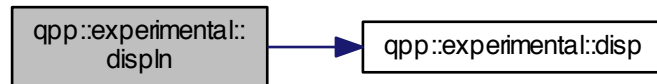
Parameters

<i>x</i>	Pointer to the first element
<i>n</i>	Number of elements to be displayed
<i>separator</i>	Separator
<i>start</i>	Left marking
<i>end</i>	Right marking
<i>os</i>	Output stream

Returns

Output stream

Here is the call graph for this function:



6.2.2.16 `template<typename Derived > std::ostream& qpp::experimental::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::experimental::disp\(\)](#)

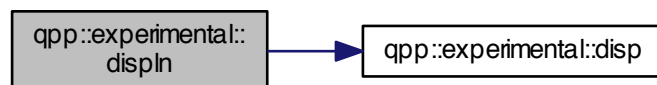
Parameters

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

Returns

Output stream

Here is the call graph for this function:



6.2.2.17 `std::ostream& qpp::experimental::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::experimental::disp\(\)](#)

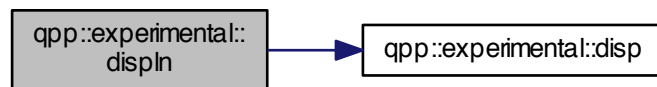
Parameters

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

Returns

Output stream

Here is the call graph for this function:



6.2.2.18 `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 $\sum_i K_i^\dagger K_i = I$

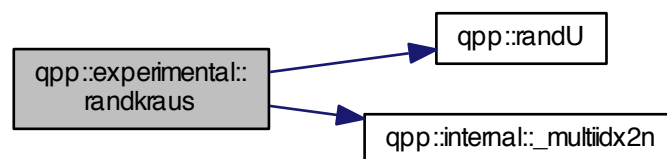
Parameters

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

Returns

Set of *n* Kraus operators satisfying the closure condition

Here is the call graph for this function:



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

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

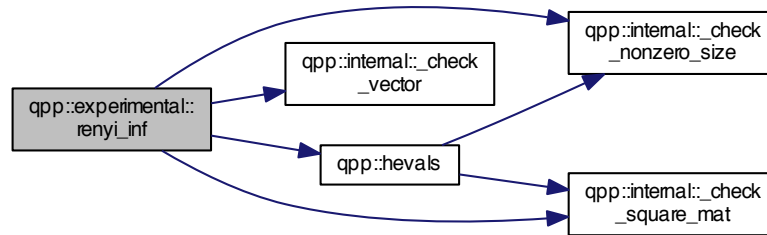
Parameters

<i>A</i>	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:



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

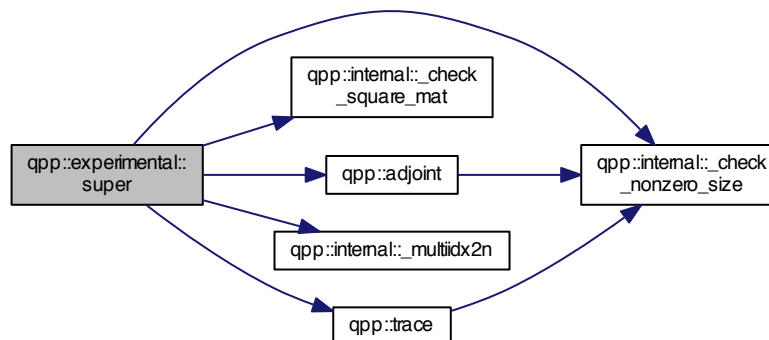
Parameters

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

Returns

Superoperator matrix representation

Here is the call graph for this function:



6.3 qpp::internal Namespace Reference

Classes

- class [IManipEigen](#)
- class [IManipPointer](#)
- class [IManipRange](#)
- 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 [_check_square_mat](#) (const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >
bool [_check_vector](#) (const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >
bool [_check_row_vector](#) (const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >
bool [_check_col_vector](#) (const Eigen::MatrixBase< Derived > &A)
- template<typename T >
bool [_check_nonzero_size](#) (const T &x)
- bool [_check_dims](#) (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)
- template<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 [_check_eq_dims](#) (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 >
[DynMat](#)< typename Derived1::Scalar > [_kron2](#) (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)
- template<typename T >
void [variadic_vector_emplace](#) (std::vector< T > &)
- template<typename T , typename First , typename... Args>
void [variadic_vector_emplace](#) (std::vector< T > &v, First &&first, Args &&...args)

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)`

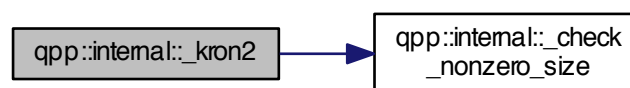
6.3.2.10 `template<typename Derived > bool qpp::internal::_check_square_mat (const Eigen::MatrixBase< Derived > & A)`

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)`

6.3.2.13 `template<typename Derived1, typename Derived2 > DynMat<typename Derived1::Scalar> qpp::internal::_kron2 (const Eigen::MatrixBase< Derived1 > & A, const Eigen::MatrixBase< Derived2 > & B)`

Here is the call graph for this function:



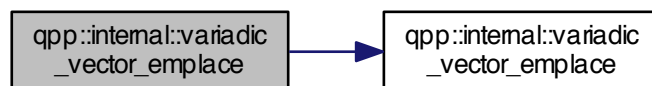
6.3.2.14 `std::size_t qpp::internal::_multiidx2n (const std::size_t * midx, std::size_t numdims, const std::size_t * dims)`
[inline]

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

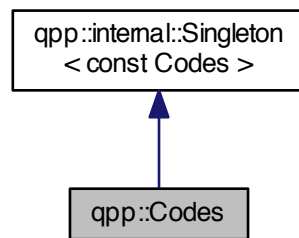
Class Documentation

7.1 qpp::Codes Class Reference

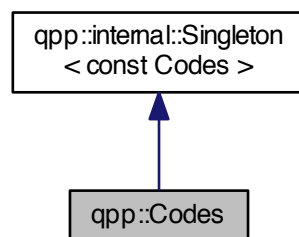
const Singleton class that defines quantum error correcting codes

```
#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 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 ()` [inline], [private]

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.

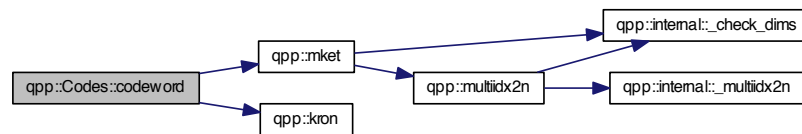
Parameters

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

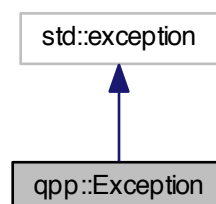
- `include/classes/codes.h`

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_CVECTOR`, `Type::MATRIX_NOT_SQUARE_OR_RVECTOR`,
`Type::MATRIX_NOT_SQUARE_OR_VECTOR`, `Type::MATRIX_MISMATCH_SUBSYS`, `Type::DIMS_INVALID`, `Type::DIMS_NOT_EQUAL`,
`Type::DIMS_MISMATCH_MATRIX`, `Type::DIMS_MISMATCH_CVECTOR`, `Type::DIMS_MISMATCH_RVECTOR`, `Type::DIMS_MISMATCH_VECTOR`,
`Type::SUBSYS_MISMATCH_DIMS`, `Type::PERM_INVALID`, `Type::NOT_QUBIT_GATE`, `Type::NOT_QUBIT_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 noexcept override
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 MATRIX_MISMATCH_SUBSYS.

DIMS_INVALID DIMS_INVALID. Matrix size mismatch subsystems' size (e.g. in `qpp::apply()`, or `qpp::channel()` `std::vector<std::size_t>` representing the dimensions has zero size or contains zeros

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 dimensions' `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 dimensions' `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 dimensions' `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 dimensions' `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 duplicates, 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 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 NO_CODEWORD. Codeword does not exist, thrown when calling `qpp::Codes::codeword()` with invalid *i*

CUSTOM_EXCEPTION CUSTOM_EXCEPTION. Custom exception, user must provide a custom message

7.2.3 Constructor & Destructor Documentation

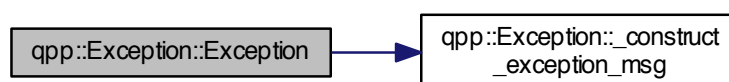
7.2.3.1 `qpp::Exception::Exception (const std::string & where, const Type & type)` `[inline]`

Constructs an exception.

Parameters

<i>where</i>	Text representing where the exception occurred
<i>type</i>	Exception 's type, see the strong enumeration <code>qpp::Exception::Type</code>

Here is the call graph for this function:



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

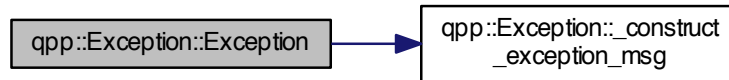
Constructs an exception.

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

Parameters

<i>where</i>	Text representing where the exception occurred
<i>custom</i>	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:

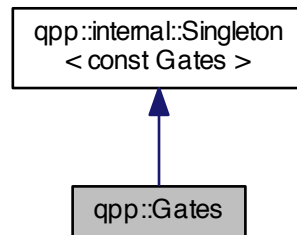
- `include/classes/exception.h`

7.3 qpp::Gates Class Reference

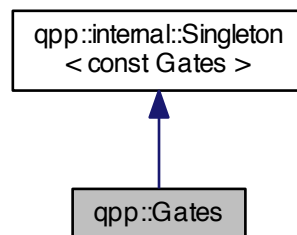
const Singleton class that implements most commonly used gates

```
#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.
- 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 multi-partite 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::Identity(4, 4)}
Controlled-NOT control target gate.
- [cmat CZ](#) {cmat::Identity(4, 4)}
Controlled-Phase gate.
- [cmat CNOTba](#) {cmat::Zero(4, 4)}
Controlled-NOT target control gate.
- [cmat SWAP](#) {cmat::Identity(4, 4)}
SWAP gate.
- [cmat TOF](#) {cmat::Identity(8, 8)}
Toffoli gate.
- [cmat FRED](#) {cmat::Identity(8, 8)}
Fredkin gate.

Private Member Functions

- [Gates](#) ()
Initializes the gates.

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 multi-partite multiple-controlled- A gate in matrix form.

Note

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

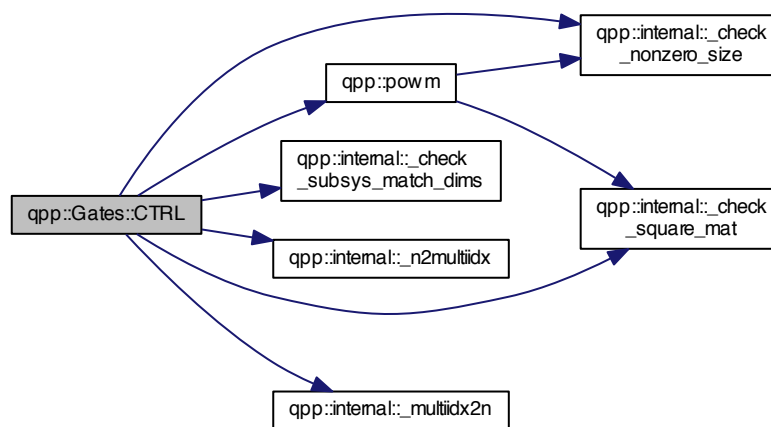
Parameters

A	Eigen expression
$ctrl$	Control subsystem indexes
$subsys$	Subsystem indexes where the gate A is applied
n	Total number of subsystems
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)`

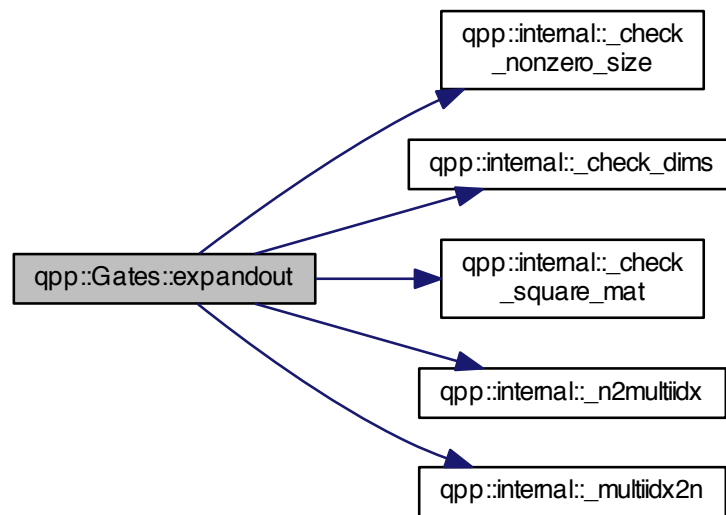
Parameters

<i>A</i>	Eigen expression
<i>pos</i>	Position
<i>dims</i>	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

<i>D</i>	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

<i>D</i>	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

<i>theta</i>	Rotation angle
<i>n</i>	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 $X = \sum_j |j \oplus 1\rangle \langle j|$

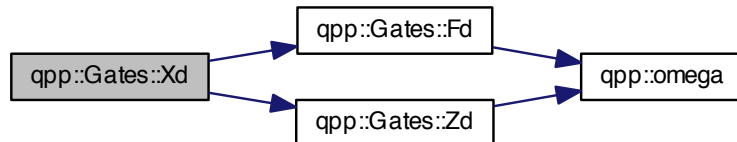
Parameters

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:

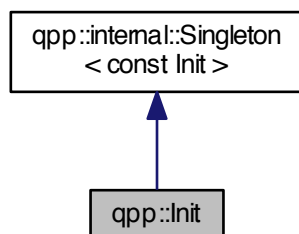
- [include/classes/gates.h](#)

7.4 qpp::Init Class Reference

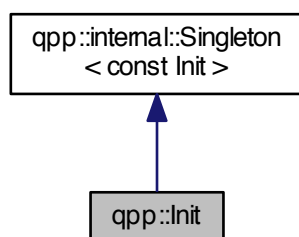
const Singleton class that performs additional initializations/cleanups

```
#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

- 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]`

Cleanups.

7.4.3 Friends And Related Function Documentation

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

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

- `include/classes/init.h`

7.5 `qpp::internal::IOManipEigen` Class Reference

```
#include <iomanip.h>
```

Public Member Functions

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

Private Attributes

- `cmat_A`
- double `_chop`

Friends

- `template<typename charT , typename traits >`
`std::basic_ostream< charT,`
`traits > & operator<< (std::basic_ostream< charT, traits > &os, const IOManipEigen &rhs)`

7.5.1 Constructor & Destructor Documentation

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

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

7.5.2 Friends And Related Function Documentation

7.5.2.1 `template<typename charT , typename traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > &os, const IOManipEigen & rhs)` [`friend`]

7.5.3 Member Data Documentation

7.5.3.1 `cmat qpp::internal::IOManipEigen::_A` [`private`]

7.5.3.2 `double qpp::internal::IOManipEigen::_chop` [`private`]

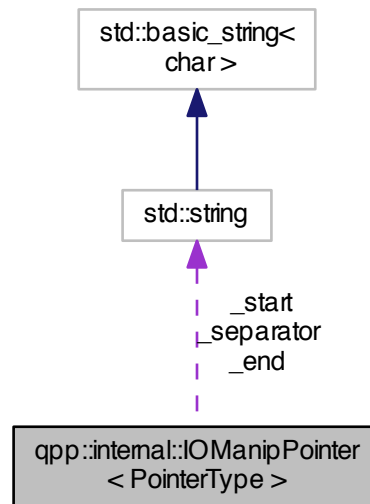
The documentation for this class was generated from the following file:

- `include/internal/classes/iomanip.h`

7.6 qpp::internal::IOManipPointer< PointerType > Class Template Reference

```
#include <iomanip.h>
```

Collaboration diagram for `qpp::internal::IOManipPointer< PointerType >`:



Public Member Functions

- [IOManipPointer](#) (const PointerType *p, const std::size_t n, const std::string &separator, const std::string &start="[", const std::string &end="]")
- [IOManipPointer](#) (const [IOManipPointer](#) &)=default
- [IOManipPointer](#) & [operator=](#) (const [IOManipPointer](#) &)=default

Private Attributes

- const PointerType * [_p](#)
- std::size_t [_n](#)
- std::string [_separator](#)
- std::string [_start](#)
- std::string [_end](#)

Friends

- template<typename charT, typename traits >
std::basic_ostream< charT,
traits > & [operator<<](#) (std::basic_ostream< charT, traits > &os, const [IOManipPointer](#) &rhs)

7.6.1 Constructor & Destructor Documentation

- 7.6.1.1 `template<typename PointerType > qpp::internal::IOManipPointer< PointerType >::IOManipPointer (const PointerType * p, const std::size_t n, const std::string & separator, const std::string & start = " [", const std::string & end = "] ") [inline],[explicit]`

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

7.6.2 Member Function Documentation

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

7.6.3 Friends And Related Function Documentation

7.6.3.1 `template<typename PointerType > template<typename charT , typename traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & os, const IOManipPointer< PointerType > & rhs)` [friend]

7.6.4 Member Data Documentation

7.6.4.1 `template<typename PointerType > std::string qpp::internal::IOManipPointer< PointerType >::_end` [private]

7.6.4.2 `template<typename PointerType > std::size_t qpp::internal::IOManipPointer< PointerType >::_n` [private]

7.6.4.3 `template<typename PointerType > const PointerType* qpp::internal::IOManipPointer< PointerType >::_p` [private]

7.6.4.4 `template<typename PointerType > std::string qpp::internal::IOManipPointer< PointerType >::_separator` [private]

7.6.4.5 `template<typename PointerType > std::string qpp::internal::IOManipPointer< PointerType >::_start` [private]

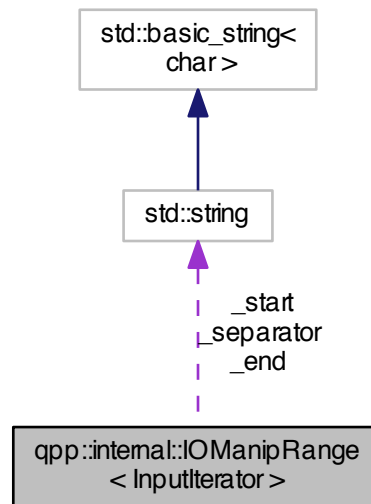
The documentation for this class was generated from the following file:

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

7.7 qpp::internal::IOManipRange< InputIterator > Class Template Reference

```
#include <iomanip.h>
```

Collaboration diagram for `qpp::internal::IOManipRange< InputIterator >`:



Public Member Functions

- [IOManipRange](#) (`InputIterator first`, `InputIterator last`, `const std::string &separator`, `const std::string &start="["`, `const std::string &end="]"`)

Private Attributes

- `InputIterator _first`
- `InputIterator _last`
- `std::string _separator`
- `std::string _start`
- `std::string _end`

Friends

- `template<typename charT, typename traits >`
`std::basic_ostream< charT,`
`traits > & operator<< (std::basic_ostream< charT, traits > &os, const IOManipRange &rhs)`

7.7.1 Constructor & Destructor Documentation

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

7.7.2 Friends And Related Function Documentation

7.7.2.1 `template<typename InputIterator > template<typename charT , typename traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & os, const IOManipRange< InputIterator > & rhs)`
`[friend]`

7.7.3 Member Data Documentation

7.7.3.1 `template<typename InputIterator > std::string qpp::internal::IOManipRange< InputIterator >::_end`
`[private]`

7.7.3.2 `template<typename InputIterator > InputIterator qpp::internal::IOManipRange< InputIterator >::_first`
`[private]`

7.7.3.3 `template<typename InputIterator > InputIterator qpp::internal::IOManipRange< InputIterator >::_last`
`[private]`

7.7.3.4 `template<typename InputIterator > std::string qpp::internal::IOManipRange< InputIterator >::_separator`
`[private]`

7.7.3.5 `template<typename InputIterator > std::string qpp::internal::IOManipRange< InputIterator >::_start`
`[private]`

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

- `include/internal/classes/iomanip.h`

7.8 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

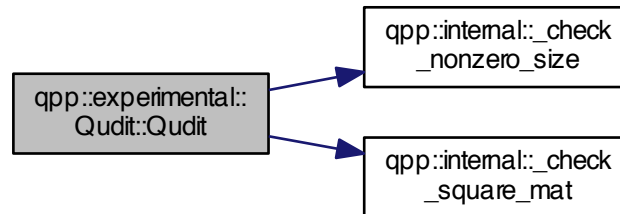
Private Attributes

- `cmat_rho`
- `std::size_t _D`

7.8.1 Constructor & Destructor Documentation

7.8.1.1 `qpp::experimental::Qudit::Qudit (const cmat & rho = States::get_instance().pz0) [inline]`

Here is the call graph for this function:



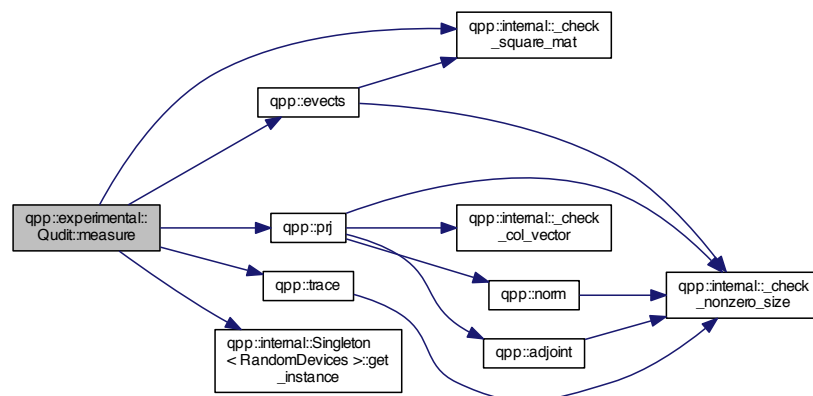
7.8.2 Member Function Documentation

7.8.2.1 `std::size_t qpp::experimental::Qudit::getD () const [inline]`

7.8.2.2 `cmat qpp::experimental::Qudit::getRho () const [inline]`

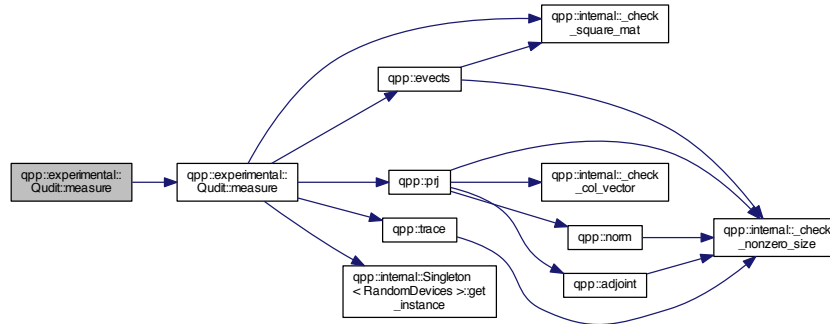
7.8.2.3 `std::size_t qpp::experimental::Qudit::measure (const cmat & U, bool destructive = false) [inline]`

Here is the call graph for this function:



7.8.2.4 `std::size_t qpp::experimental::Qudit::measure (bool destructive = false) [inline]`

Here is the call graph for this function:



7.8.3 Member Data Documentation

7.8.3.1 `std::size_t qpp::experimental::Qudit::_D [private]`

7.8.3.2 `cmat qpp::experimental::Qudit::_rho [private]`

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

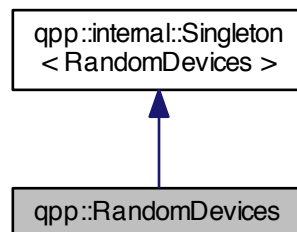
- [include/experimental/classes/qudit.h](#)

7.9 qpp::RandomDevices Class Reference

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

```
#include <random_devices.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

7.9.1 Detailed Description

Singleton 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.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 `qpp::RandomDevices::RandomDevices ()` `[inline]`, `[private]`

Initializes and seeds the random number generators.

7.9.3 Friends And Related Function Documentation

7.9.3.1 friend class internal::Singleton< RandomDevices > [friend]

7.9.4 Member Data Documentation

7.9.4.1 std::random_device qpp::RandomDevices::_rd [private]

used to seed std::mt19937 _rng

7.9.4.2 std::mt19937 qpp::RandomDevices::_rng

Mersenne twister random number generator engine.

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

- include/classes/random_devices.h

7.10 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

7.10.1 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::Codes](#), [qpp::Gates](#), [qpp::RandomDevices](#), [qpp::States](#) or [qpp.h](#) for real world examples of usage.

7.10.2 Constructor & Destructor Documentation

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

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

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

7.10.3 Member Function Documentation

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

7.10.3.2 `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:

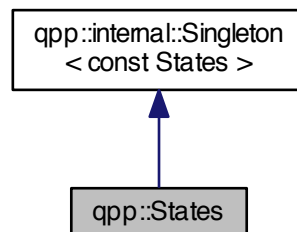
- [include/classes/singleton.h](#)

7.11 qpp::States Class Reference

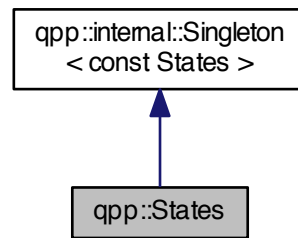
const Singleton class that implements most commonly used states

```
#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 $|+\rangle$
- **ket x1** {ket::Zero(2)}
Pauli Sigma-X 1-eigenstate $|-\rangle$
- **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 $|0\rangle$
- **ket z1** {ket::Zero(2)}
Pauli Sigma-Z 1-eigenstate $|1\rangle$
- **cmat px0** {cmat::Zero(2, 2)}
Projector onto the Pauli Sigma-X 0-eigenstate $|+\rangle\langle+|$.
- **cmat px1** {cmat::Zero(2, 2)}
Projector onto the Pauli Sigma-X 1-eigenstate $|-\rangle\langle-|$.
- **cmat py0** {cmat::Zero(2, 2)}
Projector onto the Pauli Sigma-Y 0-eigenstate.
- **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 $|0\rangle\langle 0|$.
- **cmat pz1** {cmat::Zero(2, 2)}
Projector onto the Pauli Sigma-Z 1-eigenstate $|1\rangle\langle 1|$.
- **ket b00** {ket::Zero(4)}
Bell-00 state (following the convention in Nielsen and Chuang)
- **ket b01** {ket::Zero(4)}
Bell-01 state (following the convention in Nielsen and Chuang)
- **ket b10** {ket::Zero(4)}
Bell-10 state (following the convention in Nielsen and Chuang)
- **ket b11** {ket::Zero(4)}
Bell-11 state (following the convention in Nielsen and Chuang)
- **cmat pb00** {cmat::Zero(4, 4)}

- Projector onto the Bell-00 state.*
- [cmat pb01](#) {cmat::Zero(4, 4)}
- Projector onto the Bell-01 state.*
- [cmat pb10](#) {cmat::Zero(4, 4)}
- Projector onto the Bell-10 state.*
- [cmat pb11](#) {cmat::Zero(4, 4)}
- Projector onto the Bell-11 state.*
- [ket GHZ](#) {ket::Zero(8)}
- GHZ state.*
- [ket W](#) {ket::Zero(8)}
- W state.*
- [cmat pGHZ](#) {cmat::Zero(8, 8)}
- Projector onto the GHZ state.*
- [cmat pW](#) {cmat::Zero(8, 8)}
- Projector onto the W state.*

Private Member Functions

- [States](#) ()

Friends

- class [internal::Singleton< const States >](#)

Additional Inherited Members

7.11.1 Detailed Description

const Singleton class that implements most commonly used states

7.11.2 Constructor & Destructor Documentation

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

Initialize the states

7.11.3 Friends And Related Function Documentation

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

7.11.4 Member Data Documentation

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

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

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

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

7.11.4.3 ket qpp::States::b10 {ket::Zero(4)}

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

7.11.4.4 ket qpp::States::b11 {ket::Zero(4)}

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

7.11.4.5 ket qpp::States::GHZ {ket::Zero(8)}

GHZ state.

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

Projector onto the Bell-00 state.

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

Projector onto the Bell-01 state.

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

Projector onto the Bell-10 state.

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

Projector onto the Bell-11 state.

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

Projector onto the GHZ state.

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

Projector onto the W state.

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

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

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

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

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

Projector onto the Pauli Sigma-Y 0-eigenstate.

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

Projector onto the Pauli Sigma-Y 1-eigenstate.

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

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

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

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

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

W state.

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

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

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

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

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

Pauli Sigma-Y 0-eigenstate.

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

Pauli Sigma-Y 1-eigenstate.

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

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

7.11.4.24 `ket qpp::States::z1 {ket::Zero(2)}`

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

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

- [include/classes/states.h](#)

7.12 `qpp::Timer` Class Reference

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)
Overload for std::ostream operators.

7.12.1 Detailed Description

Measures time.

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

7.12.2 Constructor & Destructor Documentation

7.12.2.1 qpp::Timer::Timer () [\[inline\]](#)

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

7.12.3 Member Function Documentation

7.12.3.1 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\(\)](#)

7.12.3.2 void qpp::Timer::tic () [\[inline\]](#)

Resets the chronometer.

Resets the starting/ending point to the current time

7.12.3.3 `const Timer& qpp::Timer::toc ()` `[inline]`

Stops the chronometer.

Set the current time as the ending point

Returns

Current instance

7.12.4 Friends And Related Function Documentation

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

Overload for `std::ostream` operators.

Parameters

<i>os</i>	Output stream
<i>rhs</i>	Timer instance

Returns

Writes to the output stream the number of seconds that passed between the instantiation/reset and invocation of `qpp::Timer::toc()`.

7.12.5 Member Data Documentation

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

7.12.5.2 `std::chrono::steady_clock::time_point qpp::Timer::_start` `[protected]`

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

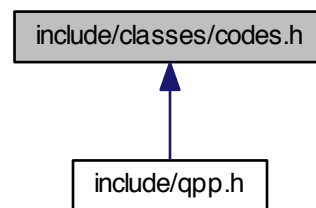
- `include/classes/timer.h`

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](#)

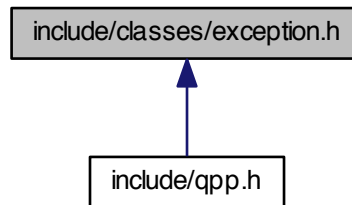
const Singleton class that defines quantum error correcting codes

Namespaces

- [qpp](#)

8.2 include/classes/exception.h File Reference

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



Classes

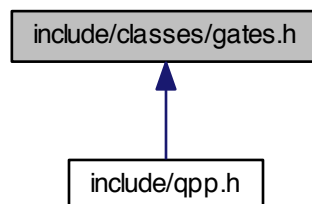
- class [qpp::Exception](#)
Generates custom exceptions, used when validating function parameters.

Namespaces

- [qpp](#)

8.3 include/classes/gates.h File Reference

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



Classes

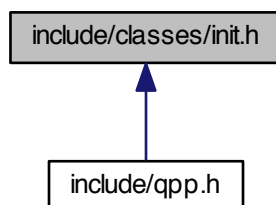
- class [qpp::Gates](#)
const Singleton class that implements most commonly used gates

Namespaces

- [qpp](#)

8.4 include/classes/init.h File Reference

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



Classes

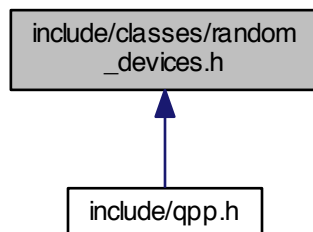
- class [qpp::Init](#)
const Singleton class that performs additional initializations/cleanups

Namespaces

- [qpp](#)

8.5 include/classes/random_devices.h File Reference

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



Classes

- class [qpp::RandomDevices](#)

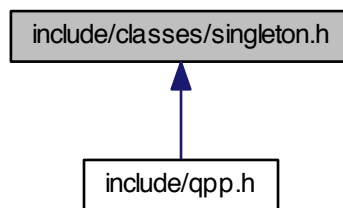
Singleton class that manages the source of randomness in the library.

Namespaces

- [qpp](#)

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 >](#)

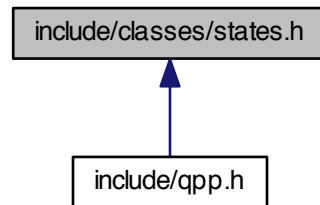
[Singleton](#) policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern)

Namespaces

- [qpp](#)
- [qpp::internal](#)

8.7 include/classes/states.h File Reference

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



Classes

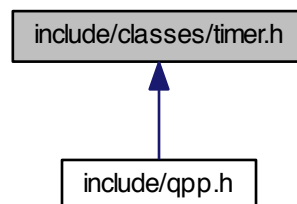
- class [qpp::States](#)
const Singleton class that implements most commonly used states

Namespaces

- [qpp](#)

8.8 include/classes/timer.h File Reference

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



Classes

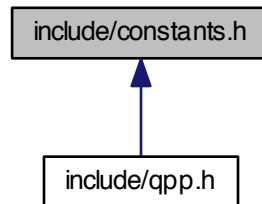
- class [qpp::Timer](#)
Measures time.

Namespaces

- [qpp](#)

8.9 include/constants.h File Reference

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



Namespaces

- [qpp](#)

Functions

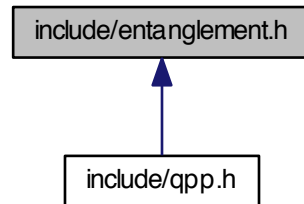
- constexpr std::complex< double > [qpp::operator""_i](#) (unsigned long long int x)
User-defined literal for complex $i = \sqrt{-1}$ (integer overload)
- constexpr std::complex< double > [qpp::operator""_i](#) (long double x)
User-defined literal for complex $i = \sqrt{-1}$ (real overload)
- std::complex< double > [qpp::omega](#) (std::size_t D)
D-th root of unity.

Variables

- constexpr double [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](#).
- 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::infy](#) = -1
Used to denote infinity.

8.10 include/entanglement.h File Reference

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



Namespaces

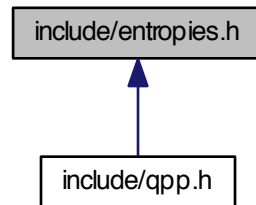
- [qpp](#)

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 >`
`cmat 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 >`
`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.
- `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.
- `template<typename Derived >`
`double qpp::gconcurrence (const Eigen::MatrixBase< Derived > &A)`
G-concurrence of the bi-partite pure state A.
- `template<typename Derived >`
`double qpp::negativity (const Eigen::MatrixBase< Derived > &A, const std::vector< 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 qpp::concurrence (const Eigen::MatrixBase< 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

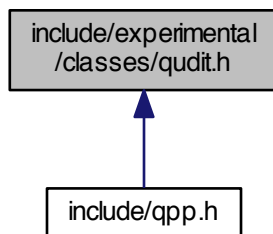
- [qpp](#)

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- α 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- α 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

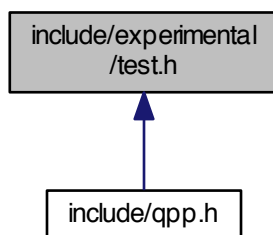
- class [qpp::experimental::Qudit](#)

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](#)
- [qpp](#)

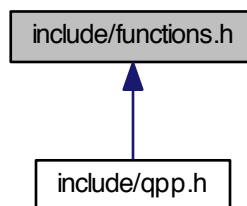
Functions

- `template<typename Derived1 , typename Derived2 >`
`DynMat< typename Derived1::Scalar > qpp::experimental::apply_old (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 multi-partite state vector or density matrix.
- `template<typename Derived >`
`cmat qpp::experimental::channel_old (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 multi-partite 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 qpp::experimental::renyi_inf (const Eigen::MatrixBase< Derived > &A)`
Renyi- ∞ entropy (min entropy) of the probability distribution/density matrix A.
- `template<typename InputIterator >`
`void qpp::experimental::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::experimental::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::experimental::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::experimental::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 & qpp::experimental::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::experimental::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::experimental::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::experimental::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::experimental::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::experimental::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 Derived1 , typename Derived2 >`
`DynMat< typename Derived1::Scalar > qpp::experimental::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, const std::vector< std::size_t > &dims)`
Applies the controlled-gate A to the part subsys of a multi-partite state vector or density matrix.
- `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 multi-partite 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.

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)`
Inverse.
- `template<typename Derived >`
`Derived::Scalar qpp::trace (const Eigen::MatrixBase< Derived > &A)`
Trace.
- `template<typename Derived >`
`Derived::Scalar qpp::det (const Eigen::MatrixBase< Derived > &A)`
Determinant.
- `template<typename Derived >`
`Derived::Scalar qpp::logdet (const Eigen::MatrixBase< Derived > &A)`
Logarithm of the determinant.
- `template<typename Derived >`
`Derived::Scalar qpp::sum (const Eigen::MatrixBase< Derived > &A)`
Element-wise sum of A.
- `template<typename Derived >`
`Derived::Scalar qpp::prod (const Eigen::MatrixBase< Derived > &A)`
Element-wise product of A.
- `template<typename Derived >`
`double qpp::norm (const Eigen::MatrixBase< Derived > &A)`
Frobenius norm.
- `template<typename Derived >`
`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)
- Matrix sin.*

 - template<typename Derived >
cmat [qpp::cosm](#) (const Eigen::MatrixBase< Derived > &A)
- Matrix cos.*

 - template<typename Derived >
cmat [qpp::spectralpovm](#) (const Eigen::MatrixBase< Derived > &A, const cplx z)
- Matrix power.*

 - template<typename Derived >
DynMat< typename Derived::Scalar > [qpp::povm](#) (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)
- Schatten norm.*

 - template<typename OutputScalar , typename Derived >
DynMat< OutputScalar > [qpp::cwise](#) (const Eigen::MatrixBase< Derived > &A, OutputScalar(*f)(const typename 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.*

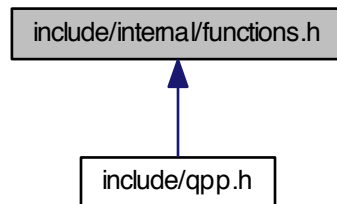
 - template<typename Derived >
DynMat< typename Derived::Scalar > [qpp::reshape](#) (const Eigen::MatrixBase< Derived > &A, std::size_t rows, std::size_t cols)
- Reshape.*

 - 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)
- `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.
- `template<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](#)

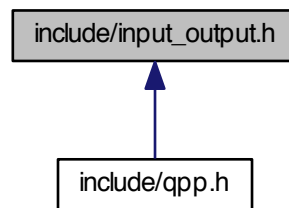
Functions

- void [qpp::internal::_n2multiidx](#) (std::size_t n, std::size_t numdims, const std::size_t *dims, std::size_t *result)
- std::size_t [qpp::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::MatrixBase< Derived > &A)
- template<typename Derived >
bool [qpp::internal::_check_dims_match_cvect](#) (const std::vector< std::size_t > &dims, const Eigen::MatrixBase< Derived > &V)
- template<typename Derived >
bool [qpp::internal::_check_dims_match_rvect](#) (const std::vector< std::size_t > &dims, const Eigen::MatrixBase< 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/input_output.h File Reference

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



Namespaces

- [qpp](#)

Functions

- `template<typename Derived >`
`internal::IOManipEigen qpp::disp (const Eigen::MatrixBase< Derived > &A, double chop=qpp::chop)`
Eigen expression ostream manipulator.
- `internal::IOManipEigen qpp::disp (cplx z, double chop=qpp::chop)`
Complex number ostream manipulator.
- `template<typename InputIterator >`
`internal::IOManipRange`
`< InputIterator > qpp::disp (const InputIterator &first, const InputIterator &last, const std::string &separator,`
`const std::string &start="[", const std::string &end=""]")`
Range ostream manipulator.
- `template<typename Container >`
`internal::IOManipRange`
`< typename`
`Container::const_iterator > qpp::disp (const Container &c, const std::string &separator, const std::string`
`&start="[", const std::string &end=""]")`
Standard container ostream manipulator. The container must support std::begin(), std::end() and forward iteration.
- `template<typename PointerType >`
`internal::IOManipPointer`
`< PointerType > qpp::disp (const PointerType *p, std::size_t n, const std::string &separator, const std::string`
`&start="[", const std::string &end=""]")`
C-style pointer ostream manipulator.
- `template<typename Derived >`
`void qpp::save (const Eigen::MatrixBase< Derived > &A, const std::string &fname)`

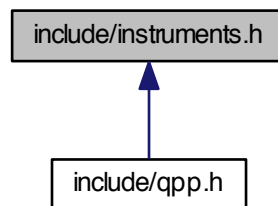
Saves Eigen expression to a binary file (internal format) in double precision.

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

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

8.17 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)`

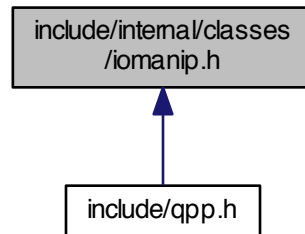
Measures the state A using the set of Kraus operators Ks (std::initializer_list overload)

- `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.

8.18 include/internal/classes/iomanip.h File Reference

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



Classes

- class `qpp::internal::IOManipRange< InputIterator >`
- class `qpp::internal::IOManipPointer< PointerType >`
- class `qpp::internal::IOManipEigen`

Namespaces

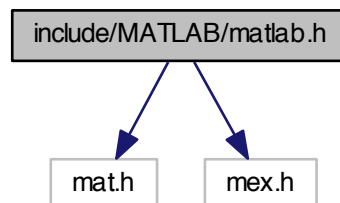
- `qpp`
- `qpp::internal`

8.19 include/MATLAB/matlab.h File Reference

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

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

Include dependency graph for matlab.h:



Namespaces

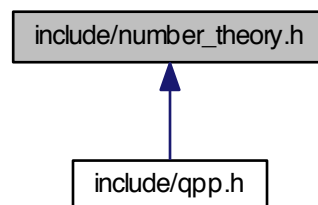
- [qpp](#)

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](#))

8.20 include/number_theory.h File Reference

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



Namespaces

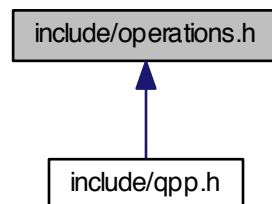
- [qpp](#)

Functions

- `std::vector< int > qpp::x2contfrac` (double x, `std::size_t` n, `std::size_t` cut=1e5)
Simple continued fraction expansion.
- `double qpp::contfrac2x` (const `std::vector< int >` &cf, `std::size_t` n)
Real representation of a simple continued fraction.
- `double qpp::contfrac2x` (const `std::vector< int >` &cf)
Real representation of a simple continued fraction.
- `std::size_t qpp::gcd` (`std::size_t` m, `std::size_t` n)
Greatest common divisor of two non-negative integers.
- `std::size_t qpp::gcd` (const `std::vector< std::size_t >` &ns)
Greatest common divisor of a list of non-negative integers.
- `std::size_t qpp::lcm` (`std::size_t` m, `std::size_t` n)
Least common multiple of two positive integers.
- `std::size_t qpp::lcm` (const `std::vector< std::size_t >` &ns)
Least common multiple of a list of positive integers.
- `std::vector< std::size_t > qpp::invperm` (const `std::vector< std::size_t >` &perm)
Inverse permutation.
- `std::vector< std::size_t > qpp::compperm` (const `std::vector< std::size_t >` &perm, const `std::vector< std::size_t >` &sigma)
Compose permutations.

8.21 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, const `std::vector< std::size_t >` &dims)
Applies the controlled-gate A to the part subsys of a multi-partite state vector or density matrix.

- `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 multi-partite 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, const std::vector< std::size_t > &dims)`
Applies the gate A to the part subsys of a multi-partite 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 d=2)`
Applies the gate A to the part subsys of a multi-partite 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, 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.
- `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 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 >`
`DynMat< typename Derived::Scalar > qpp::ptrace1 (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &dims)`
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 >`
`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.
- `template<typename Derived >`
`DynMat< typename Derived::Scalar > qpp::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 > qpp::syspermute (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &perm, const std::vector< std::size_t > &dims)`
System permutation.

8.22 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 "internal/classes/iomanip.h"
#include "input_output.h"
#include "classes/singleton.h"
#include "classes/init.h"
#include "internal/functions.h"
#include "functions.h"
#include "classes/codes.h"
#include "classes/gates.h"
#include "classes/states.h"
#include "classes/random_devices.h"
#include "operations.h"
#include "entropies.h"
#include "entanglement.h"
#include "classes/timer.h"
#include "instruments.h"
#include "number_theory.h"
#include "random.h"
#include "experimental/test.h"
#include "experimental/classes/qudit.h"
```

Include dependency graph for qpp.h:



Namespaces

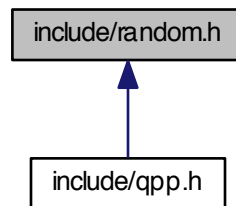
- [qpp](#)

Variables

- const Init & [qpp::init](#) = Init::get_instance()
[qpp::Init](#) const Singleton
- const Codes & [qpp::codes](#) = Codes::get_instance()
[qpp::Codes](#) const Singleton
- const Gates & [qpp::gt](#) = Gates::get_instance()
[qpp::Gates](#) const Singleton
- const States & [qpp::st](#) = States::get_instance()
[qpp::States](#) const Singleton
- RandomDevices & [qpp::rdevs](#) = RandomDevices::get_instance()
[qpp::RandomDevices](#) Singleton

8.23 include/random.h File Reference

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



Namespaces

- [qpp](#)

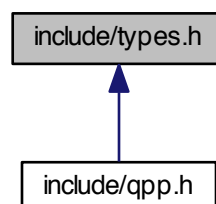
Functions

- 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 [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 $N(\text{mean}, \text{sigma})$*
- 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 $N(\text{mean}, \text{sigma})$, 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 $N(\text{mean}, \text{sigma})$, specialization for complex matrices ([qpp::cmat](#))*
- double [qpp::randn](#) (double mean=0, double sigma=1)
 - Generates a random real number (double) normally distributed in $N(\text{mean}, \text{sigma})$*
- cmat [qpp::randU](#) (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.*

8.24 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 [qpp::DynMat](#) = 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.
- template<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.

8.25 mainpage.dox File Reference

Index

- absm
 - qpp, [22](#)
- abssq
 - qpp, [23](#)
- adjoint
 - qpp, [24](#)
- anticomm
 - qpp, [24](#)
- apply
 - qpp, [25](#), [26](#)
- bra
 - qpp, [21](#)
- CUSTOM_EXCEPTION
 - qpp::Exception, [124](#)
- channel
 - qpp, [29–31](#)
- choi
 - qpp, [32](#)
- choi2kraus
 - qpp, [32](#)
- chop
 - qpp, [97](#)
- cmat
 - qpp, [21](#)
- codes
 - qpp, [97](#)
- comm
 - qpp, [33](#)
- compperm
 - qpp, [34](#)
- concurrence
 - qpp, [34](#)
- conjugate
 - qpp, [35](#)
- contrfrac2x
 - qpp, [35](#), [37](#)
- cosm
 - qpp, [37](#)
- cplx
 - qpp, [22](#)
- cwise
 - qpp, [37](#)
- DIMS_INVALID
 - qpp::Exception, [123](#)
- DIMS_MISMATCH_CVECTOR
 - qpp::Exception, [123](#)
- DIMS_MISMATCH_MATRIX
 - qpp::Exception, [123](#)
- DIMS_MISMATCH_RVECTOR
 - qpp::Exception, [123](#)
- DIMS_MISMATCH_VECTOR
 - qpp::Exception, [123](#)
- DIMS_NOT_EQUAL
 - qpp::Exception, [123](#)
- det
 - qpp, [38](#)
- disp
 - qpp, [38](#), [39](#)
- dmat
 - qpp, [22](#)
- ee
 - qpp, [97](#)
- entanglement
 - qpp, [41](#)
- eps
 - qpp, [97](#)
- evals
 - qpp, [42](#)
- evects
 - qpp, [42](#)
- expm
 - qpp, [42](#)
- FIVE_QUBIT
 - qpp::Codes, [120](#)
- funm
 - qpp, [44](#)
- gcd
 - qpp, [44](#), [45](#)
- gconcurrence
 - qpp, [45](#)
- grams
 - qpp, [46](#), [47](#)
- gt
 - qpp, [98](#)
- hevals
 - qpp, [47](#)
- hevects
 - qpp, [48](#)
- infty
 - qpp, [98](#)
- init
 - qpp, [98](#)
- inverse

- qpp, 48
- invperm
 - qpp, 50
- ket
 - qpp, 22
- kron
 - qpp, 50–52
- kronpow
 - qpp, 52
- lcm
 - qpp, 53
- load
 - qpp, 54
- logdet
 - qpp, 55
- logm
 - qpp, 56
- lognegativity
 - qpp, 56
- MATRIX_MISMATCH_SUBSYS
 - qpp::Exception, 123
- MATRIX_NOT_CVECTOR
 - qpp::Exception, 123
- MATRIX_NOT_RVECTOR
 - qpp::Exception, 123
- MATRIX_NOT_SQUARE
 - qpp::Exception, 123
- MATRIX_NOT_SQUARE_OR_CVECTOR
 - qpp::Exception, 123
- MATRIX_NOT_SQUARE_OR_RVECTOR
 - qpp::Exception, 123
- MATRIX_NOT_SQUARE_OR_VECTOR
 - qpp::Exception, 123
- MATRIX_NOT_VECTOR
 - qpp::Exception, 123
- maxn
 - qpp, 98
- measure
 - qpp, 57, 58
- mket
 - qpp, 60, 61
- mprj
 - qpp, 61, 62
- multiidx2n
 - qpp, 62
- n2multiidx
 - qpp, 63
- NINE_QUBIT_SHOR
 - qpp::Codes, 120
- NO_CODEWORD
 - qpp::Exception, 124
- NOT_BIPARTITE
 - qpp::Exception, 124
- NOT_QUBIT_GATE
 - qpp::Exception, 124
- NOT_QUBIT_SUBSYS
 - qpp::Exception, 124
- negativity
 - qpp, 63
- norm
 - qpp, 65
- OUT_OF_RANGE
 - qpp::Exception, 124
- omega
 - qpp, 66
- PERM_INVALID
 - qpp::Exception, 123
- pi
 - qpp, 98
- powm
 - qpp, 66
- prj
 - qpp, 67
- prod
 - qpp, 68
- ptrace
 - qpp, 69
- ptrace1
 - qpp, 70
- ptrace2
 - qpp, 71
- ptranspose
 - qpp, 72
- qmutualinfo
 - qpp, 73
- qpp, 13
 - absm, 22
 - abssq, 23
 - adjoint, 24
 - anticomm, 24
 - apply, 25, 26
 - bra, 21
 - channel, 29–31
 - choi, 32
 - choi2kraus, 32
 - chop, 97
 - cmat, 21
 - codes, 97
 - comm, 33
 - compperm, 34
 - concurrence, 34
 - conjugate, 35
 - contfrac2x, 35, 37
 - cosm, 37
 - cplx, 22
 - cwise, 37
 - det, 38
 - disp, 38, 39
 - dmat, 22
 - ee, 97
 - entanglement, 41

- eps, [97](#)
- evals, [42](#)
- evects, [42](#)
- expm, [42](#)
- funm, [44](#)
- gcd, [44](#), [45](#)
- gconcurrency, [45](#)
- grams, [46](#), [47](#)
- gt, [98](#)
- hevals, [47](#)
- hevects, [48](#)
- infty, [98](#)
- init, [98](#)
- inverse, [48](#)
- invperm, [50](#)
- ket, [22](#)
- kron, [50–52](#)
- kronpow, [52](#)
- lcm, [53](#)
- load, [54](#)
- logdet, [55](#)
- logm, [56](#)
- lognegativity, [56](#)
- maxn, [98](#)
- measure, [57](#), [58](#)
- mket, [60](#), [61](#)
- mprj, [61](#), [62](#)
- multiidx2n, [62](#)
- n2multiidx, [63](#)
- negativity, [63](#)
- norm, [65](#)
- omega, [66](#)
- pi, [98](#)
- powm, [66](#)
- prj, [67](#)
- prod, [68](#)
- ptrace, [69](#)
- ptrace1, [70](#)
- ptrace2, [71](#)
- ptranspose, [72](#)
- qmutualinfo, [73](#)
- rand, [74](#), [75](#)
- randint, [76](#)
- randket, [77](#)
- randkraus, [77](#)
- randn, [78](#), [79](#)
- randperm, [80](#)
- randrho, [80](#)
- rdevs, [98](#)
- renyi, [81](#)
- reshape, [82](#)
- save, [83](#)
- schatten, [84](#)
- schmidtcoeff, [85](#)
- schmidtprob, [86](#)
- shannon, [89](#)
- sinm, [90](#)
- spectralpowm, [90](#)
- sqrtm, [91](#)
- st, [98](#)
- sum, [91](#), [92](#)
- super, [92](#)
- svals, [93](#)
- syspermute, [94](#)
- trace, [95](#)
- transpose, [96](#)
- tsallis, [96](#)
- x2contfrac, [97](#)
- qpp::Codes
 - FIVE_QUBIT, [120](#)
 - NINE_QUBIT_SHOR, [120](#)
 - SEVEN_QUBIT_STEANE, [120](#)
- qpp::Exception
 - CUSTOM_EXCEPTION, [124](#)
 - DIMS_INVALID, [123](#)
 - DIMS_MISMATCH_CVECTOR, [123](#)
 - DIMS_MISMATCH_MATRIX, [123](#)
 - DIMS_MISMATCH_RVECTOR, [123](#)
 - DIMS_MISMATCH_VECTOR, [123](#)
 - DIMS_NOT_EQUAL, [123](#)
 - MATRIX_MISMATCH_SUBSYS, [123](#)
 - MATRIX_NOT_CVECTOR, [123](#)
 - MATRIX_NOT_RVECTOR, [123](#)
 - MATRIX_NOT_SQUARE, [123](#)
 - MATRIX_NOT_SQUARE_OR_CVECTOR, [123](#)
 - MATRIX_NOT_SQUARE_OR_RVECTOR, [123](#)
 - MATRIX_NOT_SQUARE_OR_VECTOR, [123](#)
 - MATRIX_NOT_VECTOR, [123](#)
 - NO_CODEWORD, [124](#)
 - NOT_BIPARTITE, [124](#)
 - NOT_QUBIT_GATE, [124](#)
 - NOT_QUBIT_SUBSYS, [124](#)
 - OUT_OF_RANGE, [124](#)
 - PERM_INVALID, [123](#)
 - SUBSYS_MISMATCH_DIMS, [123](#)
 - TYPE_MISMATCH, [124](#)
 - UNDEFINED_TYPE, [124](#)
 - UNKNOWN_EXCEPTION, [123](#)
 - ZERO_SIZE, [123](#)
- rand
 - qpp, [74](#), [75](#)
- randint
 - qpp, [76](#)
- randket
 - qpp, [77](#)
- randkraus
 - qpp, [77](#)
- randn
 - qpp, [78](#), [79](#)
- randperm
 - qpp, [80](#)
- randrho
 - qpp, [80](#)
- rdevs
 - qpp, [98](#)
- renyi

- qpp, [81](#)
- reshape
 - qpp, [82](#)
- SEVEN_QUBIT_STEANE
 - qpp::Codes, [120](#)
- SUBSYS_MISMATCH_DIMS
 - qpp::Exception, [123](#)
- save
 - qpp, [83](#)
- schatten
 - qpp, [84](#)
- schmidtcoeff
 - qpp, [85](#)
- schmidtprob
 - qpp, [86](#)
- shannon
 - qpp, [89](#)
- sinm
 - qpp, [90](#)
- spectralpowm
 - qpp, [90](#)
- sqrtn
 - qpp, [91](#)
- st
 - qpp, [98](#)
- sum
 - qpp, [91](#), [92](#)
- super
 - qpp, [92](#)
- svals
 - qpp, [93](#)
- syspermute
 - qpp, [94](#)
- TYPE_MISMATCH
 - qpp::Exception, [124](#)
- trace
 - qpp, [95](#)
- transpose
 - qpp, [96](#)
- tsallis
 - qpp, [96](#)
- UNDEFINED_TYPE
 - qpp::Exception, [124](#)
- UNKNOWN_EXCEPTION
 - qpp::Exception, [123](#)
- x2contfrac
 - qpp, [97](#)
- ZERO_SIZE
 - qpp::Exception, [123](#)