

quantum++
0.1

Generated by Doxygen 1.8.7

Sat Oct 25 2014 11:53:15

Contents

1	quantum++ - A C++11 quantum computing library	1
2	Namespace Index	3
2.1	Namespace List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Namespace Documentation	11
6.1	qpp Namespace Reference	11
6.1.1	Typedef Documentation	18
6.1.1.1	bra	18
6.1.1.2	cmat	18
6.1.1.3	cplx	18
6.1.1.4	dmat	18
6.1.1.5	DynMat	18
6.1.1.6	ket	18
6.1.2	Function Documentation	18
6.1.2.1	absm	18
6.1.2.2	adjoint	19
6.1.2.3	anticomm	19
6.1.2.4	channel	20
6.1.2.5	channel	21
6.1.2.6	choi	22
6.1.2.7	choi2kraus	23
6.1.2.8	comm	24
6.1.2.9	compperm	25

6.1.2.10	conjugate	26
6.1.2.11	cosm	26
6.1.2.12	cwise	27
6.1.2.13	det	27
6.1.2.14	disp	28
6.1.2.15	disp	28
6.1.2.16	disp	29
6.1.2.17	disp	29
6.1.2.18	displn	29
6.1.2.19	displn	30
6.1.2.20	displn	30
6.1.2.21	displn	31
6.1.2.22	entanglement	31
6.1.2.23	evals	32
6.1.2.24	evects	33
6.1.2.25	expandout	33
6.1.2.26	expm	34
6.1.2.27	funm	35
6.1.2.28	gconcurrence	35
6.1.2.29	grams	36
6.1.2.30	grams	37
6.1.2.31	grams	37
6.1.2.32	hevals	38
6.1.2.33	hevects	38
6.1.2.34	inverse	39
6.1.2.35	invperm	39
6.1.2.36	kron	40
6.1.2.37	kron	40
6.1.2.38	kron	41
6.1.2.39	kron	41
6.1.2.40	kronpow	42
6.1.2.41	load	42
6.1.2.42	loadMATLABmatrix	43
6.1.2.43	loadMATLABmatrix	43
6.1.2.44	loadMATLABmatrix	43
6.1.2.45	logdet	44
6.1.2.46	logm	44
6.1.2.47	mket	45
6.1.2.48	mket	45
6.1.2.49	mket	46

6.1.2.50	multiidx2n	46
6.1.2.51	n2multiidx	47
6.1.2.52	norm	47
6.1.2.53	omega	48
6.1.2.54	operator""_i	48
6.1.2.55	operator""_i	48
6.1.2.56	powm	48
6.1.2.57	prj	49
6.1.2.58	ptrace	50
6.1.2.59	ptrace1	51
6.1.2.60	ptrace2	52
6.1.2.61	ptranspose	53
6.1.2.62	qmutualinfo	54
6.1.2.63	rand	55
6.1.2.64	rand	55
6.1.2.65	rand	56
6.1.2.66	rand	56
6.1.2.67	randH	57
6.1.2.68	randint	57
6.1.2.69	randket	58
6.1.2.70	randkraus	58
6.1.2.71	randn	59
6.1.2.72	randn	59
6.1.2.73	randn	60
6.1.2.74	randn	60
6.1.2.75	randperm	61
6.1.2.76	randrho	61
6.1.2.77	randU	62
6.1.2.78	randV	62
6.1.2.79	renyi	62
6.1.2.80	renyi_inf	63
6.1.2.81	reshape	64
6.1.2.82	save	64
6.1.2.83	saveMATLABmatrix	64
6.1.2.84	saveMATLABmatrix	64
6.1.2.85	saveMATLABmatrix	65
6.1.2.86	schmidtcoeff	65
6.1.2.87	schmidtprob	66
6.1.2.88	schmidtU	67
6.1.2.89	schmidtV	68

6.1.2.90	shannon	69
6.1.2.91	sinm	70
6.1.2.92	spectralpowm	71
6.1.2.93	sqrtn	71
6.1.2.94	sum	72
6.1.2.95	super	72
6.1.2.96	syspermute	73
6.1.2.97	trace	74
6.1.2.98	transpose	75
6.1.2.99	tsallis	75
6.1.3	Variable Documentation	76
6.1.3.1	chop	76
6.1.3.2	ee	76
6.1.3.3	eps	76
6.1.3.4	gt	76
6.1.3.5	maxn	76
6.1.3.6	pi	77
6.1.3.7	rdevs	77
6.1.3.8	st	77
6.2	qpp::internal Namespace Reference	77
6.2.1	Detailed Description	78
6.2.2	Function Documentation	78
6.2.2.1	_check_col_vector	78
6.2.2.2	_check_dims	78
6.2.2.3	_check_dims_match_cvect	78
6.2.2.4	_check_dims_match_mat	78
6.2.2.5	_check_dims_match_rvect	78
6.2.2.6	_check_eq_dims	78
6.2.2.7	_check_nonzero_size	78
6.2.2.8	_check_perm	78
6.2.2.9	_check_row_vector	78
6.2.2.10	_check_square_mat	78
6.2.2.11	_check_subsys_match_dims	78
6.2.2.12	_check_vector	78
6.2.2.13	_kron2	78
6.2.2.14	_multiidx2n	78
6.2.2.15	_n2multiidx	78
6.2.2.16	variadic_vector_emplace	79
6.2.2.17	variadic_vector_emplace	79

7	Class Documentation	81
7.1	qpp::DiscreteDistribution< T > Class Template Reference	81
7.1.1	Constructor & Destructor Documentation	81
7.1.1.1	DiscreteDistribution	81
7.1.1.2	DiscreteDistribution	81
7.1.1.3	DiscreteDistribution	81
7.1.2	Member Function Documentation	81
7.1.2.1	probabilities	81
7.1.2.2	sample	82
7.1.3	Member Data Documentation	82
7.1.3.1	_d	82
7.2	qpp::DiscreteDistributionAbsSquare< T > Class Template Reference	82
7.2.1	Constructor & Destructor Documentation	83
7.2.1.1	DiscreteDistributionAbsSquare	83
7.2.1.2	DiscreteDistributionAbsSquare	83
7.2.1.3	DiscreteDistributionAbsSquare	83
7.2.1.4	DiscreteDistributionAbsSquare	83
7.2.2	Member Function Documentation	83
7.2.2.1	cplx2weights	83
7.2.2.2	probabilities	83
7.2.2.3	sample	83
7.2.3	Member Data Documentation	83
7.2.3.1	_d	83
7.3	qpp::Exception Class Reference	83
7.3.1	Member Enumeration Documentation	85
7.3.1.1	Type	85
7.3.2	Constructor & Destructor Documentation	86
7.3.2.1	Exception	86
7.3.2.2	Exception	86
7.3.3	Member Function Documentation	86
7.3.3.1	_construct_exception_msg	86
7.3.3.2	what	86
7.3.4	Member Data Documentation	86
7.3.4.1	_custom	86
7.3.4.2	_msg	86
7.3.4.3	_type	86
7.3.4.4	_where	86
7.4	qpp::Gates Class Reference	86
7.4.1	Constructor & Destructor Documentation	88
7.4.1.1	Gates	88

7.4.2	Member Function Documentation	88
7.4.2.1	apply	89
7.4.2.2	applyCTRL	89
7.4.2.3	CTRL	90
7.4.2.4	Fd	90
7.4.2.5	Id	90
7.4.2.6	Rn	90
7.4.2.7	Xd	91
7.4.2.8	Zd	91
7.4.3	Friends And Related Function Documentation	91
7.4.3.1	Singleton< const Gates >	91
7.4.4	Member Data Documentation	91
7.4.4.1	CNOTab	91
7.4.4.2	CNOTba	91
7.4.4.3	CZ	91
7.4.4.4	FRED	91
7.4.4.5	H	91
7.4.4.6	Id2	91
7.4.4.7	S	91
7.4.4.8	SWAP	91
7.4.4.9	T	91
7.4.4.10	TOF	92
7.4.4.11	X	92
7.4.4.12	Y	92
7.4.4.13	Z	92
7.5	qpp::NormalDistribution< T > Class Template Reference	92
7.5.1	Constructor & Destructor Documentation	92
7.5.1.1	NormalDistribution	92
7.5.2	Member Function Documentation	92
7.5.2.1	sample	92
7.5.3	Member Data Documentation	92
7.5.3.1	_d	92
7.6	qpp::Qudit Class Reference	93
7.6.1	Constructor & Destructor Documentation	93
7.6.1.1	Qudit	93
7.6.2	Member Function Documentation	93
7.6.2.1	getD	93
7.6.2.2	getRho	93
7.6.2.3	measure	94
7.6.2.4	measure	94

7.6.3	Member Data Documentation	94
7.6.3.1	_D	94
7.6.3.2	_rho	94
7.7	qpp::RandomDevices Class Reference	95
7.7.1	Constructor & Destructor Documentation	96
7.7.1.1	RandomDevices	96
7.7.2	Friends And Related Function Documentation	96
7.7.2.1	Singleton< RandomDevices >	96
7.7.3	Member Data Documentation	96
7.7.3.1	_rd	96
7.7.3.2	_rng	96
7.8	qpp::Singleton< T > Class Template Reference	96
7.8.1	Constructor & Destructor Documentation	97
7.8.1.1	Singleton	97
7.8.1.2	~Singleton	97
7.8.1.3	Singleton	97
7.8.2	Member Function Documentation	97
7.8.2.1	get_instance	97
7.8.2.2	operator=	97
7.9	qpp::States Class Reference	97
7.9.1	Constructor & Destructor Documentation	99
7.9.1.1	States	99
7.9.2	Friends And Related Function Documentation	99
7.9.2.1	Singleton< const States >	99
7.9.3	Member Data Documentation	99
7.9.3.1	b00	99
7.9.3.2	b01	99
7.9.3.3	b10	99
7.9.3.4	b11	99
7.9.3.5	GHZ	99
7.9.3.6	pb00	99
7.9.3.7	pb01	99
7.9.3.8	pb10	99
7.9.3.9	pb11	99
7.9.3.10	pGHZ	99
7.9.3.11	pW	99
7.9.3.12	px0	99
7.9.3.13	px1	99
7.9.3.14	py0	99
7.9.3.15	py1	99

7.9.3.16	pz0	99
7.9.3.17	pz1	99
7.9.3.18	W	99
7.9.3.19	x0	99
7.9.3.20	x1	99
7.9.3.21	y0	99
7.9.3.22	y1	99
7.9.3.23	z0	100
7.9.3.24	z1	100
7.10	qpp::Timer Class Reference	100
7.10.1	Constructor & Destructor Documentation	100
7.10.1.1	Timer	100
7.10.2	Member Function Documentation	100
7.10.2.1	seconds	100
7.10.2.2	tic	100
7.10.2.3	toc	100
7.10.3	Friends And Related Function Documentation	100
7.10.3.1	operator<<	100
7.10.4	Member Data Documentation	100
7.10.4.1	_end	100
7.10.4.2	_start	100
7.11	qpp::UniformIntegerDistribution< T > Class Template Reference	101
7.11.1	Constructor & Destructor Documentation	101
7.11.1.1	UniformIntegerDistribution	101
7.11.2	Member Function Documentation	101
7.11.2.1	sample	101
7.11.3	Member Data Documentation	101
7.11.3.1	_d	101
7.12	qpp::UniformRealDistribution< T > Class Template Reference	101
7.12.1	Constructor & Destructor Documentation	102
7.12.1.1	UniformRealDistribution	102
7.12.2	Member Function Documentation	102
7.12.2.1	sample	102
7.12.3	Member Data Documentation	102
7.12.3.1	_d	102
8	File Documentation	103
8.1	include/channels.h File Reference	103
8.2	include/classes/exception.h File Reference	104
8.3	include/classes/gates.h File Reference	104

8.4	include/classes/qudit.h File Reference	105
8.5	include/classes/randevs.h File Reference	105
8.6	include/classes/singleton.h File Reference	106
8.6.1	Macro Definition Documentation	106
8.6.1.1	CLASS_CONST_SINGLETON	106
8.6.1.2	CLASS_SINGLETON	106
8.7	include/classes/stat.h File Reference	107
8.8	include/classes/states.h File Reference	107
8.9	include/classes/timer.h File Reference	108
8.10	include/constants.h File Reference	108
8.11	include/entanglement.h File Reference	109
8.12	include/entropies.h File Reference	110
8.13	include/functions.h File Reference	111
8.14	include/internal.h File Reference	115
8.15	include/io.h File Reference	116
8.16	include/matlab.h File Reference	117
8.17	include/qpp.h File Reference	118
8.18	include/random.h File Reference	119
8.19	include/types.h File Reference	120
	Index	122

Chapter 1

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

Version

0.1

Author

Vlad Gheorghiu, vgheorgh@gmail.com

Date

24 October 2014

This is the main page of the documentation. More coming soon.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

qpp	11
qpp::internal	77

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

qpp::DiscreteDistribution< T >	81
qpp::DiscreteDistributionAbsSquare< T >	82
exception	
qpp::Exception	83
qpp::NormalDistribution< T >	92
qpp::Qudit	93
qpp::Singleton< T >	96
qpp::Gates	86
qpp::RandomDevices	95
qpp::Singleton< const Gates >	96
qpp::Singleton< const States >	96
qpp::States	97
qpp::Singleton< RandomDevices >	96
qpp::Timer	100
qpp::UniformIntegerDistribution< T >	101
qpp::UniformRealDistribution< T >	101

Chapter 4

Class Index

4.1 Class List

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

qpp::DiscreteDistribution< T >	81
qpp::DiscreteDistributionAbsSquare< T >	82
qpp::Exception	83
qpp::Gates	86
qpp::NormalDistribution< T >	92
qpp::Qudit	93
qpp::RandomDevices	95
qpp::Singleton< T >	96
qpp::States	97
qpp::Timer	100
qpp::UniformIntegerDistribution< T >	101
qpp::UniformRealDistribution< T >	101

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

include/channels.h	103
include/constants.h	108
include/entanglement.h	109
include/entropies.h	110
include/functions.h	111
include/internal.h	115
include/io.h	116
include/matlab.h	117
include/qpp.h	118
include/random.h	119
include/types.h	120
include/classes/exception.h	104
include/classes/gates.h	104
include/classes/qudit.h	105
include/classes/randevs.h	105
include/classes/singleton.h	106
include/classes/stat.h	107
include/classes/states.h	107
include/classes/timer.h	108

Chapter 6

Namespace Documentation

6.1 qpp Namespace Reference

Namespaces

- [internal](#)

Classes

- class [DiscreteDistribution](#)
- class [DiscreteDistributionAbsSquare](#)
- class [Exception](#)
- class [Gates](#)
- class [NormalDistribution](#)
- class [Qudit](#)
- class [RandomDevices](#)
- class [Singleton](#)
- class [States](#)
- class [Timer](#)
- class [UniformIntegerDistribution](#)
- class [UniformRealDistribution](#)

Typedefs

- using [cplx](#) = std::complex< double >
Complex number in double precision.
- using [cmat](#) = Eigen::MatrixXcd
Complex (double precision) dynamic Eigen matrix.
- using [dmat](#) = Eigen::MatrixXd
Real (double precision) dynamic Eigen matrix.
- using [ket](#) = Eigen::Matrix< [cplx](#), Eigen::Dynamic, 1 >
Complex (double precision) dynamic Eigen column matrix.
- using [bra](#) = Eigen::Matrix< [cplx](#), 1, Eigen::Dynamic >
Complex (double precision) dynamic Eigen row matrix.
- template<typename Scalar >
using [DynMat](#) = Eigen::Matrix< Scalar, Eigen::Dynamic, Eigen::Dynamic >
Dynamic Eigen matrix over the field specified by Scalar.

Functions

- `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 >`
`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.
- `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 >`
`cmat 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 >`
`cmat 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 shannon` (const Eigen::MatrixBase< Derived > &A)
Shannon/von-Neumann entropy of the probability distribution/density matrix A.
- `template<typename Derived >`
`double renyi` (const double alpha, const Eigen::MatrixBase< Derived > &A)
Renyi- α entropy of the probability distribution/density matrix A, for $\alpha \geq 0$.
- `template<typename Derived >`
`double renyi_inf` (const Eigen::MatrixBase< Derived > &A)
Renyi- ∞ entropy (min entropy) of the probability distribution/density matrix A.
- `template<typename Derived >`
`double tsallis` (const double alpha, const Eigen::MatrixBase< Derived > &A)
Tsallis- α entropy of the probability distribution/density matrix A, for $\alpha \geq 0$

- `template<typename Derived >`
`double qmutualinfo (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &subsysA,`
`const std::vector< std::size_t > &subsysB, const std::vector< std::size_t > &dims)`
Quantum mutual information between 2 subsystems of a composite system.
- `template<typename Derived >`
`DynMat< typename Derived::Scalar > transpose (const Eigen::MatrixBase< Derived > &A)`
Transpose.
- `template<typename Derived >`
`DynMat< typename Derived::Scalar > conjugate (const Eigen::MatrixBase< Derived > &A)`
Complex conjugate.
- `template<typename Derived >`
`DynMat< typename Derived::Scalar > adjoint (const Eigen::MatrixBase< Derived > &A)`
Adjoint.
- `template<typename Derived >`
`DynMat< typename Derived::Scalar > inverse (const Eigen::MatrixBase< Derived > &A)`
Inverse.
- `template<typename Derived >`
`Derived::Scalar trace (const Eigen::MatrixBase< Derived > &A)`
Trace.
- `template<typename Derived >`
`Derived::Scalar det (const Eigen::MatrixBase< Derived > &A)`
Determinant.
- `template<typename Derived >`
`Derived::Scalar logdet (const Eigen::MatrixBase< Derived > &A)`
Logarithm of the determinant.
- `template<typename Derived >`
`Derived::Scalar sum (const Eigen::MatrixBase< Derived > &A)`
Element-wise sum.
- `template<typename Derived >`
`double norm (const Eigen::MatrixBase< Derived > &A)`
Trace norm.
- `template<typename Derived >`
`cmat evals (const Eigen::MatrixBase< Derived > &A)`
Eigenvalues.
- `template<typename Derived >`
`cmat evecs (const Eigen::MatrixBase< Derived > &A)`
Eigenvectors.
- `template<typename Derived >`
`dmat hevals (const Eigen::MatrixBase< Derived > &A)`
Hermitian eigenvalues.
- `template<typename Derived >`
`cmat hevecs (const Eigen::MatrixBase< Derived > &A)`
Hermitian eigenvectors.
- `template<typename Derived >`
`cmat funm (const Eigen::MatrixBase< Derived > &A, cplx(*f)(const cplx &))`
Functional calculus $f(A)$
- `template<typename Derived >`
`cmat sqrtm (const Eigen::MatrixBase< Derived > &A)`
Matrix square root.
- `template<typename Derived >`
`cmat absm (const Eigen::MatrixBase< Derived > &A)`
Matrix absolute value.

- `template<typename Derived >`
`cmat expm` (const Eigen::MatrixBase< Derived > &A)
Matrix exponential.
- `template<typename Derived >`
`cmat logm` (const Eigen::MatrixBase< Derived > &A)
Matrix logarithm.
- `template<typename Derived >`
`cmat sinm` (const Eigen::MatrixBase< Derived > &A)
Matrix sin.
- `template<typename Derived >`
`cmat cosm` (const Eigen::MatrixBase< Derived > &A)
Matrix cos.
- `template<typename Derived >`
`cmat spectralpowm` (const Eigen::MatrixBase< Derived > &A, const `cplx` z)
Matrix power.
- `template<typename Derived >`
`DynMat< typename Derived::Scalar > powm` (const Eigen::MatrixBase< Derived > &A, std::size_t n)
Matrix power.
- `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 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 >`
`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 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 > expandout` (const Eigen::MatrixBase< Derived > &A, std::size_t pos, const std::vector< std::size_t > &dims)
Expand out.
- `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)
Multi-partite qubit ket.
- `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)
Multi-partite qudit ket (same dimensions overload)
- `std::vector< std::size_t > invperm` (const std::vector< std::size_t > &perm)
Inverse permutation.
- `std::vector< std::size_t > compperm` (const std::vector< std::size_t > &perm, const std::vector< std::size_t > &sigma)
Compose permutations.
- `template<typename T >`
`void 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 >`
`void 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 >`
`void 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 >`
`void 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 >`
`void 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 >`
`void 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.

- `void 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.
- `void displn (const cplx z, double chop=qpp::chop, std::ostream &os=std::cout)`
Displays a number (implicitly converted to `std::complex<double>`) in friendly form. Adds a new line.

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

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

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

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

- `template<typename Derived >`
`Derived loadMATLABmatrix (const std::string &mat_file, const std::string &var_name)`

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

- `template<>`
`dmat loadMATLABmatrix (const std::string &mat_file, const std::string &var_name)`
Loads an Eigen dynamic matrix from a MATLAB .mat file, specialization for double matrices (`qpp::dmat`)

- `template<>`
`cmat loadMATLABmatrix (const std::string &mat_file, const std::string &var_name)`
Loads an Eigen dynamic matrix from a MATLAB .mat file, specialization for complex matrices (`qpp::cmat`)

- `template<typename Derived >`
`void saveMATLABmatrix (const Eigen::MatrixBase< Derived > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)`

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

- `template<>`
`void saveMATLABmatrix (const Eigen::MatrixBase< dmat > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)`

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

- `template<>`
`void saveMATLABmatrix (const Eigen::MatrixBase< cmat > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)`

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

- `template<typename 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 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 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 (double) 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(mean, sigma)
- `template<>`
`dmat randn` (`std::size_t rows`, `std::size_t cols`, `double mean`, `double sigma`)
Generates a random matrix with entries normally distributed in N(mean, sigma), specialization for double matrices ([qpp::dmat](#))
- `template<>`
`cmat randn` (`std::size_t rows`, `std::size_t cols`, `double mean`, `double sigma`)
Generates a random matrix with entries (both real and imaginary) normally distributed in N(mean, sigma), specialization for complex matrices ([qpp::cmat](#))
- `double randn` (`double mean=0`, `double sigma=1`)
Generates a random real number (double) normally distributed in N(mean, sigma)
- `cmat randU` (`std::size_t D`)
Generates a random unitary matrix.
- `cmat randV` (`std::size_t Din`, `std::size_t Dout`)
Generates a random isometry matrix.
- `std::vector< cmat > randkraus` (`std::size_t n`, `std::size_t D`)
Generates a set of random Kraus operators.
- `cmat randH` (`std::size_t D`)
Generates a random Hermitian matrix.
- `ket randket` (`std::size_t D`)
Generates a random normalized ket (pure state vector)
- `cmat randrho` (`std::size_t D`)
Generates a random density matrix.
- `std::vector< std::size_t > randperm` (`std::size_t n`)
Generates a random uniformly distributed permutation.

Variables

- `constexpr double chop` = 1e-10
Used in [qpp::disp\(\)](#) and [qpp::displn\(\)](#) for setting to zero numbers that have their absolute value smaller than [qpp::ct←→::chop](#).
- `constexpr double eps` = 1e-12
Used to decide whether a number or expression in double precision is zero or not.
- `constexpr std::size_t maxn` = 64
Maximum number of qubits.
- `constexpr double pi` = 3.141592653589793238462643383279502884
 π
- `constexpr double ee` = 2.718281828459045235360287471352662497
Base of natural logarithm, e.
- `RandomDevices & rdevs` = `RandomDevices::get_instance()`
[qpp::RandomDevices Singleton](#)
- `const Gates & gt` = `Gates::get_instance()`

- `qpp::Gates` *const Singleton*
- `const States & st = States::get_instance()`
- `qpp::States` *const Singleton*

6.1.1 Typedef Documentation

6.1.1.1 using qpp::bra = typedef Eigen::Matrix<cplx, 1, Eigen::Dynamic>

Complex (double precision) dynamic Eigen row matrix.

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

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 Eigen::MatrixXd

Real (double precision) dynamic Eigen matrix.

6.1.1.5 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>
```

6.1.1.6 using qpp::ket = typedef Eigen::Matrix<cplx, Eigen::Dynamic, 1>

Complex (double precision) dynamic Eigen column matrix.

6.1.2 Function Documentation

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

Matrix absolut value.

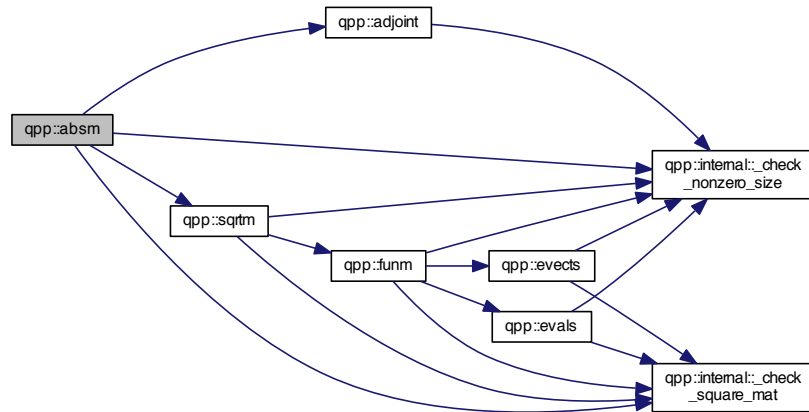
Parameters

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

Returns

Matrix absolut value of A , as a dynamic matrix over the complex field

Here is the call graph for this function:



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

Adjoint.

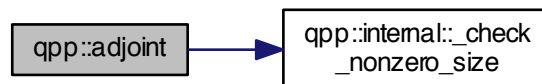
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.3 `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

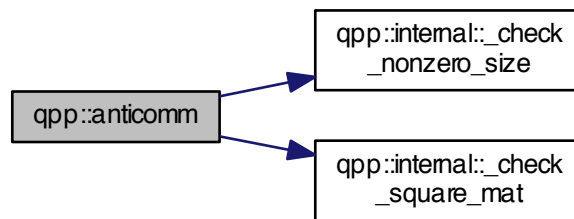
Parameters

A	Eigen expression
B	Eigen expression

Returns

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

Here is the call graph for this function:



6.1.2.4 `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 ρ .

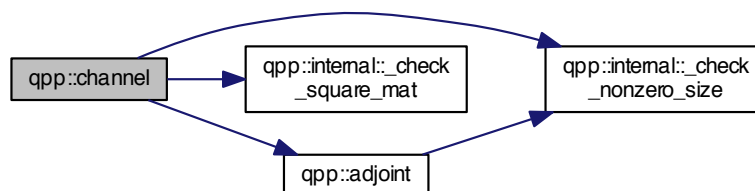
Parameters

ρ	Eigen expression
Ks	<code>std::vector</code> of Eigen expressions representing the set of Kraus operators

Returns

Output density matrix, as a dynamic matrix over the complex field

Here is the call graph for this function:



6.1.2.5 `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*.

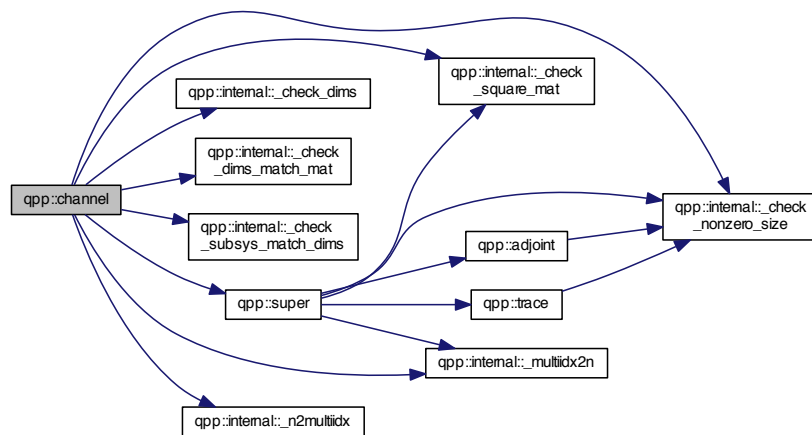
Parameters

<i>rho</i>	Eigen expression
<i>Ks</i>	std::vector of Eigen expressions representing the set of Kraus operators
<i>subsys</i>	Subsystems' indexes
<i>dims</i>	Dimensions of the multi-partite system

Returns

Output density matrix, as a dynamic matrix over the complex field

Here is the call graph for this function:



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

Note

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

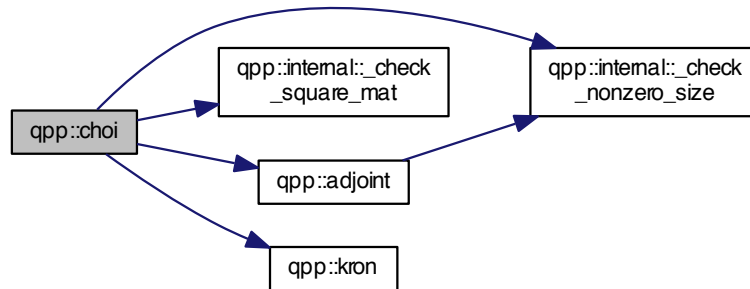
Parameters

<i>Ks</i>	std::vector of Eigen expressions representing the set of Kraus operators
-----------	--

Returns

Choi matrix representation, as a dynamic matrix over the complex field

Here is the call graph for this function:



6.1.2.7 `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 $i \neq j$

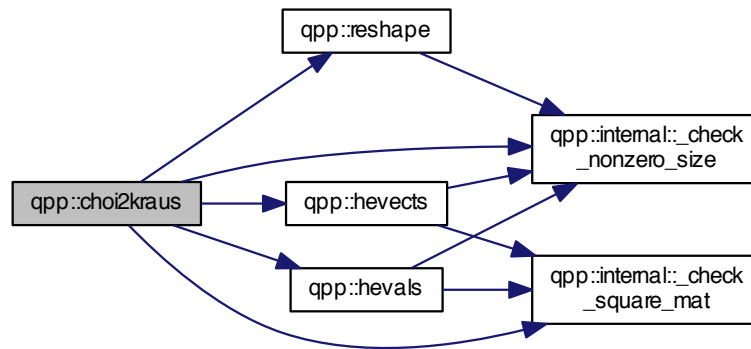
Parameters

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

Returns

std::vector of dynamic matrices over the complex field representing the set of Kraus operators

Here is the call graph for this function:



6.1.2.8 `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

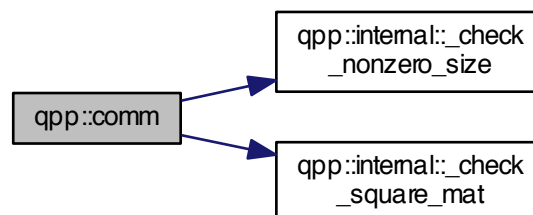
Parameters

A	Eigen expression
B	Eigen expression

Returns

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

Here is the call graph for this function:



6.1.2.9 `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:



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

Complex conjugate.

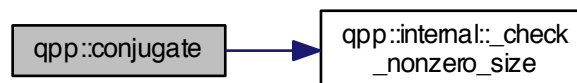
Parameters

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

Returns

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

Here is the call graph for this function:



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

Matrix cos.

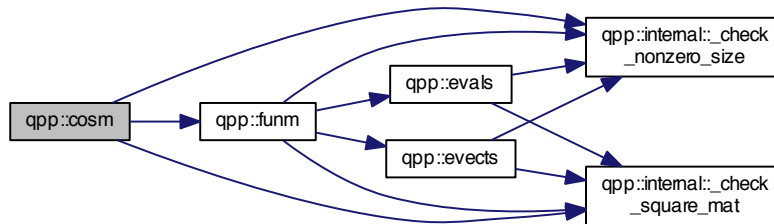
Parameters

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

Returns

Matrix cosine of A , as a dynamic matrix over the complex field

Here is the call graph for this function:



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

Functor.

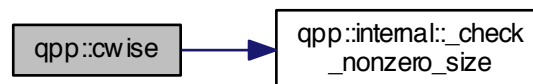
Parameters

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:



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

Determinant.

Parameters

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

Returns

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

Here is the call graph for this function:



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

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

See also

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

Parameters

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

6.1.2.15 `template<typename T> void qpp::disp (const T * x, const std::size_t n, const std::string & separator, const std::string & start = " [", const std::string & end = "] ", std::ostream & os = std::cout)`

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

See also

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

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

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

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

See also

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

Parameters

<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

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

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

See also

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

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

Here is the call graph for this function:



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

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

See also

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

Parameters

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

Here is the call graph for this function:



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

Displays a C-style array. Adds a newline.

See also

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

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

Here is the call graph for this function:



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

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

See also

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

Parameters

<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

Here is the call graph for this function:



6.1.2.21 `void qpp::dispIn (const cplx z, double chop = qpp::chop, std::ostream & os = std::cout)`

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

See also

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

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

Here is the call graph for this function:



6.1.2.22 `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\(\)](#)

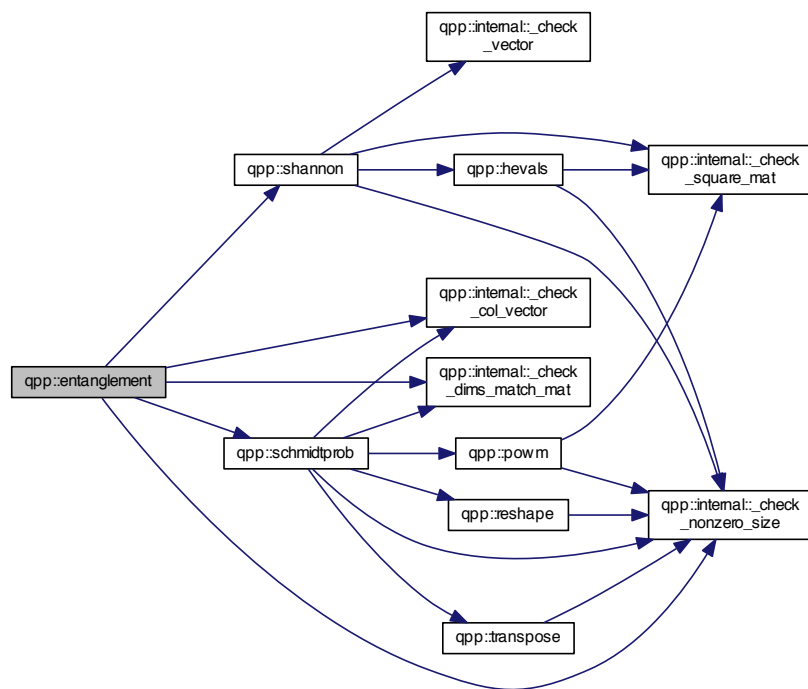
Parameters

<i>A</i>	Eigen expression
<i>dims</i>	Subsystems' dimensions

Returns

Entanglement, with the logarithm in base 2

Here is the call graph for this function:



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

Eigenvalues.

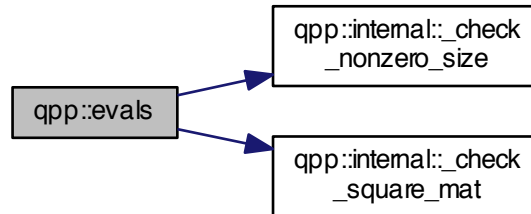
Parameters

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

Returns

Eigenvalues of A , as a diagonal dynamic matrix over the complex field, with the eigenvalues on the diagonal

Here is the call graph for this function:



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

Eigenvectors.

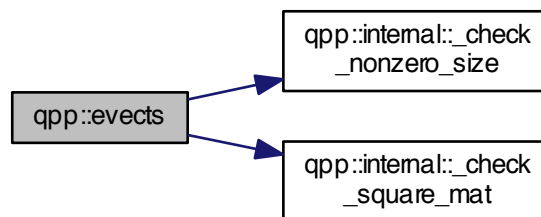
Parameters

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

Returns

Eigenvectors of A , as columns of a dynamic matrix over the complex field

Here is the call graph for this function:



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

Expand out.

Expand out A as a matrix in a multi-partite system

Faster than using `qpp::kron(I, I, ..., I, A, I, ..., I)`

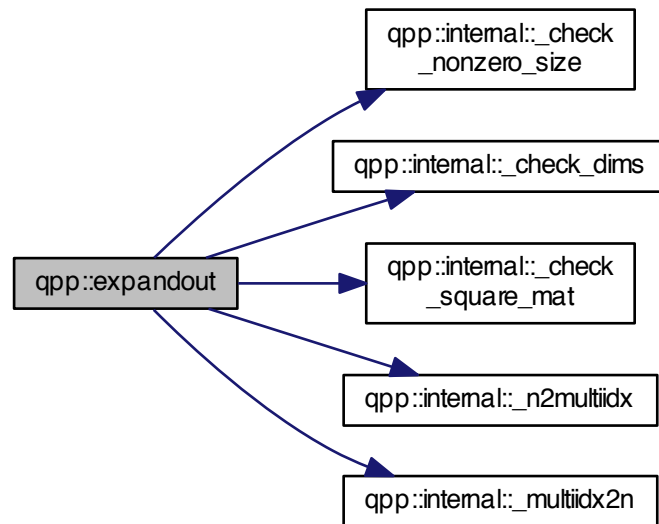
Parameters

<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

Here is the call graph for this function:



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

Matrix exponential.

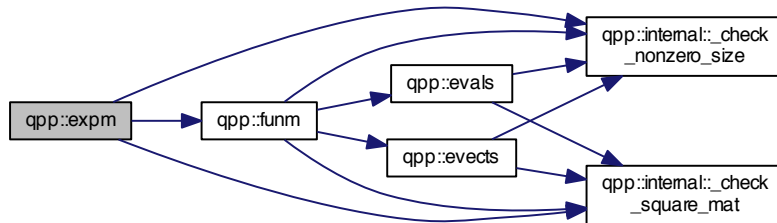
Parameters

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

Returns

Matrix exponential of A , as a dynamic matrix over the complex field

Here is the call graph for this function:



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

Functional calculus $f(A)$

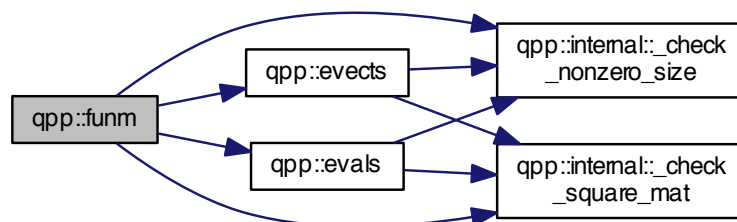
Parameters

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

Returns

$f(A)$, as a dynamic matrix over the complex field

Here is the call graph for this function:



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

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

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

See also

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

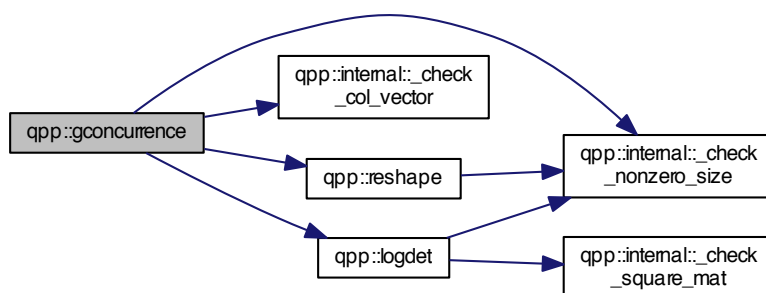
Parameters

<i>A</i>	Eigen expression
<i>dims</i>	Subsystems' dimensions

Returns

G-concurrence

Here is the call graph for this function:



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

Gram-Schmidt orthogonalization (std::vector overload)

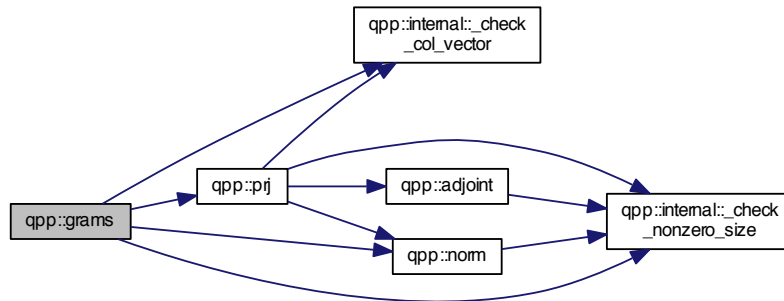
Parameters

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

Returns

Gram-Schmidt vectors of V s as columns of a dynamic matrix over the same scalar field

Here is the call graph for this function:



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

Gram-Schmidt orthogonalization (std::initializer_list overload)

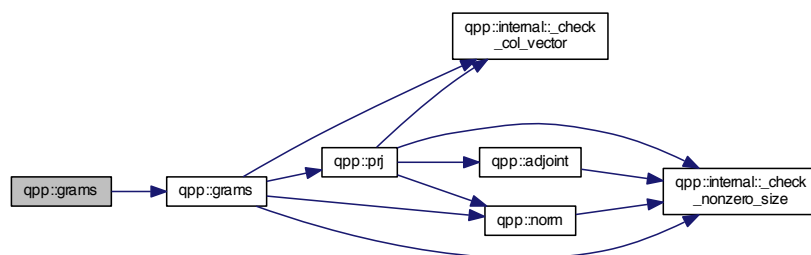
Parameters

V s	std::initializer_list of Eigen expressions as column vectors
-------	--

Returns

Gram-Schmidt vectors of V s as columns of a dynamic matrix over the same scalar field

Here is the call graph for this function:



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

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

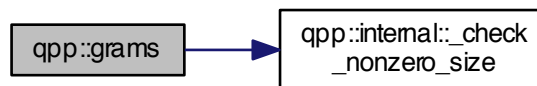
Parameters

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

Returns

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

Here is the call graph for this function:



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

Hermitian eigenvalues.

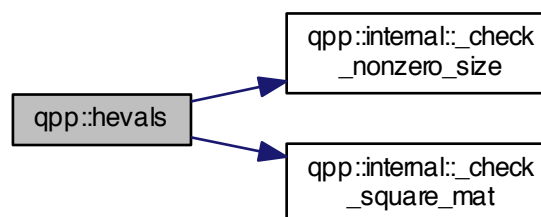
Parameters

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

Returns

Eigenvalues of Hermitian A , as a diagonal dynamic matrix over the real field, with eigenvalues on the diagonal

Here is the call graph for this function:



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

Hermitian eigenvectors.

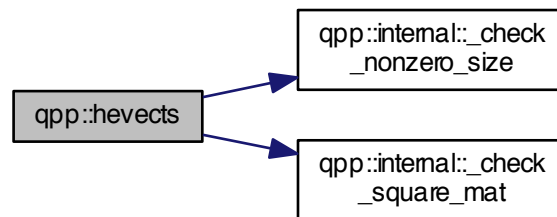
Parameters

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

Returns

Eigenvectors of Hermitian A , as columns of a dynamic matrix over the complex field

Here is the call graph for this function:



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

Inverse.

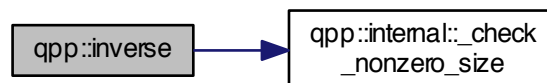
Parameters

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

Returns

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

Here is the call graph for this function:



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

Inverse permutation.

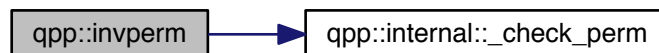
Parameters

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

Returns

Inverse of the permutation *perm*

Here is the call graph for this function:



6.1.2.36 `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*

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

Kronecker product (variadic overload)

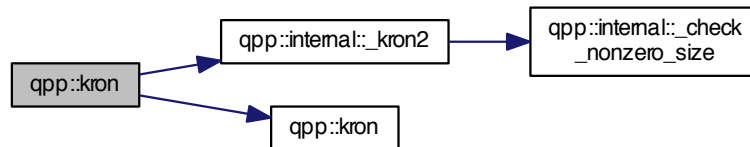
Parameters

<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

Here is the call graph for this function:



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

Kronecker product (std::vector overload)

Parameters

As	std::vector of Eigen expressions
-----------	----------------------------------

Returns

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

Here is the call graph for this function:



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

Kronecker product (std::initializer_list overload)

Parameters

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

Returns

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

Here is the call graph for this function:



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

Kronecker power.

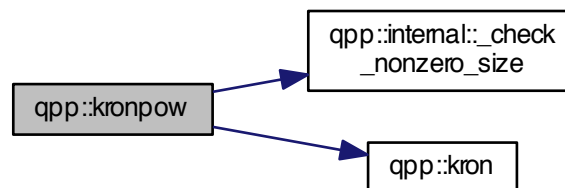
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.41 `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

6.1.2.42 `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)

6.1.2.43 `template<> dmat qpp::loadMATLABmatrix (const std::string & mat_file, const std::string & var_name)`

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

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

Example:

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

Note

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

Parameters

<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](#))

6.1.2.44 `template<> cmat qpp::loadMATLABmatrix (const std::string & mat_file, const std::string & var_name)`

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

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

Example:

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

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](#))

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

Logarithm of the determinant.

Especially useful when the determinant overflows/underflows

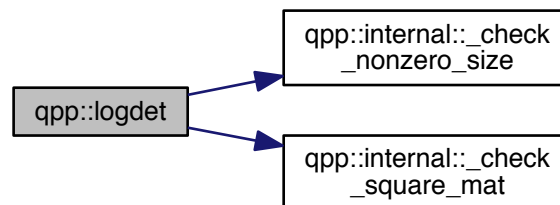
Parameters

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

Returns

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

Here is the call graph for this function:



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

Matrix logarithm.

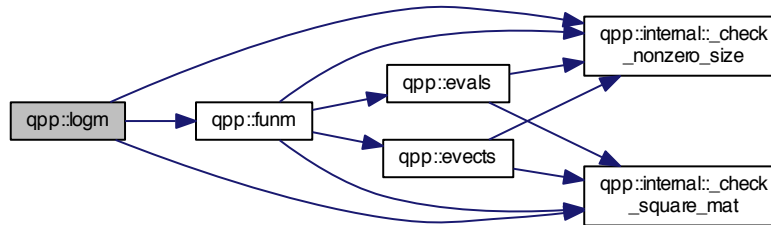
Parameters

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

Returns

Matrix logarithm of A , as a dynamic matrix over the complex field

Here is the call graph for this function:



6.1.2.47 ket qpp::mket (const std::vector< std::size_t > & mask)

Multi-partite qubit ket.

Constructs the multi-partite qubit ket $|\text{mask}\rangle$, where *mask* is a std::vector of 0's and 1's

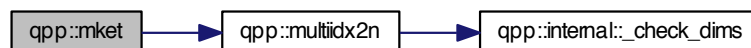
Parameters

<i>mask</i>	std::vector of 0's and 1's
-------------	----------------------------

Returns

Multi-partite qubit state vector, as a dynamic column vector over the complex field

Here is the call graph for this function:



6.1.2.48 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*

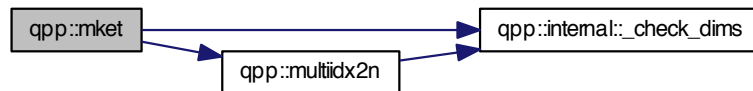
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.49 ket qpp::mket (const std::vector< std::size_t > & mask, std::size_t d)

Multi-partite qudit ket (same dimensions overload)

Constructs the multi-partite qudit ket $|\text{mask}\rangle$ in a multi-partite system, 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 .

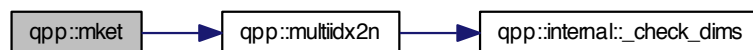
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.50 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:



6.1.2.51 `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:



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

Trace norm.

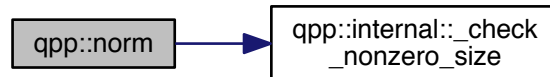
Parameters

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

Returns

Trace norm (Frobenius norm) of A , as a real number

Here is the call graph for this function:



6.1.2.53 `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 $\exp(2\pi i/D)$

6.1.2.54 `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>
```

6.1.2.55 `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>
```

6.1.2.56 `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 $A^0 = I$

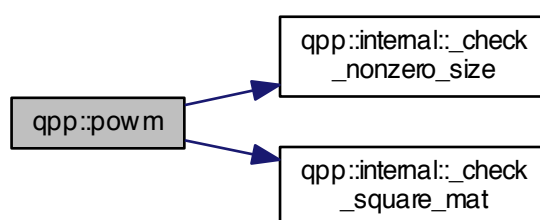
Parameters

A	Eigen expression
n	Non-negative integer

Returns

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

Here is the call graph for this function:



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

Projector.

Normalized projector onto state vector

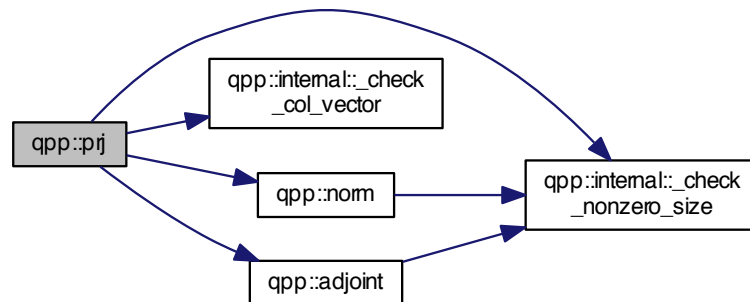
Parameters

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

Here is the call graph for this function:



6.1.2.58 `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

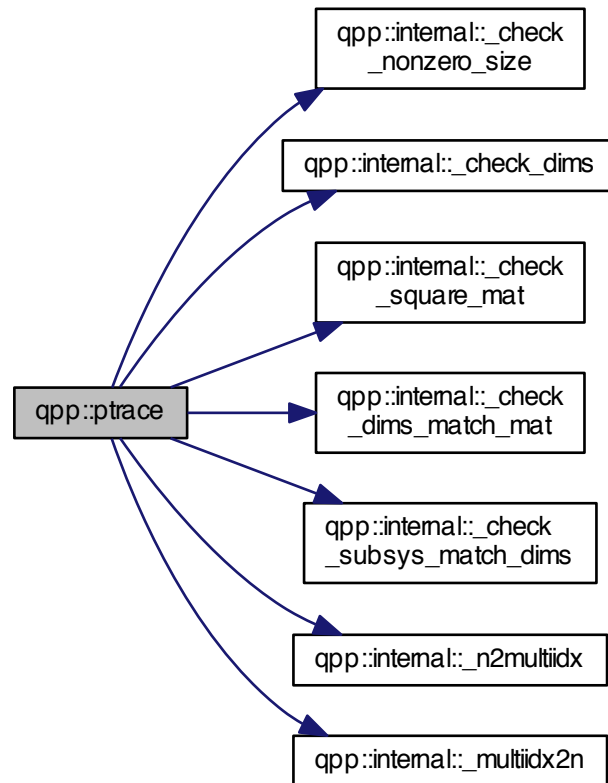
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.59 `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

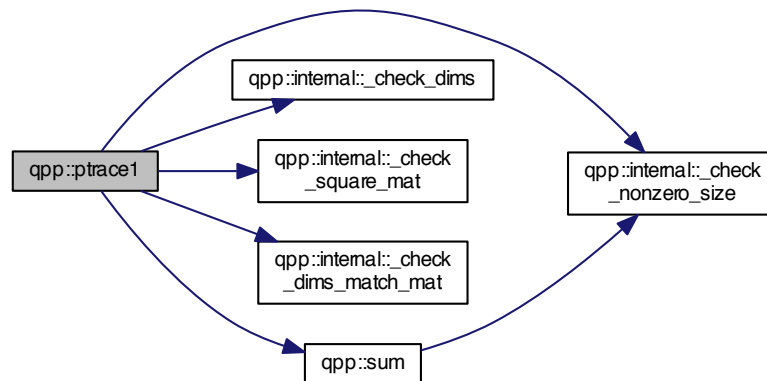
Parameters

<i>A</i>	Eigen expression
<i>dims</i>	Dimensions of 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

Here is the call graph for this function:



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

Partial trace.

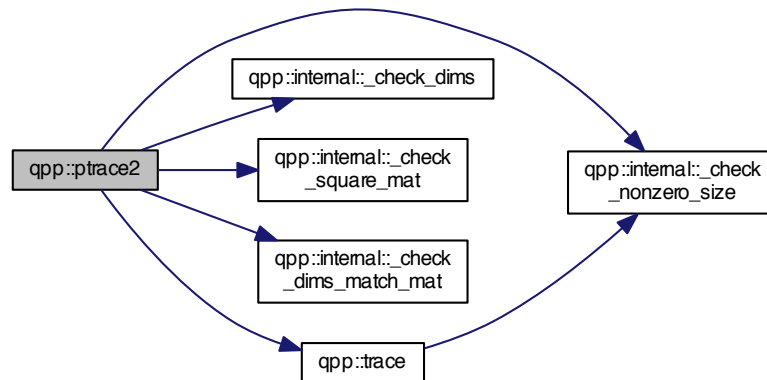
Parameters

A	Eigen expression
$dims$	Dimensions of 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

Here is the call graph for this function:



6.1.2.61 `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

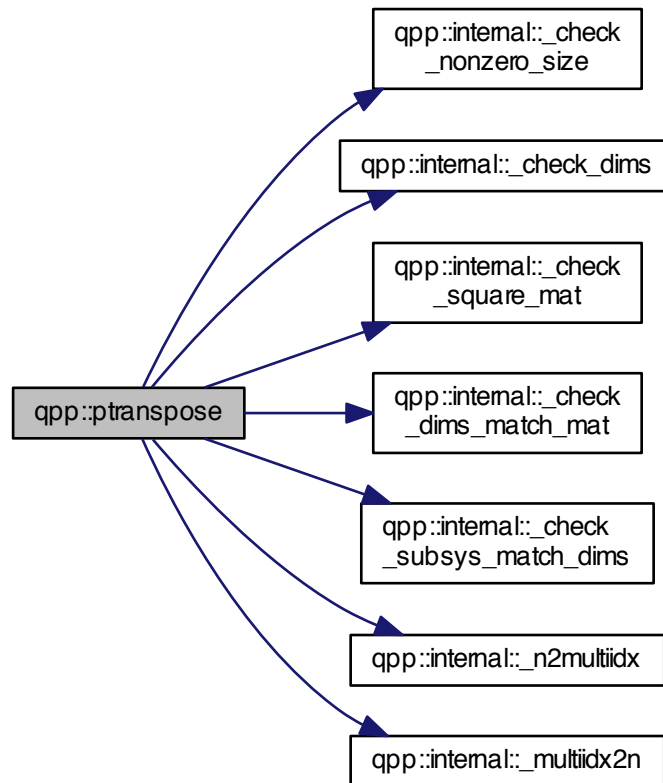
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.62 `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.

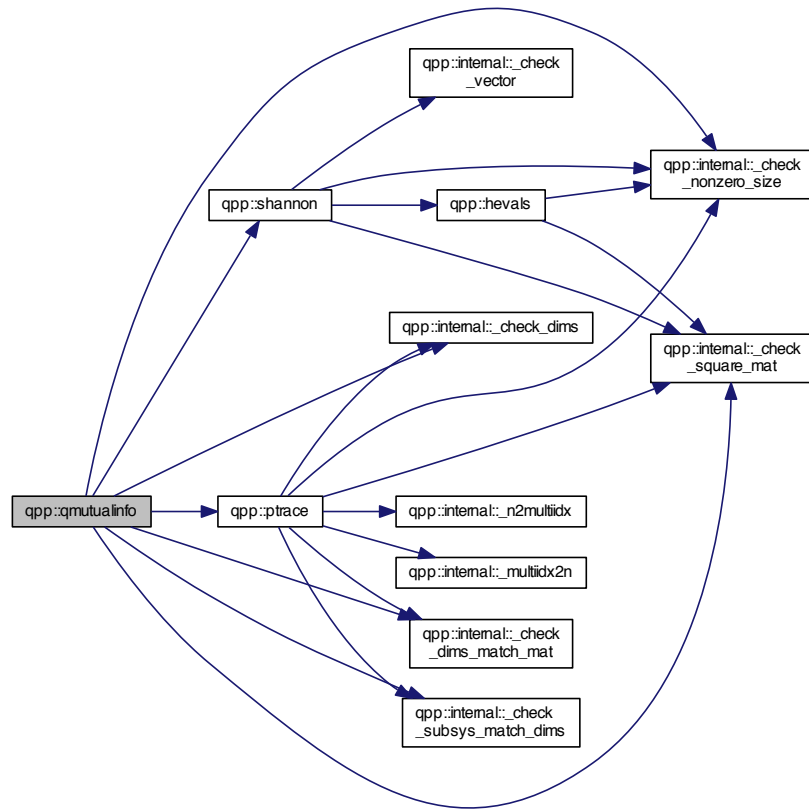
Parameters

<i>A</i>	Eigen expression
<i>subsysA</i>	Indexes of the first subsystem
<i>subsysB</i>	Indexes of the second subsystem
<i>dims</i>	Subsystems' dimensions

Returns

Mutual information between the 2 subsystems

Here is the call graph for this function:



6.1.2.63 `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`

6.1.2.64 `template<> dmat qpp::rand (std::size_t rows, std::size_t cols, double a, double b)`

Generates a random 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 dynamic double matrix, 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 double dynamic matrix ([qpp::dmat](#))

6.1.2.65 `template<> cmat qpp::rand (std::size_t rows, std::size_t cols, double a, double b)`

Generates a random 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 dynamic complex matrix, 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 dynamic matrix ([qpp::cmat](#))

Here is the call graph for this function:



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

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

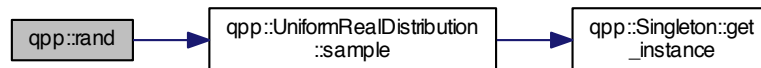
Parameters

<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:

**6.1.2.67 cmat qpp::randH (std::size_t *D*)**

Generates a random Hermitian matrix.

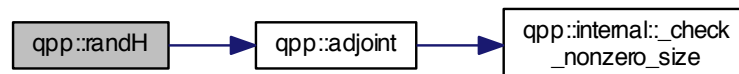
Parameters

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

Returns

Random Hermitian dynamic matrix

Here is the call graph for this function:

**6.1.2.68 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].

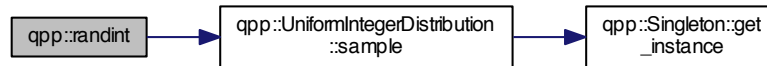
Parameters

<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:



6.1.2.69 ket qpp::randket (std::size_t D)

Generates a random normalized ket (pure state vector)

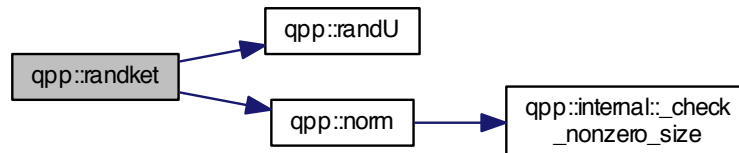
Parameters

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

Returns

Random normalized ket dynamic column-matrix

Here is the call graph for this function:



6.1.2.70 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 $\sum_i K_i^\dagger K_i = I$

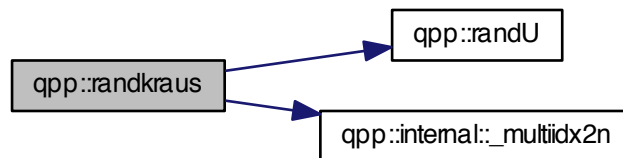
Parameters

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:



6.1.2.71 `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`

6.1.2.72 `template<> dmat qpp::randn (std::size_t rows, std::size_t cols, double mean, double sigma)`

Generates a random 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 dynamic double matrix, with entries normally distributed in N(0,2)
auto mat = randn<dmat>(3, 3, 0, 2);
```

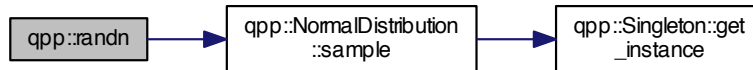
Parameters

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

Here is the call graph for this function:



6.1.2.73 `template<> cmat qpp::randn (std::size_t rows, std::size_t cols, double mean, double sigma)`

Generates a random 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 dynamic complex matrix, 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 dynamic matrix ([qpp::cmat](#))

Here is the call graph for this function:



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

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

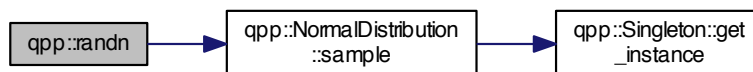
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.75 `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:



6.1.2.76 `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, as a complex dynamic matrix

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

Generates a random unitary matrix.

Parameters

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

Returns

Random unitary dynamic matrix

6.1.2.78 `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 dynamic matrix

Here is the call graph for this function:

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

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

Parameters

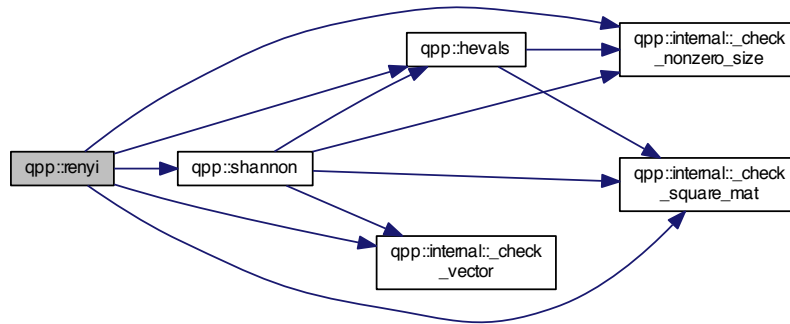
<i>alpha</i>	Non-negative real number
--------------	--------------------------

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

Returns

Renyi- α entropy, with the logarithm in base 2

Here is the call graph for this function:



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

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

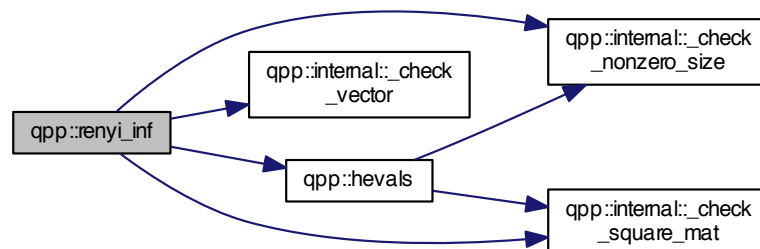
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.81 `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)

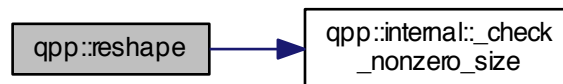
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.82 `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

6.1.2.83 `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)

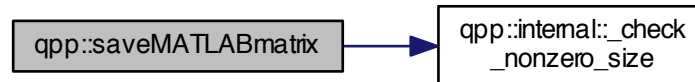
6.1.2.84 `template<> void qpp::saveMATLABmatrix (const Eigen::MatrixBase<dmat > & A, const std::string & mat_file, const std::string & var_name, const std::string & mode)`

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

Parameters

<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:



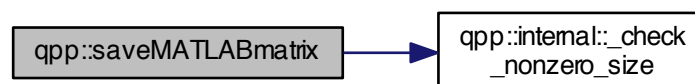
6.1.2.85 `template<> void qpp::saveMATLABmatrix (const Eigen::MatrixBase< cmat > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)`

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

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:



6.1.2.86 `template<typename Derived > cmat 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\(\)](#)

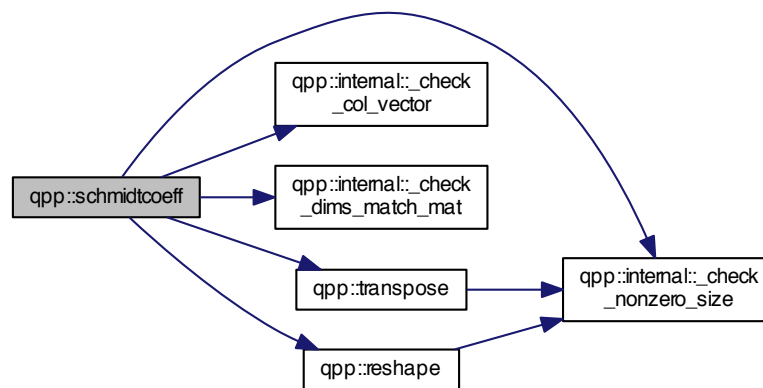
Parameters

<i>A</i>	Eigen expression
<i>dims</i>	Subsystems' dimensions

Returns

Schmidt coefficients of A , as a dynamic matrix over the complex field, with the Schmidt coefficients on the diagonal

Here is the call graph for this function:



6.1.2.87 `template<typename Derived> cmat 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\(\)`](#)

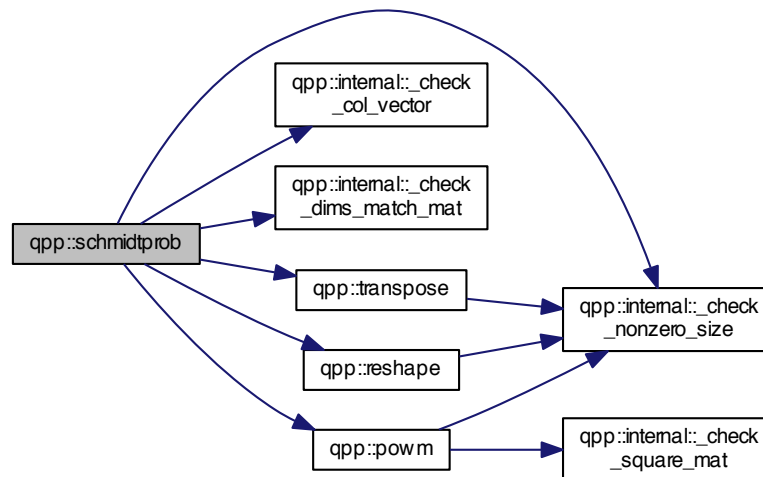
Parameters

<i>A</i>	Eigen expression
<i>dims</i>	Subsystems' dimensions

Returns

Schmidt probabilities of A , as a dynamic matrix over the complex field, with the Schmidt probabilities on the diagonal

Here is the call graph for this function:



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

Schmidt basis on Alice's side.

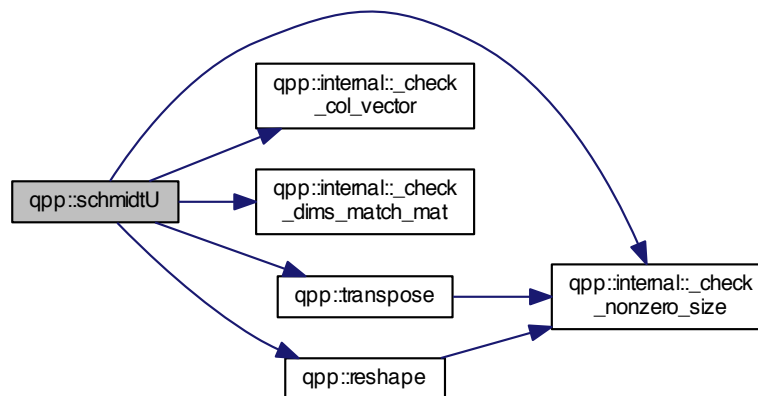
Parameters

A	Eigen expression
$dims$	Subsystems' dimensions

Returns

Unitary matrix U representing the Schmidt basis on Alice's side, as a dynamic matrix over the complex field, acting on the computational basis as $U|j\rangle = |\bar{j}\rangle$ (Schmidt vector)

Here is the call graph for this function:



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

Schmidt basis on Bob's side.

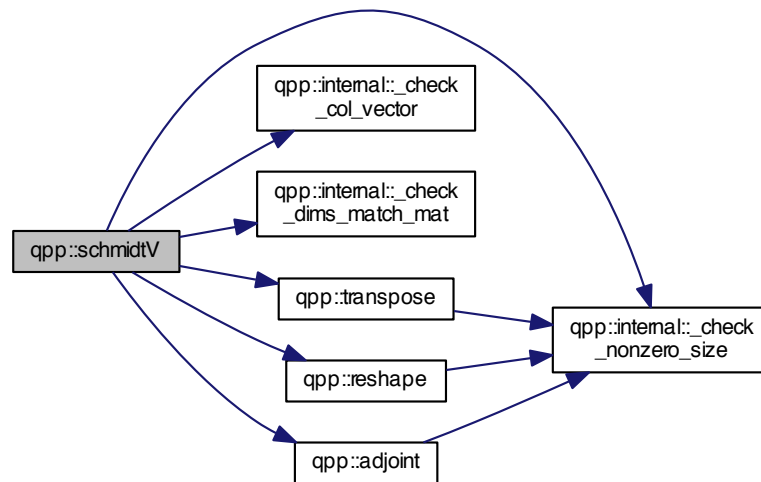
Parameters

<i>A</i>	Eigen expression
<i>dims</i>	Subsystems' dimensions

Returns

Unitary matrix V representing the Schmidt basis on Bob's side, as a dynamic matrix over the complex field, acting on the computational basis as $V|j\rangle = |\bar{j}\rangle$ (Schmidt vector)

Here is the call graph for this function:



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

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

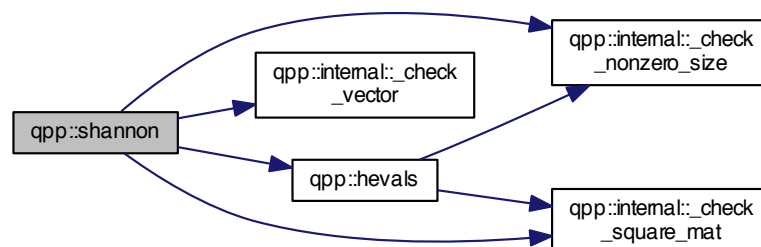
Parameters

<i>A</i>	Eigen expression, representing a probability distribution (dynamic column vector) or a density matrix (dynamic matrix over the complex field)
----------	---

Returns

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

Here is the call graph for this function:



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

Matrix sin.

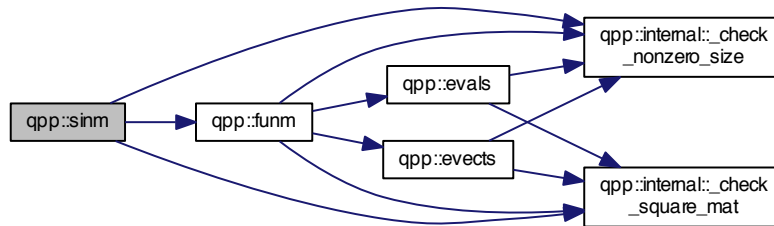
Parameters

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

Returns

Matrix sine of A , as a dynamic matrix over the complex field

Here is the call graph for this function:



6.1.2.92 `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 $A^0 = I$

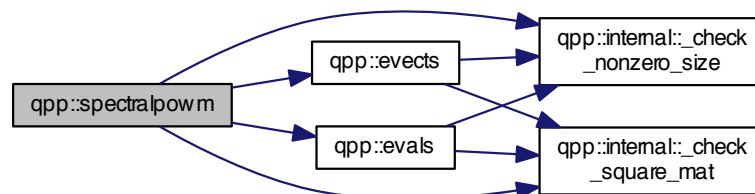
Parameters

A	Eigen expression
z	Complex number

Returns

Matrix power A^z , as a dynamic matrix over the complex field

Here is the call graph for this function:



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

Matrix square root.

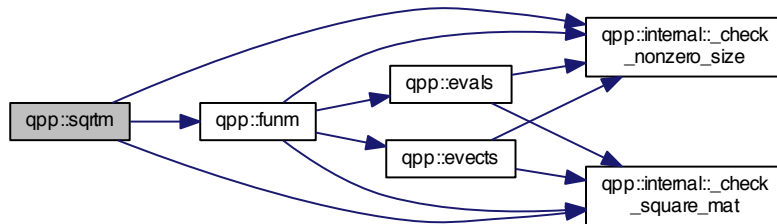
Parameters

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

Returns

Matrix square root of A , as a dynamic matrix over the complex field

Here is the call graph for this function:



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

Element-wise sum.

Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.95 `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 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.

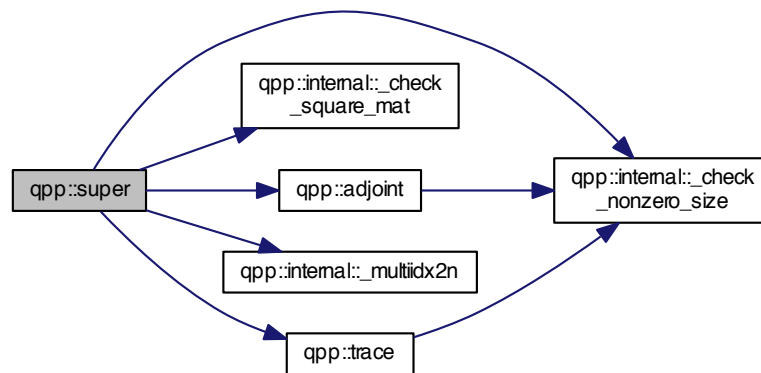
Parameters

<i>Ks</i>	std::vector of Eigen expressions representing the set of Kraus operators
-----------	--

Returns

Superoperator matrix representation, as a dynamic matrix over the complex field

Here is the call graph for this function:



6.1.2.96 `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*

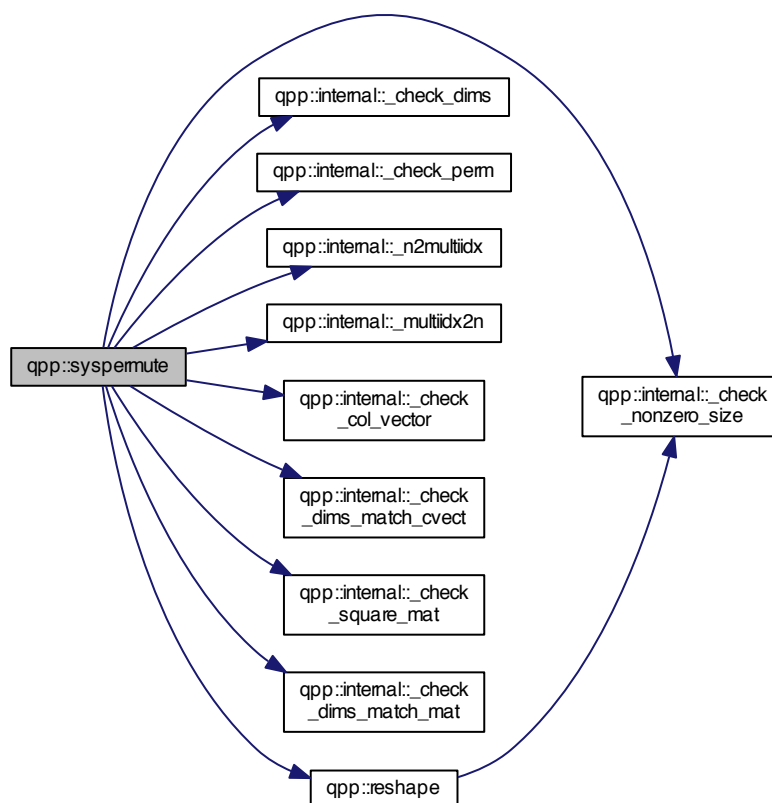
Parameters

<i>A</i>	Eigen expression
<i>perm</i>	Permutation
<i>dims</i>	Subsystems' dimensions

Returns

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

Here is the call graph for this function:



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

Trace.

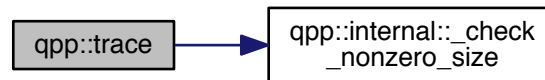
Parameters

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

Returns

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

Here is the call graph for this function:



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

Transpose.

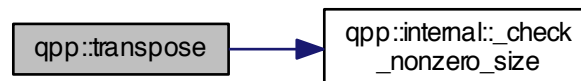
Parameters

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

Returns

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

Here is the call graph for this function:



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

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

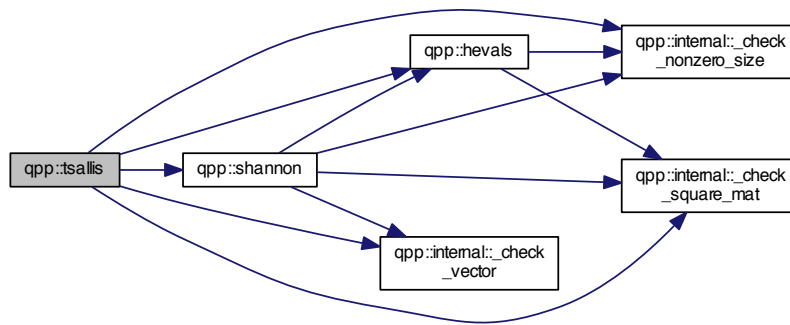
<i>alpha</i>	Non-negative real number
--------------	--------------------------

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

Returns

Renyi- α entropy, with the logarithm in base 2

Here is the call graph for this function:



6.1.3 Variable Documentation

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

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

6.1.3.2 constexpr double qpp::ee = 2.718281828459045235360287471352662497

Base of natural logarithm, e .

6.1.3.3 constexpr double qpp::eps = 1e-12

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

Example:

```
if(std::abs(x) < qpp::eps) // x is zero
```

6.1.3.4 const Gates& qpp::gt = Gates::get_instance()

[qpp::Gates](#) const [Singleton](#)

Initializes the gates, see the class [qpp::Gates](#)

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

Maximum number of qubits.

Used internally to statically allocate arrays (for speed reasons)

6.1.3.6 constexpr double qpp::pi = 3.141592653589793238462643383279502884

π

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

[qpp::RandomDevices](#) Singleton

Initializes the random devices, see the class [qpp::RandomDevices](#)

6.1.3.8 const States& qpp::st = States::get_instance()

[qpp::States](#) const Singleton

Initializes the states, see the class [qpp::States](#)

6.2 qpp::internal Namespace Reference

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)

6.2.1 Detailed Description

Internal functions, do not modify or use them directly

6.2.2 Function Documentation

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

6.2.2.2 `bool qpp::internal::_check_dims (const std::vector< std::size_t > & dims)`

6.2.2.3 `template<typename Derived > bool qpp::internal::_check_dims_match_cvect (const std::vector< std::size_t > & dims, const Eigen::MatrixBase< Derived > & V)`

6.2.2.4 `template<typename Derived > bool qpp::internal::_check_dims_match_mat (const std::vector< std::size_t > & dims, const Eigen::MatrixBase< Derived > & A)`

6.2.2.5 `template<typename Derived > bool qpp::internal::_check_dims_match_rvect (const std::vector< std::size_t > & dims, const Eigen::MatrixBase< Derived > & V)`

6.2.2.6 `bool qpp::internal::_check_eq_dims (const std::vector< std::size_t > & dims, std::size_t dim)`

6.2.2.7 `template<typename T > bool qpp::internal::_check_nonzero_size (const T & x)`

6.2.2.8 `bool qpp::internal::_check_perm (const std::vector< std::size_t > & perm)`

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

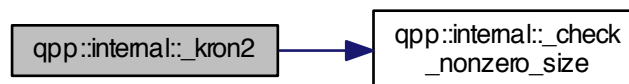
6.2.2.10 `template<typename Derived > bool qpp::internal::_check_square_mat (const Eigen::MatrixBase< Derived > & A)`

6.2.2.11 `bool qpp::internal::_check_subsys_match_dims (const std::vector< std::size_t > & subsys, const std::vector< std::size_t > & dims)`

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

6.2.2.13 `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:



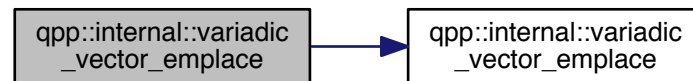
6.2.2.14 `std::size_t qpp::internal::_multiidx2n (const std::size_t * midx, std::size_t numdims, const std::size_t * dims)`

6.2.2.15 `void qpp::internal::_n2multiidx (std::size_t n, std::size_t numdims, const std::size_t * dims, std::size_t * result)`

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

6.2.2.17 `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::DiscreteDistribution< T > Class Template Reference

```
#include <stat.h>
```

Public Member Functions

- `template<typename InputIterator >`
`DiscreteDistribution` (InputIterator first, InputIterator last)
- `DiscreteDistribution` (std::initializer_list< double > weights)
- `DiscreteDistribution` (std::vector< double > weights)
- `T sample` ()
- `std::vector< double > probabilities` () const

Protected Attributes

- `std::discrete_distribution< T > _d`

7.1.1 Constructor & Destructor Documentation

7.1.1.1 `template<typename T = std::size_t> template<typename InputIterator > qpp::DiscreteDistribution< T >::DiscreteDistribution (InputIterator first, InputIterator last)` [inline]

7.1.1.2 `template<typename T = std::size_t> qpp::DiscreteDistribution< T >::DiscreteDistribution (std::initializer_list< double > weights)` [inline]

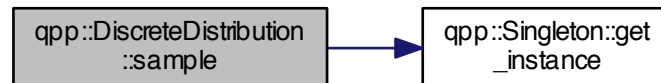
7.1.1.3 `template<typename T = std::size_t> qpp::DiscreteDistribution< T >::DiscreteDistribution (std::vector< double > weights)` [inline]

7.1.2 Member Function Documentation

7.1.2.1 `template<typename T = std::size_t> std::vector<double> qpp::DiscreteDistribution< T >::probabilities ()`
`const` [inline]

7.1.2.2 `template<typename T = std::size_t> T qpp::DiscreteDistribution< T >::sample () [inline]`

Here is the call graph for this function:



7.1.3 Member Data Documentation

7.1.3.1 `template<typename T = std::size_t> std::discrete_distribution<T> qpp::DiscreteDistribution< T >::_d [protected]`

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

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

7.2 qpp::DiscreteDistributionAbsSquare< T > Class Template Reference

```
#include <stat.h>
```

Public Member Functions

- `template<typename InputIterator >`
[DiscreteDistributionAbsSquare](#) (InputIterator first, InputIterator last)
- [DiscreteDistributionAbsSquare](#) (std::initializer_list< [cplx](#) > amplitudes)
- [DiscreteDistributionAbsSquare](#) (std::vector< [cplx](#) > amplitudes)
- `template<typename Derived >`
[DiscreteDistributionAbsSquare](#) (const Eigen::MatrixBase< Derived > &V)
- `T sample ()`
- `std::vector< double > probabilities () const`

Protected Member Functions

- `template<typename InputIterator >`
`std::vector< double > cplx2weights (InputIterator first, InputIterator last) const`

Protected Attributes

- `std::discrete_distribution< T > _d`

7.2.1 Constructor & Destructor Documentation

7.2.1.1 `template<typename T = std::size_t> template<typename InputIterator > qpp::DiscreteDistributionAbsSquare< T >::DiscreteDistributionAbsSquare (InputIterator first, InputIterator last)` `[inline]`

7.2.1.2 `template<typename T = std::size_t> qpp::DiscreteDistributionAbsSquare< T >::DiscreteDistributionAbsSquare (std::initializer_list< cplx > amplitudes)` `[inline]`

7.2.1.3 `template<typename T = std::size_t> qpp::DiscreteDistributionAbsSquare< T >::DiscreteDistributionAbsSquare (std::vector< cplx > amplitudes)` `[inline]`

7.2.1.4 `template<typename T = std::size_t> template<typename Derived > qpp::DiscreteDistributionAbsSquare< T >::DiscreteDistributionAbsSquare (const Eigen::MatrixBase< Derived > & V)` `[inline]`

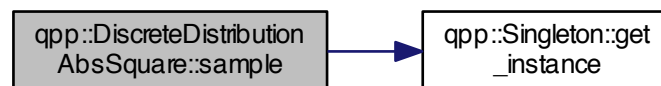
7.2.2 Member Function Documentation

7.2.2.1 `template<typename T = std::size_t> template<typename InputIterator > std::vector<double> qpp::DiscreteDistributionAbsSquare< T >::cplx2weights (InputIterator first, InputIterator last) const` `[inline]`, `[protected]`

7.2.2.2 `template<typename T = std::size_t> std::vector<double> qpp::DiscreteDistributionAbsSquare< T >::probabilities () const` `[inline]`

7.2.2.3 `template<typename T = std::size_t> T qpp::DiscreteDistributionAbsSquare< T >::sample ()` `[inline]`

Here is the call graph for this function:



7.2.3 Member Data Documentation

7.2.3.1 `template<typename T = std::size_t> std::discrete_distribution<T> qpp::DiscreteDistributionAbsSquare< T >::_d` `[protected]`

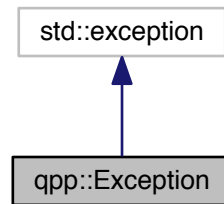
The documentation for this class was generated from the following file:

- `include/classes/stat.h`

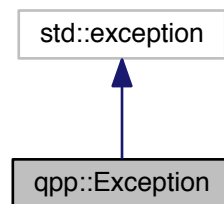
7.3 qpp::Exception Class Reference

```
#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::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::CUSTOM_EXCEPTION` }

Public Member Functions

- `Exception` (const std::string &where, const `Type` &type)
- `Exception` (const std::string &where, const std::string &custom)
- virtual const char * `what` () const noexcept override

Private Member Functions

- `std::string _construct_exception_msg ()`

Private Attributes

- `std::string _where`
- `std::string _msg`
- `Type _type`
- `std::string _custom`

7.3.1 Member Enumeration Documentation

7.3.1.1 enum `qpp::Exception::Type` [strong]

Enumerator

- UNKNOWN_EXCEPTION** Unknown exception
- ZERO_SIZE** Zero sized object, e.g. empty `Eigen::Matrix` or `std::vector` with no elements
- MATRIX_NOT_SQUARE** `Eigen::Matrix` is not square
- MATRIX_NOT_CVECTOR** `Eigen::Matrix` is not a column vector
- MATRIX_NOT_RVECTOR** `Eigen::Matrix` is not a row vector
- MATRIX_NOT_VECTOR** `Eigen::Matrix` is not a row/column vector
- MATRIX_NOT_SQUARE_OR_CVECTOR** `Eigen::Matrix` is not square nor a column vector
- MATRIX_NOT_SQUARE_OR_RVECTOR** `Eigen::Matrix` is not square nor a row vector
- MATRIX_NOT_SQUARE_OR_VECTOR** `Eigen::Matrix` is not square nor a row/column vector
- DIMS_INVALID** `std::vector<std::size_t>` representing the dimensions has zero size or contains zeros
- DIMS_NOT_EQUAL** `std::vector<std::size_t>` representing the dimensions contains non-equal elements
- 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** 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** 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** 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** `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** Invalid `std::vector<std::size_t>` permutation
- NOT_QUBIT_GATE** `Eigen::Matrix` is not 2 x 2
- NOT_QUBIT_SUBSYS** Subsystems are not 2-dimensional
- NOT_BIPARTITE** `std::vector<std::size_t>` representing the dimensions has size different from 2
- OUT_OF_RANGE** Parameter out of range
- TYPE_MISMATCH** Types do not match (i.e. `Matrix<double>` vs `Matrix<cplx>`)
- UNDEFINED_TYPE** Templated function not defined for this type
- CUSTOM_EXCEPTION** Custom exception, user must provide a custom message

7.3.2 Constructor & Destructor Documentation

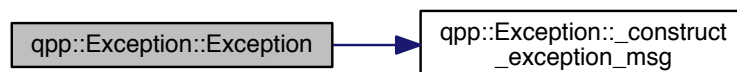
7.3.2.1 `qpp::Exception::Exception (const std::string & where, const Type & type)` `[inline]`

Here is the call graph for this function:



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

Here is the call graph for this function:



7.3.3 Member Function Documentation

7.3.3.1 `std::string qpp::Exception::_construct_exception_msg ()` `[inline]`, `[private]`

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

7.3.4 Member Data Documentation

7.3.4.1 `std::string qpp::Exception::_custom` `[private]`

7.3.4.2 `std::string qpp::Exception::_msg` `[private]`

7.3.4.3 `Type qpp::Exception::_type` `[private]`

7.3.4.4 `std::string qpp::Exception::_where` `[private]`

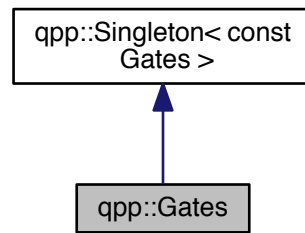
The documentation for this class was generated from the following file:

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

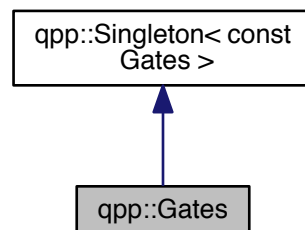
7.4 qpp::Gates Class Reference

```
#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
- [cmat Zd](#) (std::size_t D) const
- [cmat Fd](#) (std::size_t D) const
- [cmat Xd](#) (std::size_t D) const
- template<typename Derived = Eigen::MatrixXcd>
Derived [ld](#) (std::size_t D) const
- template<typename Derived1 , typename Derived2 >
[DynMat](#)< typename Derived1::Scalar > [applyCTRL](#) (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< std::size_t > &ctrl, const std::vector< std::size_t > &subsys, std::size_t n, std::size_t d=2) const
- 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) const
- 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

Public Attributes

- [cmat Id2](#) { cmat::Identity(2, 2) }
- [cmat H](#) { cmat::Zero(2, 2) }
- [cmat X](#) { cmat::Zero(2, 2) }
- [cmat Y](#) { cmat::Zero(2, 2) }
- [cmat Z](#) { cmat::Zero(2, 2) }
- [cmat S](#) { cmat::Zero(2, 2) }
- [cmat T](#) { cmat::Zero(2, 2) }
- [cmat CNOTab](#) { cmat::Identity(4, 4) }
- [cmat CZ](#) { cmat::Identity(4, 4) }
- [cmat CNOTba](#) { cmat::Zero(4, 4) }
- [cmat SWAP](#) { cmat::Identity(4, 4) }
- [cmat TOF](#) { cmat::Identity(8, 8) }
- [cmat FRED](#) { cmat::Identity(8, 8) }

Private Member Functions

- [Gates](#) ()

Friends

- class [Singleton](#)< const [Gates](#) >

Additional Inherited Members

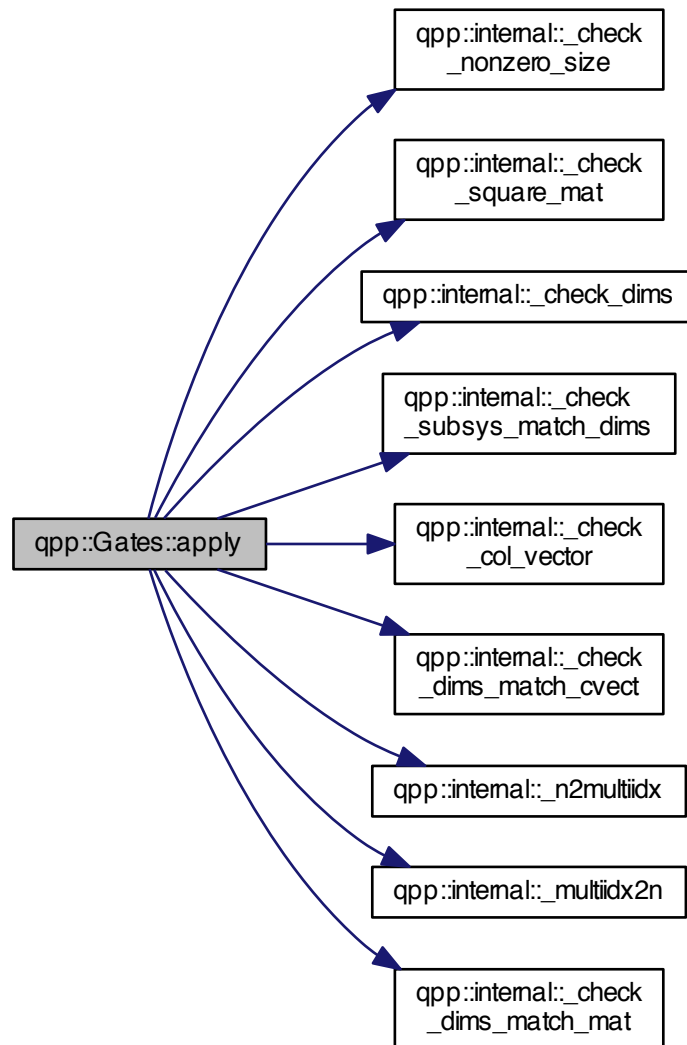
7.4.1 Constructor & Destructor Documentation

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

7.4.2 Member Function Documentation

7.4.2.1 `template<typename Derived1 , typename Derived2 > DynMat<typename Derived1::Scalar> qpp::Gates::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) const [inline]`

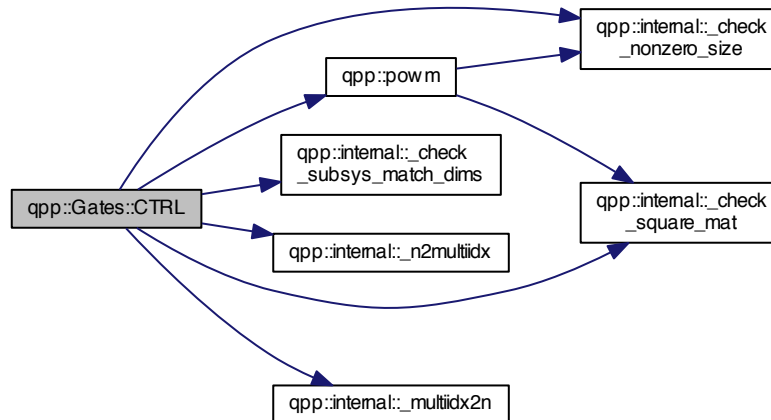
Here is the call graph for this function:



7.4.2.2 `template<typename Derived1 , typename Derived2 > DynMat<typename Derived1::Scalar> qpp::Gates::applyCTRL
(const Eigen::MatrixBase< Derived1 > & state, const Eigen::MatrixBase< Derived2 > & A, const std::vector<
std::size_t > & ctrl, const std::vector< std::size_t > & subsys, std::size_t n, std::size_t d=2) const [inline]`

7.4.2.3 `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]`

Here is the call graph for this function:



7.4.2.4 `cmat qpp::Gates::Fd (std::size_t D) const [inline]`

Here is the call graph for this function:



7.4.2.5 `template<typename Derived = Eigen::MatrixXcd> Derived qpp::Gates::Id (std::size_t D) const [inline]`

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

7.4.2.7 `cmat qpp::Gates::Xd (std::size_t D) const [inline]`

Here is the call graph for this function:



7.4.2.8 `cmat qpp::Gates::Zd (std::size_t D) const [inline]`

Here is the call graph for this function:



7.4.3 Friends And Related Function Documentation

7.4.3.1 `friend class Singleton< const Gates > [friend]`

7.4.4 Member Data Documentation

7.4.4.1 `cmat qpp::Gates::CNOTab { cmat::Identity(4, 4) }`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

7.5 `qpp::NormalDistribution< T >` Class Template Reference

```
#include <stat.h>
```

Public Member Functions

- [NormalDistribution](#) (T mean=0, T sigma=1)
- T [sample](#) ()

Protected Attributes

- `std::normal_distribution< T > _d`

7.5.1 Constructor & Destructor Documentation

7.5.1.1 `template<typename T = double> qpp::NormalDistribution< T >::NormalDistribution (T mean = 0, T sigma = 1) [inline]`

7.5.2 Member Function Documentation

7.5.2.1 `template<typename T = double> T qpp::NormalDistribution< T >::sample () [inline]`

Here is the call graph for this function:



7.5.3 Member Data Documentation

7.5.3.1 `template<typename T = double> std::normal_distribution<T> qpp::NormalDistribution< T >::_d [protected]`

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

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

7.6 qpp::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

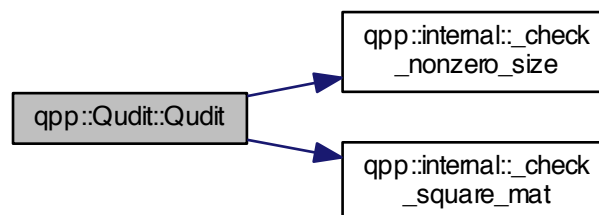
Private Attributes

- [cmat _rho](#)
- [std::size_t _D](#)

7.6.1 Constructor & Destructor Documentation

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

Here is the call graph for this function:



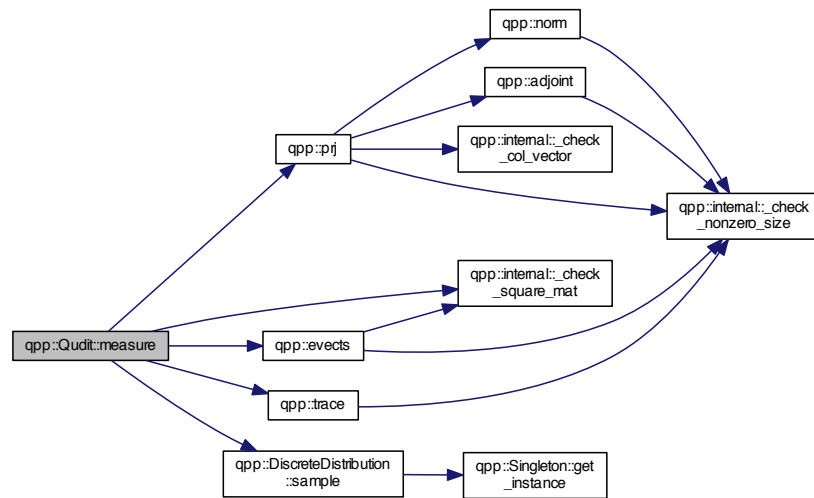
7.6.2 Member Function Documentation

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

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

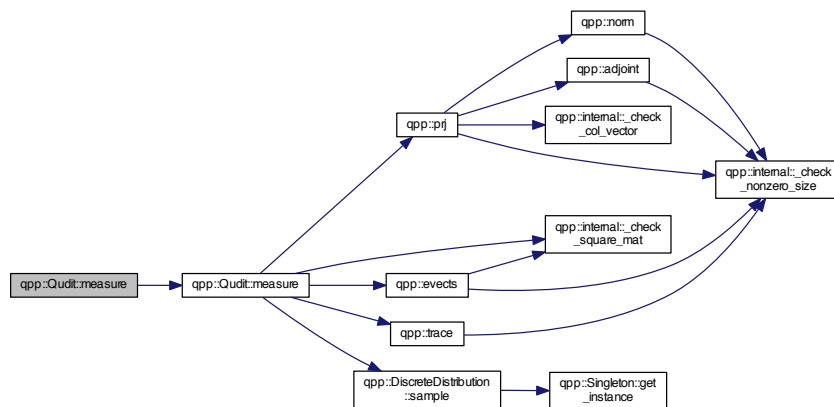
7.6.2.3 `std::size_t qpp::Qudit::measure (const cmat & U, bool destructive = false) [inline]`

Here is the call graph for this function:



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

Here is the call graph for this function:



7.6.3 Member Data Documentation

7.6.3.1 `std::size_t qpp::Qudit::_D [private]`

7.6.3.2 `cmat qpp::Qudit::_rho [private]`

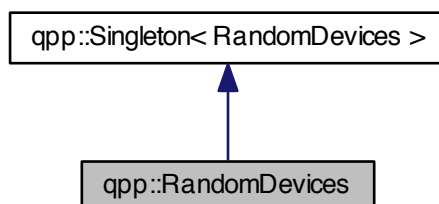
The documentation for this class was generated from the following file:

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

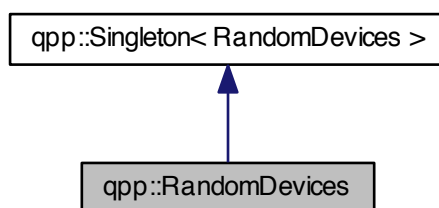
7.7 qpp::RandomDevices Class Reference

```
#include <randevs.h>
```

Inheritance diagram for qpp::RandomDevices:



Collaboration diagram for qpp::RandomDevices:



Public Attributes

- [std::mt19937 _rng](#)

Private Member Functions

- [RandomDevices \(\)](#)

Private Attributes

- [std::random_device _rd](#)

Friends

- class [Singleton< RandomDevices >](#)

Additional Inherited Members

7.7.1 Constructor & Destructor Documentation

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

7.7.2 Friends And Related Function Documentation

7.7.2.1 `friend class Singleton< RandomDevices >` `[friend]`

7.7.3 Member Data Documentation

7.7.3.1 `std::random_device qpp::RandomDevices::_rd` `[private]`

7.7.3.2 `std::mt19937 qpp::RandomDevices::_rng`

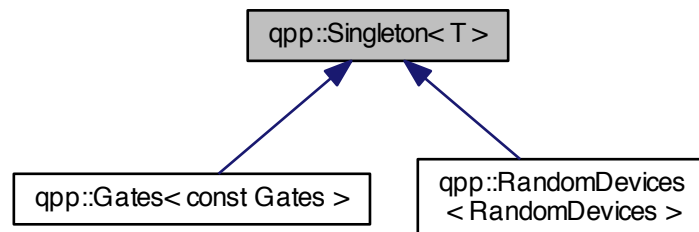
The documentation for this class was generated from the following file:

- `include/classes/randevs.h`

7.8 `qpp::Singleton< T >` Class Template Reference

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

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



Static Public Member Functions

- static `T & get_instance ()`

Protected Member Functions

- `Singleton ()`=default
- virtual `~Singleton ()`
- `Singleton (const Singleton &)=delete`
- `Singleton & operator= (const Singleton &)=delete`

7.8.1 Constructor & Destructor Documentation

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

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

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

7.8.2 Member Function Documentation

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

7.8.2.2 `template<typename T> Singleton& qpp::Singleton< T >::operator= (const Singleton< T > &)` `[protected]`, `[delete]`

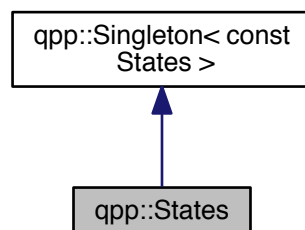
The documentation for this class was generated from the following file:

- `include/classes/singleton.h`

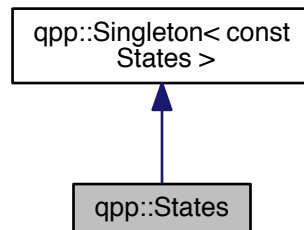
7.9 qpp::States Class Reference

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

Inheritance diagram for qpp::States:



Collaboration diagram for qpp::States:



Public Attributes

- [ket x0](#) { ket::Zero(2) }
- [ket x1](#) { ket::Zero(2) }
- [ket y0](#) { ket::Zero(2) }
- [ket y1](#) { ket::Zero(2) }
- [ket z0](#) { ket::Zero(2) }
- [ket z1](#) { ket::Zero(2) }
- [cmat px0](#) { cmat::Zero(2, 2) }
- [cmat px1](#) { cmat::Zero(2, 2) }
- [cmat py0](#) { cmat::Zero(2, 2) }
- [cmat py1](#) { cmat::Zero(2, 2) }
- [cmat pz0](#) { cmat::Zero(2, 2) }
- [cmat pz1](#) { cmat::Zero(2, 2) }
- [ket b00](#) { ket::Zero(4) }
- [ket b01](#) { ket::Zero(4) }
- [ket b10](#) { ket::Zero(4) }
- [ket b11](#) { ket::Zero(4) }
- [cmat pb00](#) { cmat::Zero(4, 4) }
- [cmat pb01](#) { cmat::Zero(4, 4) }
- [cmat pb10](#) { cmat::Zero(4, 4) }
- [cmat pb11](#) { cmat::Zero(4, 4) }
- [ket GHZ](#) { ket::Zero(8) }
- [ket W](#) { ket::Zero(8) }
- [cmat pGHZ](#) { cmat::Zero(8, 8) }
- [cmat pW](#) { cmat::Zero(8, 8) }

Private Member Functions

- [States](#) ()

Friends

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

Additional Inherited Members

7.9.1 Constructor & Destructor Documentation

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

7.9.2 Friends And Related Function Documentation

7.9.2.1 `friend class Singleton< const States > [friend]`

7.9.3 Member Data Documentation

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

7.10 qpp::Timer Class Reference

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

Public Member Functions

- [Timer](#) ()
- void [tic](#) ()
- void [toc](#) ()
- double [seconds](#) () const

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)`

7.10.1 Constructor & Destructor Documentation

7.10.1.1 `qpp::Timer::Timer ()` [[inline](#)]

7.10.2 Member Function Documentation

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

7.10.2.2 `void qpp::Timer::tic ()` [[inline](#)]

7.10.2.3 `void qpp::Timer::toc ()` [[inline](#)]

7.10.3 Friends And Related Function Documentation

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

7.10.4 Member Data Documentation

7.10.4.1 `std::chrono::steady_clock::time_point` `qpp::Timer::_end` [[protected](#)]

7.10.4.2 `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](#)

7.11 qpp::UniformIntegerDistribution< T > Class Template Reference

```
#include <stat.h>
```

Public Member Functions

- [UniformIntegerDistribution](#) (T a=std::numeric_limits< T >::min(), T b=std::numeric_limits< T >::max())
- T [sample](#) ()

Protected Attributes

- std::uniform_int_distribution< T > [_d](#)

7.11.1 Constructor & Destructor Documentation

7.11.1.1 `template<typename T = int> qpp::UniformIntegerDistribution< T >::UniformIntegerDistribution (T a = std::numeric_limits<T>::min(), T b = std::numeric_limits<T>::max()) [inline]`

7.11.2 Member Function Documentation

7.11.2.1 `template<typename T = int> T qpp::UniformIntegerDistribution< T >::sample () [inline]`

Here is the call graph for this function:



7.11.3 Member Data Documentation

7.11.3.1 `template<typename T = int> std::uniform_int_distribution<T> qpp::UniformIntegerDistribution< T >::_d [protected]`

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

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

7.12 qpp::UniformRealDistribution< T > Class Template Reference

```
#include <stat.h>
```

Public Member Functions

- [UniformRealDistribution](#) (T a=0, T b=1)
- T [sample](#) ()

Protected Attributes

- std::uniform_real_distribution< T > [_d](#)

7.12.1 Constructor & Destructor Documentation

- 7.12.1.1 `template<typename T = double> qpp::UniformRealDistribution< T >::UniformRealDistribution (T a = 0, T b = 1) [inline]`

7.12.2 Member Function Documentation

- 7.12.2.1 `template<typename T = double> T qpp::UniformRealDistribution< T >::sample () [inline]`

Here is the call graph for this function:



7.12.3 Member Data Documentation

- 7.12.3.1 `template<typename T = double> std::uniform_real_distribution<T> qpp::UniformRealDistribution< T >::_d [protected]`

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

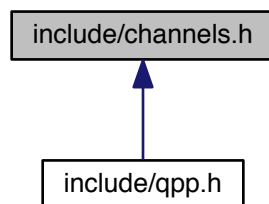
- include/classes/[stat.h](#)

Chapter 8

File Documentation

8.1 include/channels.h File Reference

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



Namespaces

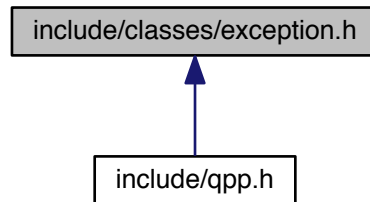
- [qpp](#)

Functions

- `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 >`
`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.

8.2 include/classes/exception.h File Reference

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



Classes

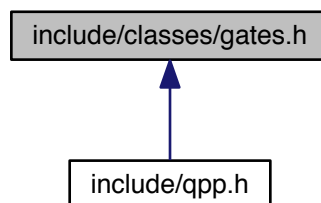
- class [qpp::Exception](#)

Namespaces

- [qpp](#)

8.3 include/classes/gates.h File Reference

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



Classes

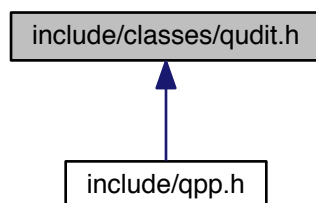
- class [qpp::Gates](#)

Namespaces

- [qpp](#)

8.4 include/classes/qudit.h File Reference

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



Classes

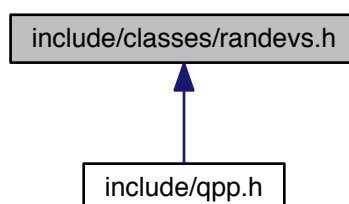
- class [qpp::Qudit](#)

Namespaces

- [qpp](#)

8.5 include/classes/randevs.h File Reference

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



Classes

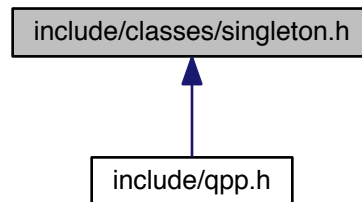
- class [qpp::RandomDevices](#)

Namespaces

- [qpp](#)

8.6 include/classes/singleton.h File Reference

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



Classes

- class [qpp::Singleton< T >](#)

Namespaces

- [qpp](#)

Macros

- `#define` [CLASS_SINGLETON\(Foo\)](#)
- `#define` [CLASS_CONST_SINGLETON\(Foo\)](#)

8.6.1 Macro Definition Documentation

8.6.1.1 `#define CLASS_CONST_SINGLETON(Foo)`

Value:

```
class Foo: public Singleton<const Foo>\n{\n    friend class Singleton<const Foo>;
```

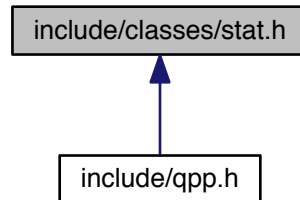
8.6.1.2 `#define CLASS_SINGLETON(Foo)`

Value:

```
class Foo: public Singleton<Foo>\n{\n    friend class Singleton<Foo>;
```

8.7 include/classes/stat.h File Reference

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



Classes

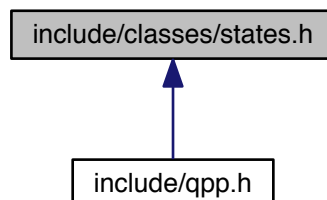
- class [qpp::NormalDistribution< T >](#)
- class [qpp::UniformRealDistribution< T >](#)
- class [qpp::UniformIntegerDistribution< T >](#)
- class [qpp::DiscreteDistribution< T >](#)
- class [qpp::DiscreteDistributionAbsSquare< T >](#)

Namespaces

- [qpp](#)

8.8 include/classes/states.h File Reference

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



Classes

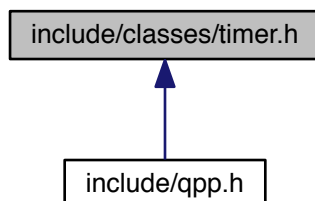
- class [qpp::States](#)

Namespaces

- [qpp](#)

8.9 include/classes/timer.h File Reference

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



Classes

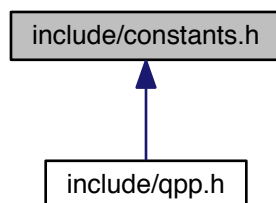
- class [qpp::Timer](#)

Namespaces

- [qpp](#)

8.10 include/constants.h File Reference

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



Namespaces

- [qpp](#)

Functions

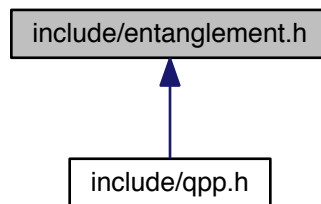
- constexpr std::complex< double > [qpp::operator""_i](#) (unsigned long long int x)
User-defined literal for complex $i = \sqrt{-1}$ (integer overload)
- constexpr std::complex< double > [qpp::operator""_i](#) (long double x)
User-defined literal for complex $i = \sqrt{-1}$ (real overload)
- std::complex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

8.11 include/entanglement.h File Reference

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



Namespaces

- [qpp](#)

Functions

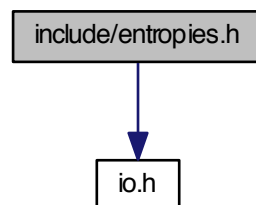
- template<typename Derived >
cmat [qpp::schmidtcoeff](#) (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &dims)
Schmidt coefficients of the bi-partite pure state A.

- `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 >`
`cmat 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.

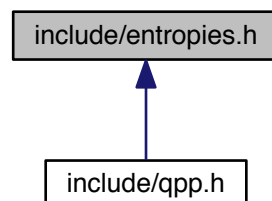
8.12 include/entropies.h File Reference

```
#include "io.h"
```

Include dependency graph for entropies.h:



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



Namespaces

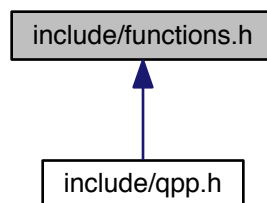
- [qpp](#)

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

8.13 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.
- `template<typename Derived >`
`double qpp::norm (const Eigen::MatrixBase< Derived > &A)`
Trace norm.
- `template<typename Derived >`
`cmat qpp::evals (const Eigen::MatrixBase< Derived > &A)`
Eigenvalues.
- `template<typename Derived >`
`cmat qpp::evecs (const Eigen::MatrixBase< Derived > &A)`
Eigenvectors.
- `template<typename Derived >`
`dmat qpp::hevals (const Eigen::MatrixBase< Derived > &A)`
Hermitian eigenvalues.
- `template<typename Derived >`
`cmat qpp::hevecs (const Eigen::MatrixBase< Derived > &A)`
Hermitian eigenvectors.
- `template<typename Derived >`
`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::spectralpowm](#) (const Eigen::MatrixBase< Derived > &A, const cplx z)

Matrix power.

- template<typename Derived >
DynMat< typename Derived::Scalar > [qpp::powm](#) (const Eigen::MatrixBase< Derived > &A, std::size_t n)

Matrix power.

- template<typename 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 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.

- 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.

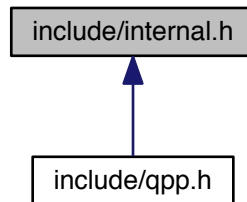
- template<typename Derived >
DynMat< typename Derived::Scalar > [qpp::ptranspose](#) (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &subsys, const std::vector< std::size_t > &dims)

Partial transpose.

- `template<typename 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::expandout (const Eigen::MatrixBase< Derived > &A, std::size_t pos, const std::vector< std::size_t > &dims)`
Expand out.
- `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)`
Multi-partite qubit ket.
- `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)`
Multi-partite qudit ket (same dimensions overload)
- `std::vector< std::size_t > qpp::invperm (const std::vector< std::size_t > &perm)`
Inverse permutation.
- `std::vector< std::size_t > qpp::compperm (const std::vector< std::size_t > &perm, const std::vector< std::size_t > &sigma)`
Compose permutations.

8.14 include/internal.h File Reference

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



Namespaces

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

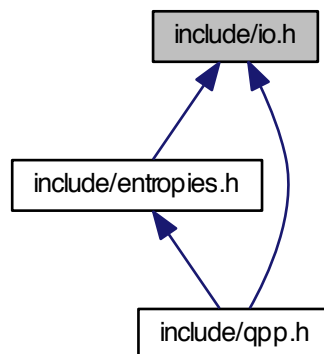
Functions

- 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)`

8.15 include/io.h File Reference

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



Namespaces

- [qpp](#)

Functions

- `template<typename T >`
`void qpp::disp (const T &x, const std::string &separator, const std::string &start="[" , const std::string &end="]", std::ostream &os=std::cout)`
Displays a standard container that supports std::begin, std::end and forward iteration. Does not add a newline.
- `template<typename T >`
`void qpp::displn (const T &x, const std::string &separator, const std::string &start="[" , const std::string &end="]", std::ostream &os=std::cout)`
Displays a standard container that supports std::begin, std::end and forward iteration. Adds a newline.
- `template<typename T >`
`void qpp::disp (const T *x, const std::size_t n, const std::string &separator, const std::string &start="[" , const std::string &end="]", std::ostream &os=std::cout)`
Displays a C-style array. Does not add a newline.
- `template<typename T >`
`void qpp::displn (const T *x, const std::size_t n, const std::string &separator, const std::string &start="[" , const std::string &end="]", std::ostream &os=std::cout)`
Displays a C-style array. Adds a newline.
- `template<typename Derived >`
`void qpp::disp (const Eigen::MatrixBase< Derived > &A, double chop=qpp::chop, std::ostream &os=std::cout)`

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

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

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

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

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

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

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

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

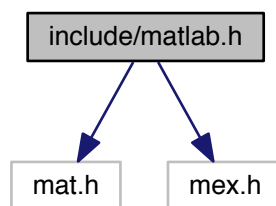
Saves Eigen expression to a binary file (internal format) in double precision.

- `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.

8.16 include/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](#))

8.17 include/qpp.h File Reference

```
#include <algorithm>
#include <chrono>
#include <cmath>
#include <complex>
#include <cstdlib>
#include <cstring>
#include <exception>
#include <fstream>
#include <functional>
#include <iomanip>
#include <iostream>
#include <iterator>
#include <limits>
#include <numeric>
#include <ostream>
#include <random>
#include <sstream>
#include <stdexcept>
#include <string>
#include <type_traits>
#include <utility>
#include <vector>
#include <Eigen/Dense>
#include <Eigen/SVD>
#include "constants.h"
#include "types.h"
#include "classes/exception.h"
#include "classes/singleton.h"
#include "classes/states.h"
#include "classes/randevs.h"
#include "internal.h"
#include "functions.h"
#include "classes/gates.h"
#include "classes/stat.h"
#include "entropies.h"
#include "entanglement.h"
#include "channels.h"
#include "io.h"
#include "random.h"
#include "classes/qudit.h"
#include "classes/timer.h"
```

Include dependency graph for qpp.h:



Namespaces

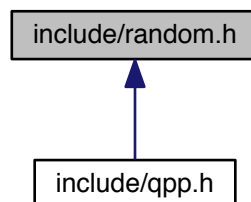
- [qpp](#)

Variables

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

8.18 include/random.h File Reference

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



Namespaces

- [qpp](#)

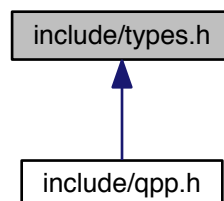
Functions

- `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 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 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 (double) 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 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 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.

8.19 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.
- using [qpp::cmat](#) = Eigen::MatrixXcd
Complex (double precision) dynamic Eigen matrix.
- using [qpp::dmat](#) = Eigen::MatrixXd
Real (double precision) dynamic Eigen matrix.
- using [qpp::ket](#) = Eigen::Matrix< cplx, Eigen::Dynamic, 1 >
Complex (double precision) dynamic Eigen column matrix.
- using [qpp::bra](#) = Eigen::Matrix< cplx, 1, Eigen::Dynamic >
Complex (double precision) dynamic Eigen row matrix.
- template<typename Scalar >
using [qpp::DynMat](#) = Eigen::Matrix< Scalar, Eigen::Dynamic, Eigen::Dynamic >
Dynamic Eigen matrix over the field specified by Scalar.

Index

absm
 qpp, [18](#)
adjoint
 qpp, [19](#)
anticomm
 qpp, [19](#)

bra
 qpp, [18](#)

CUSTOM_EXCEPTION
 qpp::Exception, [85](#)
channel
 qpp, [20](#)
choi
 qpp, [22](#)
choi2kraus
 qpp, [23](#)
chop
 qpp, [76](#)
cmat
 qpp, [18](#)
comm
 qpp, [24](#)
compperm
 qpp, [24](#)
conjugate
 qpp, [26](#)
cosm
 qpp, [26](#)
cplx
 qpp, [18](#)
cwise
 qpp, [27](#)

DIMS_INVALID
 qpp::Exception, [85](#)
DIMS_MISMATCH_CVECTOR
 qpp::Exception, [85](#)
DIMS_MISMATCH_MATRIX
 qpp::Exception, [85](#)
DIMS_MISMATCH_RVECTOR
 qpp::Exception, [85](#)
DIMS_MISMATCH_VECTOR
 qpp::Exception, [85](#)
DIMS_NOT_EQUAL
 qpp::Exception, [85](#)
det
 qpp, [27](#)
disp
 qpp, [28](#), [29](#)
displn
 qpp, [29–31](#)
dmat
 qpp, [18](#)

ee
 qpp, [76](#)
entanglement
 qpp, [31](#)
eps
 qpp, [76](#)
evals
 qpp, [32](#)
evects
 qpp, [33](#)
expandout
 qpp, [33](#)
expm
 qpp, [34](#)

funm
 qpp, [35](#)

gconcurrency
 qpp, [35](#)
grams
 qpp, [36](#), [37](#)
gt
 qpp, [76](#)

hevals
 qpp, [38](#)
hevects
 qpp, [38](#)

inverse
 qpp, [39](#)
invperm
 qpp, [39](#)

ket
 qpp, [18](#)
kron
 qpp, [40](#), [41](#)
kronpow
 qpp, [42](#)

load
 qpp, [42](#)
logdet

- qpp, [44](#)
- logm
 - qpp, [44](#)
- MATRIX_NOT_CVECTOR
 - qpp::Exception, [85](#)
- MATRIX_NOT_RVECTOR
 - qpp::Exception, [85](#)
- MATRIX_NOT_SQUARE
 - qpp::Exception, [85](#)
- MATRIX_NOT_SQUARE_OR_CVECTOR
 - qpp::Exception, [85](#)
- MATRIX_NOT_SQUARE_OR_RVECTOR
 - qpp::Exception, [85](#)
- MATRIX_NOT_SQUARE_OR_VECTOR
 - qpp::Exception, [85](#)
- MATRIX_NOT_VECTOR
 - qpp::Exception, [85](#)
- maxn
 - qpp, [76](#)
- mket
 - qpp, [45](#), [46](#)
- multiidx2n
 - qpp, [46](#)
- n2multiidx
 - qpp, [47](#)
- NOT_BIPARTITE
 - qpp::Exception, [85](#)
- NOT_QUBIT_GATE
 - qpp::Exception, [85](#)
- NOT_QUBIT_SUBSYS
 - qpp::Exception, [85](#)
- norm
 - qpp, [47](#)
- OUT_OF_RANGE
 - qpp::Exception, [85](#)
- omega
 - qpp, [48](#)
- PERM_INVALID
 - qpp::Exception, [85](#)
- pi
 - qpp, [76](#)
- powm
 - qpp, [48](#)
- prj
 - qpp, [49](#)
- ptrace
 - qpp, [50](#)
- ptrace1
 - qpp, [51](#)
- ptrace2
 - qpp, [52](#)
- ptranspose
 - qpp, [53](#)
- qmutualinfo
 - qpp, [54](#)
- qpp, [11](#)
 - absm, [18](#)
 - adjoint, [19](#)
 - anticomm, [19](#)
 - bra, [18](#)
 - channel, [20](#)
 - choi, [22](#)
 - choi2kraus, [23](#)
 - chop, [76](#)
 - cmat, [18](#)
 - comm, [24](#)
 - compperm, [24](#)
 - conjugate, [26](#)
 - cosm, [26](#)
 - cplx, [18](#)
 - cwise, [27](#)
 - det, [27](#)
 - disp, [28](#), [29](#)
 - displn, [29–31](#)
 - dmat, [18](#)
 - ee, [76](#)
 - entanglement, [31](#)
 - eps, [76](#)
 - evals, [32](#)
 - evects, [33](#)
 - expandout, [33](#)
 - expm, [34](#)
 - funm, [35](#)
 - gconcurrence, [35](#)
 - grams, [36](#), [37](#)
 - gt, [76](#)
 - hevals, [38](#)
 - hevects, [38](#)
 - inverse, [39](#)
 - invperm, [39](#)
 - ket, [18](#)
 - kron, [40](#), [41](#)
 - kronpow, [42](#)
 - load, [42](#)
 - logdet, [44](#)
 - logm, [44](#)
 - maxn, [76](#)
 - mket, [45](#), [46](#)
 - multiidx2n, [46](#)
 - n2multiidx, [47](#)
 - norm, [47](#)
 - omega, [48](#)
 - pi, [76](#)
 - powm, [48](#)
 - prj, [49](#)
 - ptrace, [50](#)
 - ptrace1, [51](#)
 - ptrace2, [52](#)
 - ptranspose, [53](#)
 - qmutualinfo, [54](#)
 - rand, [55](#), [56](#)
 - randint, [57](#)

- randket, 58
- randkraus, 58
- randn, 59, 60
- randperm, 61
- randrho, 61
- rdevs, 77
- renyi, 62
- reshape, 63
- save, 64
- schmidtcoeff, 65
- schmidtprob, 66
- shannon, 69
- sinm, 69
- spectralpowm, 71
- sqrtn, 71
- st, 77
- sum, 72
- super, 72
- syspermute, 73
- trace, 74
- transpose, 75
- tsallis, 75
- qpp::Exception
 - CUSTOM_EXCEPTION, 85
 - DIMS_INVALID, 85
 - DIMS_MISMATCH_CVECTOR, 85
 - DIMS_MISMATCH_MATRIX, 85
 - DIMS_MISMATCH_RVECTOR, 85
 - DIMS_MISMATCH_VECTOR, 85
 - DIMS_NOT_EQUAL, 85
 - MATRIX_NOT_CVECTOR, 85
 - MATRIX_NOT_RVECTOR, 85
 - MATRIX_NOT_SQUARE, 85
 - MATRIX_NOT_SQUARE_OR_CVECTOR, 85
 - MATRIX_NOT_SQUARE_OR_RVECTOR, 85
 - MATRIX_NOT_SQUARE_OR_VECTOR, 85
 - MATRIX_NOT_VECTOR, 85
 - NOT_BIPARTITE, 85
 - NOT_QUBIT_GATE, 85
 - NOT_QUBIT_SUBSYS, 85
 - OUT_OF_RANGE, 85
 - PERM_INVALID, 85
 - SUBSYS_MISMATCH_DIMS, 85
 - TYPE_MISMATCH, 85
 - UNDEFINED_TYPE, 85
 - UNKNOWN_EXCEPTION, 85
 - ZERO_SIZE, 85
- rand
 - qpp, 55, 56
- randint
 - qpp, 57
- randket
 - qpp, 58
- randkraus
 - qpp, 58
- randn
 - qpp, 59, 60
- randperm
 - qpp, 61
- randrho
 - qpp, 61
- rdevs
 - qpp, 77
- renyi
 - qpp, 62
- reshape
 - qpp, 63
- SUBSYS_MISMATCH_DIMS
 - qpp::Exception, 85
- save
 - qpp, 64
- schmidtcoeff
 - qpp, 65
- schmidtprob
 - qpp, 66
- shannon
 - qpp, 69
- sinm
 - qpp, 69
- spectralpowm
 - qpp, 71
- sqrtn
 - qpp, 71
- st
 - qpp, 77
- sum
 - qpp, 72
- super
 - qpp, 72
- syspermute
 - qpp, 73
- TYPE_MISMATCH
 - qpp::Exception, 85
- trace
 - qpp, 74
- transpose
 - qpp, 75
- tsallis
 - qpp, 75
- UNDEFINED_TYPE
 - qpp::Exception, 85
- UNKNOWN_EXCEPTION
 - qpp::Exception, 85
- ZERO_SIZE
 - qpp::Exception, 85