

quantum++

0.1

Generated by Doxygen 1.8.7

Fri Oct 24 2014 18:26:08



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Namespace Documentation</b>	<b>9</b>
5.1	qpp Namespace Reference . . . . .	9
5.1.1	Typedef Documentation . . . . .	15
5.1.1.1	bra . . . . .	15
5.1.1.2	cmat . . . . .	15
5.1.1.3	cplx . . . . .	15
5.1.1.4	dmat . . . . .	15
5.1.1.5	DynMat . . . . .	16
5.1.1.6	ket . . . . .	16
5.1.2	Function Documentation . . . . .	16
5.1.2.1	absm . . . . .	16
5.1.2.2	adjoint . . . . .	16
5.1.2.3	anticomm . . . . .	17
5.1.2.4	channel . . . . .	18
5.1.2.5	channel . . . . .	19
5.1.2.6	choi . . . . .	20
5.1.2.7	choi2kraus . . . . .	21
5.1.2.8	comm . . . . .	22
5.1.2.9	compperm . . . . .	23
5.1.2.10	conjugate . . . . .	24
5.1.2.11	cosm . . . . .	24

5.1.2.12	<code>cwise</code>	25
5.1.2.13	<code>det</code>	25
5.1.2.14	<code>disp</code>	26
5.1.2.15	<code>disp</code>	26
5.1.2.16	<code>disp</code>	27
5.1.2.17	<code>disp</code>	27
5.1.2.18	<code>displn</code>	27
5.1.2.19	<code>displn</code>	28
5.1.2.20	<code>displn</code>	28
5.1.2.21	<code>displn</code>	29
5.1.2.22	<code>entanglement</code>	29
5.1.2.23	<code>evals</code>	30
5.1.2.24	<code>evects</code>	31
5.1.2.25	<code>expandout</code>	31
5.1.2.26	<code>expm</code>	32
5.1.2.27	<code>funm</code>	33
5.1.2.28	<code>gconcurrence</code>	33
5.1.2.29	<code>grams</code>	34
5.1.2.30	<code>grams</code>	35
5.1.2.31	<code>grams</code>	35
5.1.2.32	<code>hevals</code>	36
5.1.2.33	<code>hevects</code>	36
5.1.2.34	<code>inverse</code>	37
5.1.2.35	<code>invperm</code>	37
5.1.2.36	<code>kron</code>	38
5.1.2.37	<code>kron</code>	38
5.1.2.38	<code>kron</code>	39
5.1.2.39	<code>kron</code>	39
5.1.2.40	<code>kronpow</code>	40
5.1.2.41	<code>load</code>	40
5.1.2.42	<code>loadMATLABmatrix</code>	41
5.1.2.43	<code>loadMATLABmatrix</code>	41
5.1.2.44	<code>loadMATLABmatrix</code>	41
5.1.2.45	<code>logdet</code>	41
5.1.2.46	<code>logm</code>	42
5.1.2.47	<code>mket</code>	42
5.1.2.48	<code>mket</code>	43
5.1.2.49	<code>mket</code>	43
5.1.2.50	<code>multiidx2n</code>	44
5.1.2.51	<code>n2multiidx</code>	44

5.1.2.52	<code>norm</code>	45
5.1.2.53	<code>omega</code>	45
5.1.2.54	<code>operator""_i</code>	46
5.1.2.55	<code>operator""_i</code>	46
5.1.2.56	<code>powm</code>	46
5.1.2.57	<code>prj</code>	47
5.1.2.58	<code>ptrace</code>	47
5.1.2.59	<code>ptrace1</code>	48
5.1.2.60	<code>ptrace2</code>	49
5.1.2.61	<code>ptranspose</code>	50
5.1.2.62	<code>qmutualinfo</code>	51
5.1.2.63	<code>rand</code>	52
5.1.2.64	<code>rand</code>	52
5.1.2.65	<code>rand</code>	52
5.1.2.66	<code>rand</code>	53
5.1.2.67	<code>randH</code>	53
5.1.2.68	<code>randint</code>	53
5.1.2.69	<code>randket</code>	54
5.1.2.70	<code>randkraus</code>	54
5.1.2.71	<code>randn</code>	54
5.1.2.72	<code>randn</code>	54
5.1.2.73	<code>randn</code>	55
5.1.2.74	<code>randn</code>	55
5.1.2.75	<code>randperm</code>	55
5.1.2.76	<code>randrho</code>	56
5.1.2.77	<code>randU</code>	56
5.1.2.78	<code>randV</code>	56
5.1.2.79	<code>renyi</code>	56
5.1.2.80	<code>renyi_inf</code>	57
5.1.2.81	<code>reshape</code>	57
5.1.2.82	<code>save</code>	59
5.1.2.83	<code>saveMATLABmatrix</code>	59
5.1.2.84	<code>saveMATLABmatrix</code>	59
5.1.2.85	<code>saveMATLABmatrix</code>	60
5.1.2.86	<code>schmidtcoeff</code>	60
5.1.2.87	<code>schmidtprob</code>	61
5.1.2.88	<code>schmidtU</code>	62
5.1.2.89	<code>schmidtV</code>	63
5.1.2.90	<code>shannon</code>	64
5.1.2.91	<code>sinm</code>	65

5.1.2.92	spectralpowm	66
5.1.2.93	sqrtn	66
5.1.2.94	sum	67
5.1.2.95	super	67
5.1.2.96	syspermute	68
5.1.2.97	trace	69
5.1.2.98	transpose	70
5.1.2.99	tsallis	70
5.1.3	Variable Documentation	71
5.1.3.1	chop	71
5.1.3.2	ee	71
5.1.3.3	eps	71
5.1.3.4	gt	71
5.1.3.5	maxn	71
5.1.3.6	pi	72
5.1.3.7	rdevs	72
5.1.3.8	st	72
5.2	qpp::internal Namespace Reference	72
5.2.1	Detailed Description	73
5.2.2	Function Documentation	73
5.2.2.1	_check_col_vector	73
5.2.2.2	_check_dims	73
5.2.2.3	_check_dims_match_cvect	73
5.2.2.4	_check_dims_match_mat	73
5.2.2.5	_check_dims_match_rvect	73
5.2.2.6	_check_eq_dims	73
5.2.2.7	_check_nonzero_size	73
5.2.2.8	_check_perm	73
5.2.2.9	_check_row_vector	73
5.2.2.10	_check_square_mat	73
5.2.2.11	_check_subsys_match_dims	73
5.2.2.12	_check_vector	73
5.2.2.13	_kron2	73
5.2.2.14	_multiidx2n	73
5.2.2.15	_n2multiidx	73
5.2.2.16	variadic_vector_emplace	74
5.2.2.17	variadic_vector_emplace	74
<b>6</b>	<b>Class Documentation</b>	<b>75</b>
6.1	qpp::DiscreteDistribution Class Reference	75

6.1.1	Constructor & Destructor Documentation	75
6.1.1.1	DiscreteDistribution	75
6.1.1.2	DiscreteDistribution	75
6.1.1.3	DiscreteDistribution	75
6.1.2	Member Function Documentation	75
6.1.2.1	probabilities	75
6.1.2.2	sample	76
6.1.3	Member Data Documentation	76
6.1.3.1	_d	76
6.2	qpp::DiscreteDistributionAbsSquare Class Reference	76
6.2.1	Constructor & Destructor Documentation	77
6.2.1.1	DiscreteDistributionAbsSquare	77
6.2.1.2	DiscreteDistributionAbsSquare	77
6.2.1.3	DiscreteDistributionAbsSquare	77
6.2.1.4	DiscreteDistributionAbsSquare	77
6.2.2	Member Function Documentation	77
6.2.2.1	cplx2weights	77
6.2.2.2	probabilities	77
6.2.2.3	sample	77
6.2.3	Member Data Documentation	77
6.2.3.1	_d	77
6.3	qpp::Exception Class Reference	77
6.3.1	Member Enumeration Documentation	79
6.3.1.1	Type	79
6.3.2	Constructor & Destructor Documentation	80
6.3.2.1	Exception	80
6.3.2.2	Exception	80
6.3.3	Member Function Documentation	80
6.3.3.1	_construct_exception_msg	80
6.3.3.2	what	80
6.3.4	Member Data Documentation	80
6.3.4.1	_custom	80
6.3.4.2	_msg	80
6.3.4.3	_type	80
6.3.4.4	_where	80
6.4	qpp::Gates Class Reference	80
6.4.1	Constructor & Destructor Documentation	82
6.4.1.1	Gates	82
6.4.2	Member Function Documentation	82
6.4.2.1	apply	83

6.4.2.2	applyCTRL . . . . .	83
6.4.2.3	CTRL . . . . .	84
6.4.2.4	Fd . . . . .	84
6.4.2.5	Id . . . . .	84
6.4.2.6	Rn . . . . .	84
6.4.2.7	Xd . . . . .	85
6.4.2.8	Zd . . . . .	85
6.4.3	Friends And Related Function Documentation . . . . .	85
6.4.3.1	Singleton< const Gates > . . . . .	85
6.4.4	Member Data Documentation . . . . .	85
6.4.4.1	CNOTab . . . . .	85
6.4.4.2	CNOTba . . . . .	85
6.4.4.3	CZ . . . . .	85
6.4.4.4	FRED . . . . .	85
6.4.4.5	H . . . . .	85
6.4.4.6	Id2 . . . . .	85
6.4.4.7	S . . . . .	85
6.4.4.8	SWAP . . . . .	85
6.4.4.9	T . . . . .	85
6.4.4.10	TOF . . . . .	86
6.4.4.11	X . . . . .	86
6.4.4.12	Y . . . . .	86
6.4.4.13	Z . . . . .	86
6.5	qpp::NormalDistribution Class Reference . . . . .	86
6.5.1	Constructor & Destructor Documentation . . . . .	86
6.5.1.1	NormalDistribution . . . . .	86
6.5.2	Member Function Documentation . . . . .	86
6.5.2.1	sample . . . . .	86
6.5.3	Member Data Documentation . . . . .	86
6.5.3.1	_d . . . . .	86
6.6	qpp::Qudit Class Reference . . . . .	87
6.6.1	Constructor & Destructor Documentation . . . . .	87
6.6.1.1	Qudit . . . . .	87
6.6.2	Member Function Documentation . . . . .	87
6.6.2.1	getD . . . . .	87
6.6.2.2	getRho . . . . .	87
6.6.2.3	measure . . . . .	88
6.6.2.4	measure . . . . .	88
6.6.3	Member Data Documentation . . . . .	88
6.6.3.1	_D . . . . .	88



6.6.3.2	<a href="#">_rho</a>	88
6.7	<a href="#">qpp::RandomDevices Class Reference</a>	89
6.7.1	<a href="#">Constructor &amp; Destructor Documentation</a>	90
6.7.1.1	<a href="#">RandomDevices</a>	90
6.7.2	<a href="#">Friends And Related Function Documentation</a>	90
6.7.2.1	<a href="#">Singleton&lt; RandomDevices &gt;</a>	90
6.7.3	<a href="#">Member Data Documentation</a>	90
6.7.3.1	<a href="#">_rd</a>	90
6.7.3.2	<a href="#">_rng</a>	90
6.8	<a href="#">qpp::Singleton&lt; T &gt; Class Template Reference</a>	90
6.8.1	<a href="#">Constructor &amp; Destructor Documentation</a>	91
6.8.1.1	<a href="#">Singleton</a>	91
6.8.1.2	<a href="#">~Singleton</a>	91
6.8.1.3	<a href="#">Singleton</a>	91
6.8.2	<a href="#">Member Function Documentation</a>	91
6.8.2.1	<a href="#">get_instance</a>	91
6.8.2.2	<a href="#">operator=</a>	91
6.9	<a href="#">qpp::States Class Reference</a>	91
6.9.1	<a href="#">Constructor &amp; Destructor Documentation</a>	93
6.9.1.1	<a href="#">States</a>	93
6.9.2	<a href="#">Friends And Related Function Documentation</a>	93
6.9.2.1	<a href="#">Singleton&lt; const States &gt;</a>	93
6.9.3	<a href="#">Member Data Documentation</a>	93
6.9.3.1	<a href="#">b00</a>	93
6.9.3.2	<a href="#">b01</a>	93
6.9.3.3	<a href="#">b10</a>	93
6.9.3.4	<a href="#">b11</a>	93
6.9.3.5	<a href="#">GHZ</a>	93
6.9.3.6	<a href="#">pb00</a>	93
6.9.3.7	<a href="#">pb01</a>	93
6.9.3.8	<a href="#">pb10</a>	93
6.9.3.9	<a href="#">pb11</a>	93
6.9.3.10	<a href="#">pGHZ</a>	93
6.9.3.11	<a href="#">pW</a>	93
6.9.3.12	<a href="#">px0</a>	93
6.9.3.13	<a href="#">px1</a>	93
6.9.3.14	<a href="#">py0</a>	93
6.9.3.15	<a href="#">py1</a>	93
6.9.3.16	<a href="#">pz0</a>	93
6.9.3.17	<a href="#">pz1</a>	93

6.9.3.18	W	93
6.9.3.19	x0	93
6.9.3.20	x1	93
6.9.3.21	y0	93
6.9.3.22	y1	93
6.9.3.23	z0	94
6.9.3.24	z1	94
6.10	qpp::Timer Class Reference	94
6.10.1	Constructor & Destructor Documentation	94
6.10.1.1	Timer	94
6.10.2	Member Function Documentation	94
6.10.2.1	seconds	94
6.10.2.2	tic	94
6.10.2.3	toc	94
6.10.3	Friends And Related Function Documentation	94
6.10.3.1	operator<<	94
6.10.4	Member Data Documentation	94
6.10.4.1	_end	94
6.10.4.2	_start	94
6.11	qpp::UniformIntDistribution Class Reference	95
6.11.1	Constructor & Destructor Documentation	95
6.11.1.1	UniformIntDistribution	95
6.11.2	Member Function Documentation	95
6.11.2.1	sample	95
6.11.3	Member Data Documentation	95
6.11.3.1	_d	95
6.12	qpp::UniformRealDistribution Class Reference	95
6.12.1	Constructor & Destructor Documentation	96
6.12.1.1	UniformRealDistribution	96
6.12.2	Member Function Documentation	96
6.12.2.1	sample	96
6.12.3	Member Data Documentation	96
6.12.3.1	_d	96
<b>7</b>	<b>File Documentation</b>	<b>97</b>
7.1	include/channels.h File Reference	97
7.2	include/classes/exception.h File Reference	98
7.3	include/classes/gates.h File Reference	98
7.4	include/classes/qudit.h File Reference	99
7.5	include/classes/randevs.h File Reference	99

7.6	include/classes/singleton.h File Reference	100
7.6.1	Macro Definition Documentation	100
7.6.1.1	CLASS_CONST_SINGLETON	100
7.6.1.2	CLASS_SINGLETON	100
7.7	include/classes/stat.h File Reference	101
7.8	include/classes/states.h File Reference	101
7.9	include/classes/timer.h File Reference	102
7.10	include/constants.h File Reference	102
7.11	include/entanglement.h File Reference	103
7.12	include/entropies.h File Reference	104
7.13	include/functions.h File Reference	105
7.14	include/internal.h File Reference	109
7.15	include/io.h File Reference	110
7.16	include/matlab.h File Reference	111
7.17	include/qpp.h File Reference	112
7.18	include/random.h File Reference	113
7.19	include/types.h File Reference	114
	<b>Index</b>	<b>115</b>



# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">qpp</a> . . . . .	9
<a href="#">qpp::internal</a> . . . . .	72



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

qpp::DiscreteDistribution . . . . .	75
qpp::DiscreteDistributionAbsSquare . . . . .	76
exception	
qpp::Exception . . . . .	77
qpp::NormalDistribution . . . . .	86
qpp::Qudit . . . . .	87
qpp::Singleton< T > . . . . .	90
qpp::Gates . . . . .	80
qpp::RandomDevices . . . . .	89
qpp::Singleton< const Gates > . . . . .	90
qpp::Singleton< const States > . . . . .	90
qpp::States . . . . .	91
qpp::Singleton< RandomDevices > . . . . .	90
qpp::Timer . . . . .	94
qpp::UniformIntDistribution . . . . .	95
qpp::UniformRealDistribution . . . . .	95





## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">qpp::DiscreteDistribution</a>	75
<a href="#">qpp::DiscreteDistributionAbsSquare</a>	76
<a href="#">qpp::Exception</a>	77
<a href="#">qpp::Gates</a>	80
<a href="#">qpp::NormalDistribution</a>	86
<a href="#">qpp::Qudit</a>	87
<a href="#">qpp::RandomDevices</a>	89
<a href="#">qpp::Singleton&lt; T &gt;</a>	90
<a href="#">qpp::States</a>	91
<a href="#">qpp::Timer</a>	94
<a href="#">qpp::UniformIntDistribution</a>	95
<a href="#">qpp::UniformRealDistribution</a>	95



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

include/channels.h	97
include/constants.h	102
include/entanglement.h	103
include/entropies.h	104
include/functions.h	105
include/internal.h	109
include/io.h	110
include/matlab.h	111
include/qpp.h	112
include/random.h	113
include/types.h	114
include/classes/exception.h	98
include/classes/gates.h	98
include/classes/qudit.h	99
include/classes/randevs.h	99
include/classes/singleton.h	100
include/classes/stat.h	101
include/classes/states.h	101
include/classes/timer.h	102



## Chapter 5

# Namespace Documentation

### 5.1 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 [UniformIntDistribution](#)
- 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-  $\alpha$  entropy of the probability distribution/density matrix A, for  $\alpha \geq 0$ .*
- `template<typename Derived >`  
`double renyi_inf` (const Eigen::MatrixBase< Derived > &A)  
*Renyi-  $\infty$  entropy (min entropy) of the probability distribution/density matrix A.*
- `template<typename Derived >`  
`double tsallis` (const double alpha, const Eigen::MatrixBase< Derived > &A)  
*Tsallis-  $\alpha$  entropy of the probability distribution/density matrix A, for  $\alpha \geq 0$*

- template<typename Derived >  
double [qmutualinfo](#) (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size\_t > &subsysA, const std::vector< std::size\_t > &subsysB, const std::vector< std::size\_t > &dims)  
*Quantum mutual information between 2 subsystems of a composite system.*
- template<typename Derived >  
[DynMat](#)< typename Derived::Scalar > [transpose](#) (const Eigen::MatrixBase< Derived > &A)  
*Transpose.*
- template<typename Derived >  
[DynMat](#)< typename Derived::Scalar > [conjugate](#) (const Eigen::MatrixBase< Derived > &A)  
*Complex conjugate.*
- template<typename Derived >  
[DynMat](#)< typename Derived::Scalar > [adjoint](#) (const Eigen::MatrixBase< Derived > &A)  
*Adjoint.*
- template<typename Derived >  
[DynMat](#)< typename Derived::Scalar > [inverse](#) (const Eigen::MatrixBase< Derived > &A)  
*Inverse.*
- template<typename Derived >  
Derived::Scalar [trace](#) (const Eigen::MatrixBase< Derived > &A)  
*Trace.*
- template<typename Derived >  
Derived::Scalar [det](#) (const Eigen::MatrixBase< Derived > &A)  
*Determinant.*
- template<typename Derived >  
Derived::Scalar [logdet](#) (const Eigen::MatrixBase< Derived > &A)  
*Logarithm of the determinant.*
- template<typename Derived >  
Derived::Scalar [sum](#) (const Eigen::MatrixBase< Derived > &A)  
*Element-wise sum.*
- 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 absolut value.*

- `template<typename Derived >`  
`cmat expm` (const Eigen::MatrixBase< Derived > &A)  
*Matrix exponential.*
- `template<typename Derived >`  
`cmat logm` (const Eigen::MatrixBase< Derived > &A)  
*Matrix logarithm.*
- `template<typename Derived >`  
`cmat sinm` (const Eigen::MatrixBase< Derived > &A)  
*Matrix sin.*
- `template<typename Derived >`  
`cmat cosm` (const Eigen::MatrixBase< Derived > &A)  
*Matrix cos.*
- `template<typename Derived >`  
`cmat spectralpowm` (const Eigen::MatrixBase< Derived > &A, const `cplx` z)  
*Matrix power.*
- `template<typename Derived >`  
`DynMat< typename Derived::Scalar > powm` (const Eigen::MatrixBase< Derived > &A, std::size\_t n)  
*Matrix power.*
- `template<typename 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 dispn (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 dispn (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 dispn (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)`

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

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

- `double rand (double a=0, double b=1)`
- `long long randint (long long a, long long b)`

- `template<typename Derived >`  
Derived `randn` (`std::size_t` rows, `std::size_t` cols, double mean=0, double sigma=1)
- `template<>`  
`dmat randn` (`std::size_t` rows, `std::size_t` cols, double mean, double sigma)
- `template<>`  
`cmat randn` (`std::size_t` rows, `std::size_t` cols, double mean, double sigma)
- double `randn` (double mean=0, double sigma=1)
- `cmat randU` (`std::size_t` D)
- `cmat randV` (`std::size_t` Din, `std::size_t` Dout)
- `std::vector< cmat > randkraus` (`std::size_t` n, `std::size_t` D)
- `cmat randH` (`std::size_t` D)
- `ket randket` (`std::size_t` D)
- `cmat randrho` (`std::size_t` D)
- `std::vector< std::size_t > randperm` (`std::size_t` n)

## Variables

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

### 5.1.1 Typedef Documentation

#### 5.1.1.1 `using qpp::bra = typedef Eigen::Matrix<cplx, 1, Eigen::Dynamic>`

Complex (double precision) dynamic Eigen row matrix.

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

Complex (double precision) dynamic Eigen matrix.

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

Complex number in double precision.

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

Real (double precision) dynamic Eigen matrix.

5.1.1.5 `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>
```

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

Complex (double precision) dynamic Eigen column matrix.

## 5.1.2 Function Documentation

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

Matrix absolut value.

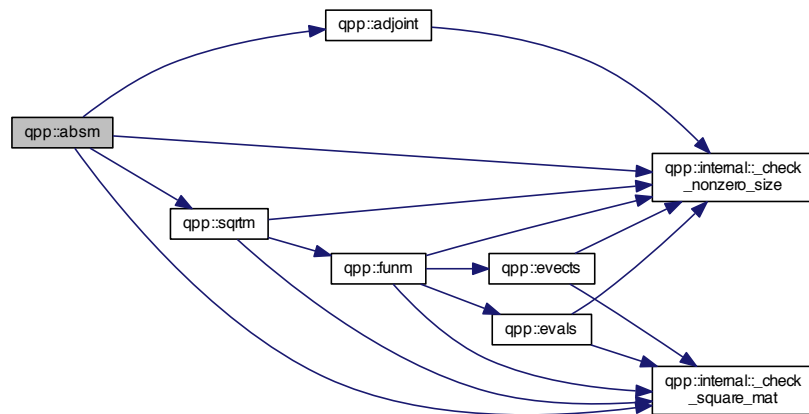
Parameters

<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:



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

Adjoint.

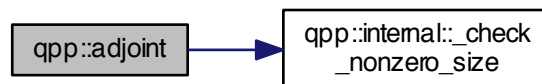
## Parameters

$A$	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:



5.1.2.3 `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

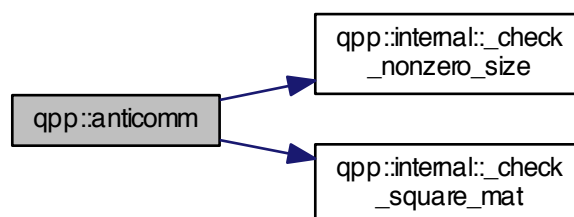
## Parameters

$A$	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:



5.1.2.4 `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 *rho*.

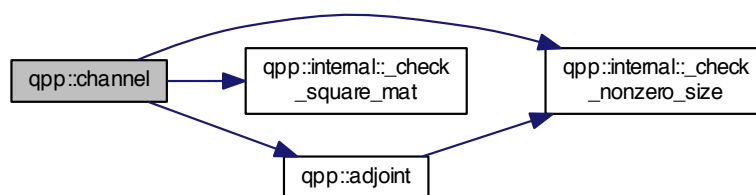
## Parameters

<i>rho</i>	Eigen expression
<i>Ks</i>	std::vector 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:



**5.1.2.5** `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*.

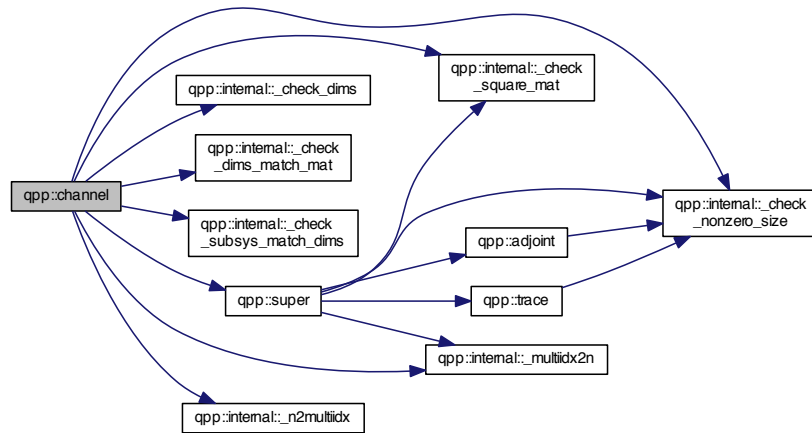
## Parameters

<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:



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

**Note**

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

**Parameters**

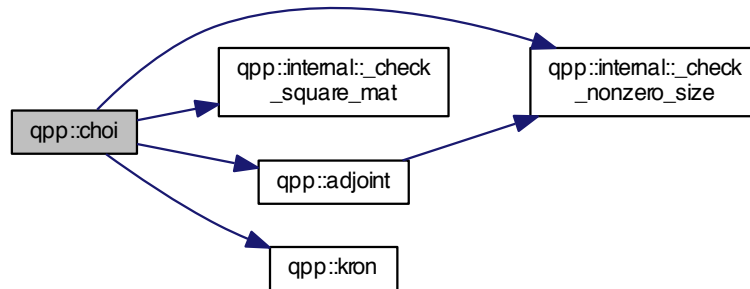
$K_s$	std::vector of Eigen expressions representing the set of Kraus operators
-------	--



## Returns

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

Here is the call graph for this function:



#### 5.1.2.7 `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  $i \neq j$

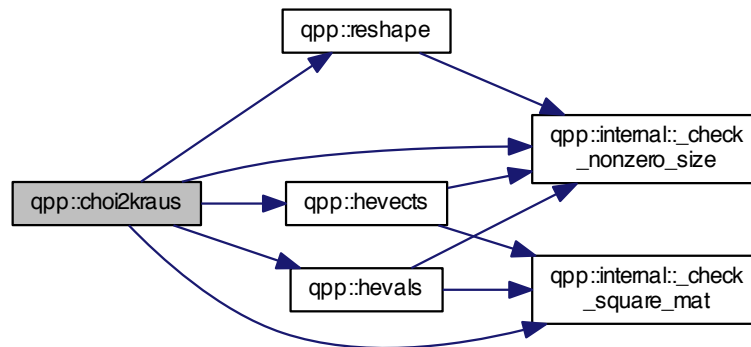
## Parameters

$A$	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:



**5.1.2.8** `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

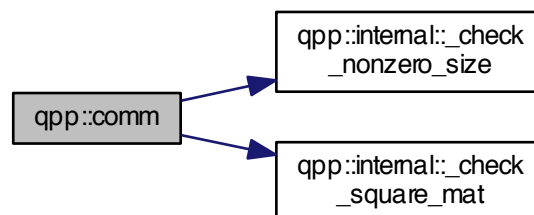
**Parameters**

$A$	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:



5.1.2.9 `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:



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

Complex conjugate.

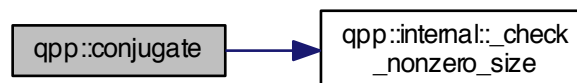
## Parameters

<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:



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

Matrix cos.

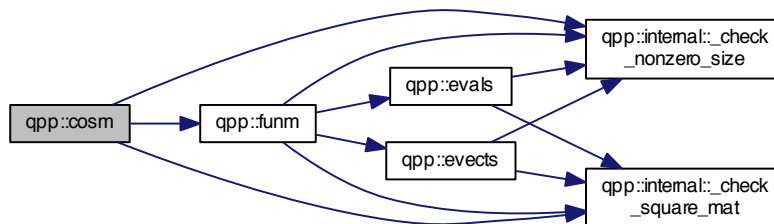
## Parameters

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

## Returns

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

Here is the call graph for this function:



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

Functor.

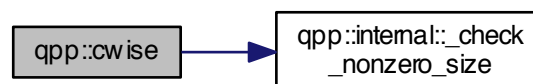
## Parameters

$A$	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:



**5.1.2.13** `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:



5.1.2.14 `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

5.1.2.15 `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

**5.1.2.16** `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

**5.1.2.17** `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:



**5.1.2.18** `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:



**5.1.2.19** `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:



**5.1.2.20** `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:



5.1.2.21 `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:



5.1.2.22 `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\(\)](#)

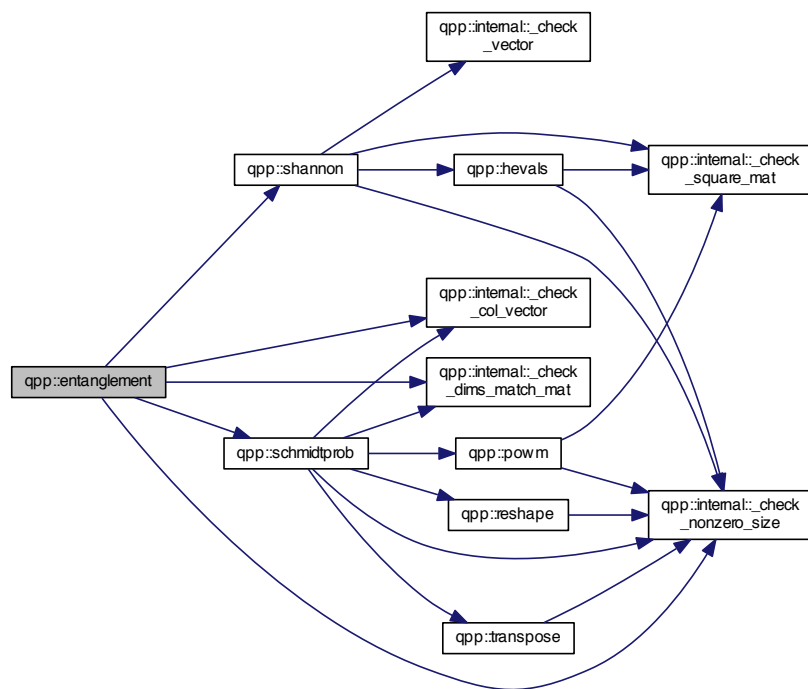
## Parameters

<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:



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

Eigenvalues.

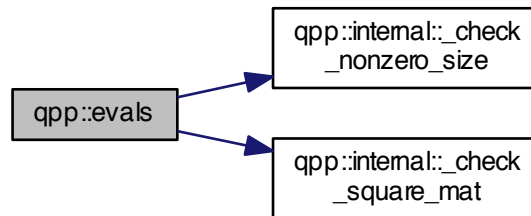
## Parameters

<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:



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

Eigenvectors.

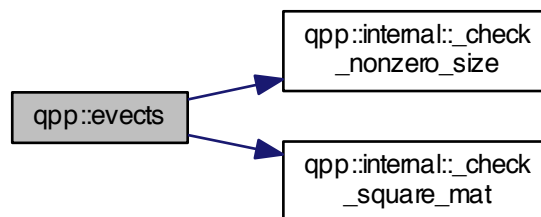
## Parameters

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

## Returns

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

Here is the call graph for this function:



5.1.2.25 `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)`

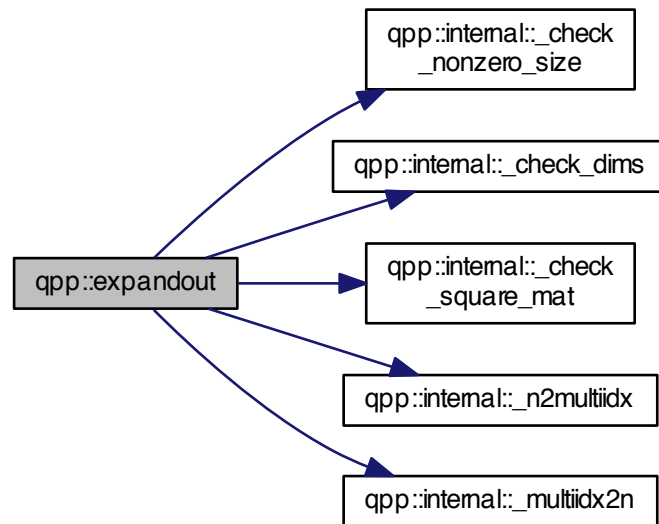
## Parameters

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

## Returns

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

Here is the call graph for this function:



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

Matrix exponential.

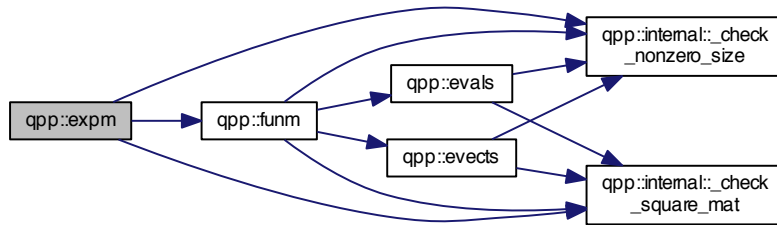
## Parameters

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

## Returns

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

Here is the call graph for this function:



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

Functional calculus  $f(A)$

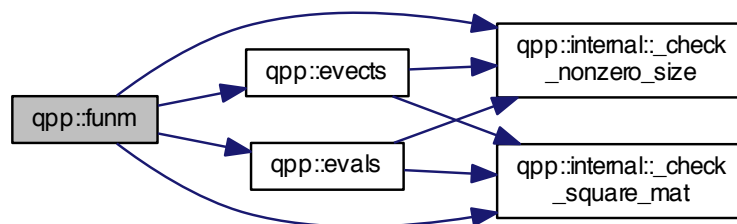
## Parameters

$A$	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:



5.1.2.28 `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\(\)](#)

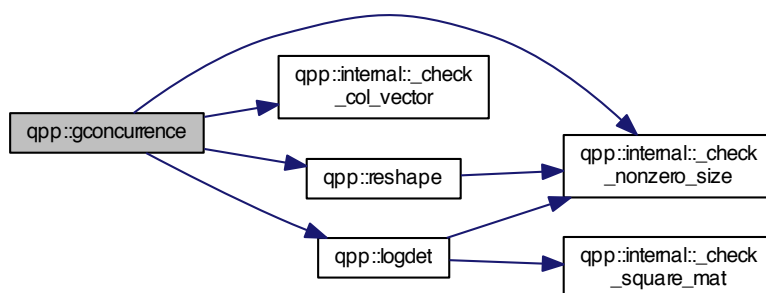
Parameters

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

Returns

G-concurrence

Here is the call graph for this function:



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

Gram-Schmidt orthogonalization (std::vector overload)

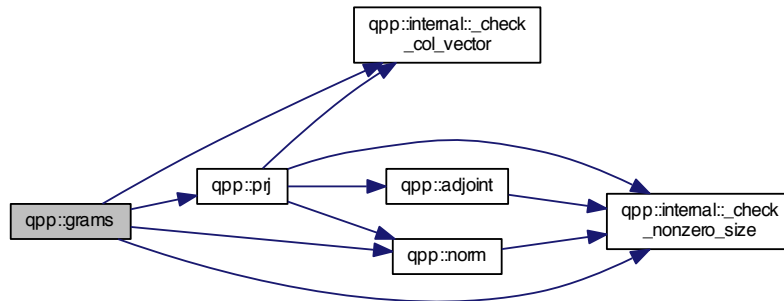
Parameters

<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:



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

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

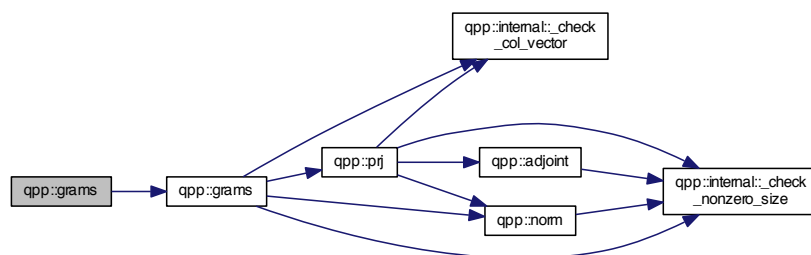
## Parameters

$V$ 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:



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

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

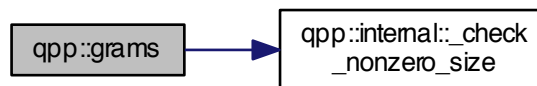
## Parameters

$A$	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:



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

Hermitian eigenvalues.

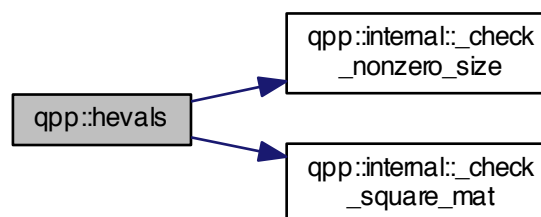
## Parameters

$A$	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:



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

Hermitian eigenvectors.



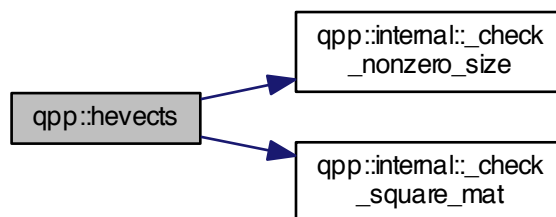
## Parameters

$A$	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:



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

Inverse.

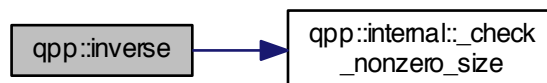
## Parameters

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

## Returns

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

Here is the call graph for this function:



5.1.2.35 `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:



### 5.1.2.36 `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*

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

Kronecker product (variadic overload)

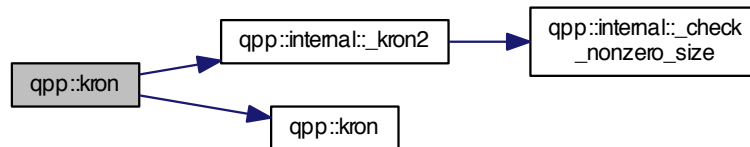
## Parameters

<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:



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

Kronecker product (std::vector overload)

**Parameters**

<b>As</b>	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:



**5.1.2.39** `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:



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

Kronecker power.

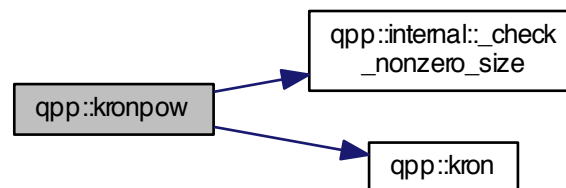
**Parameters**

<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:



**5.1.2.41** `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

**5.1.2.42** `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)

**5.1.2.43** `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](#))

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](#))

**5.1.2.44** `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](#))

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](#))

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

Logarithm of the determinant.

Especially useful when the determinant overflows/underflows

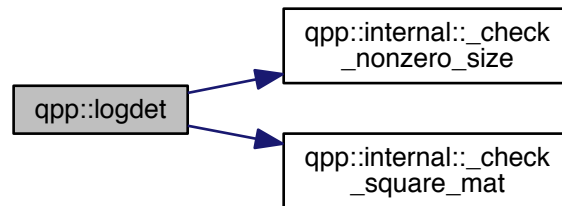
## Parameters

$A$	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:



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

Matrix logarithm.

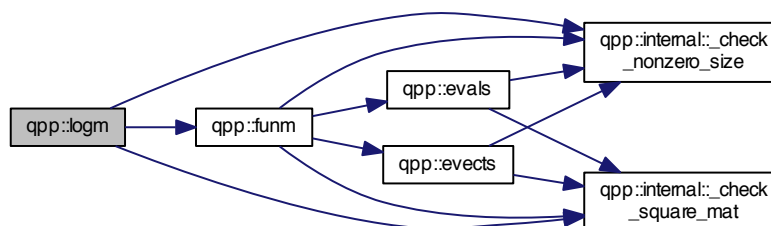
## Parameters

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

## Returns

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

Here is the call graph for this function:



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

Multi-partite qubit ket.

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

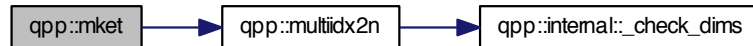
## Parameters

<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:



#### 5.1.2.48 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*.

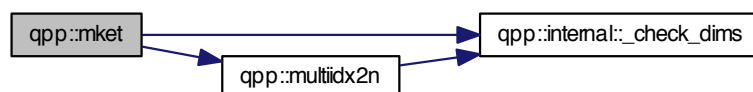
## Parameters

<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:



#### 5.1.2.49 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*.

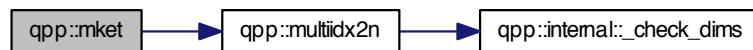
## Parameters

<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:



5.1.2.50 `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:



5.1.2.51 `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:



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

Trace norm.

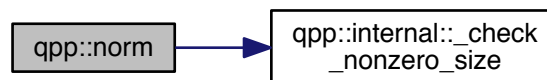
## Parameters

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

## Returns

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

Here is the call graph for this function:



### 5.1.2.53 `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  $\exp(2\pi i/D)$

#### 5.1.2.54 `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>
```

#### 5.1.2.55 `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>
```

#### 5.1.2.56 `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  $A^0 = I$

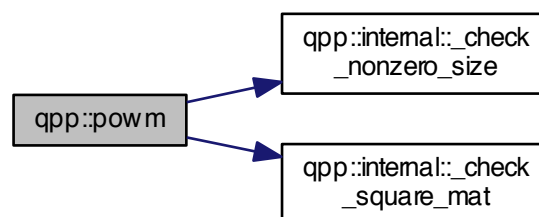
**Parameters**

$A$	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:



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

Projector.

Normalized projector onto state vector

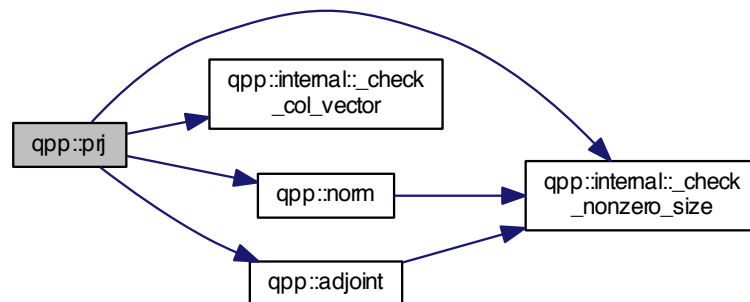
Parameters

$V$	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:



5.1.2.58 `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

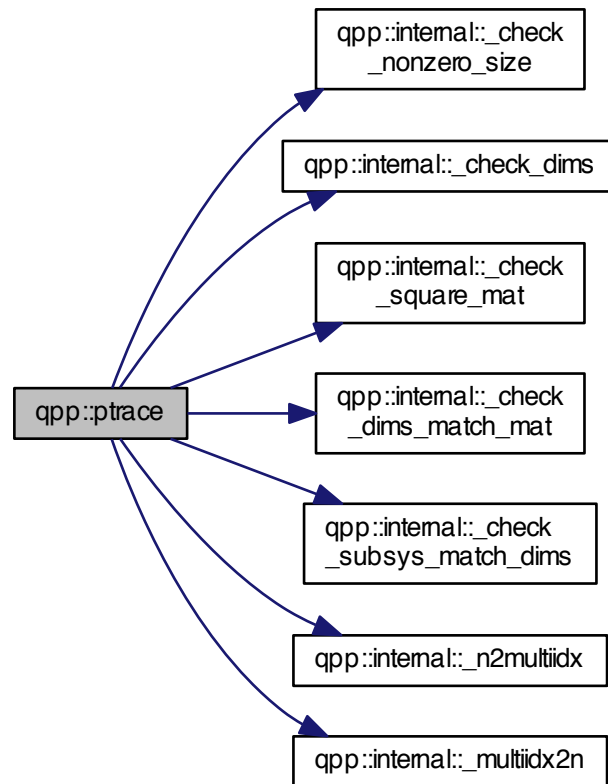
Parameters

$A$	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:



5.1.2.59 `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

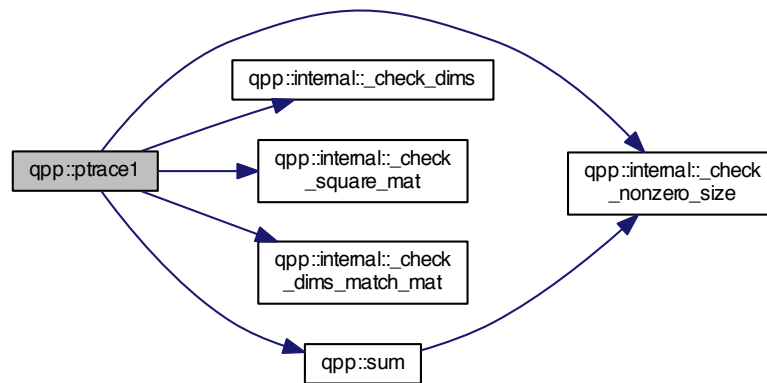
## Parameters

<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:



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

Partial trace.

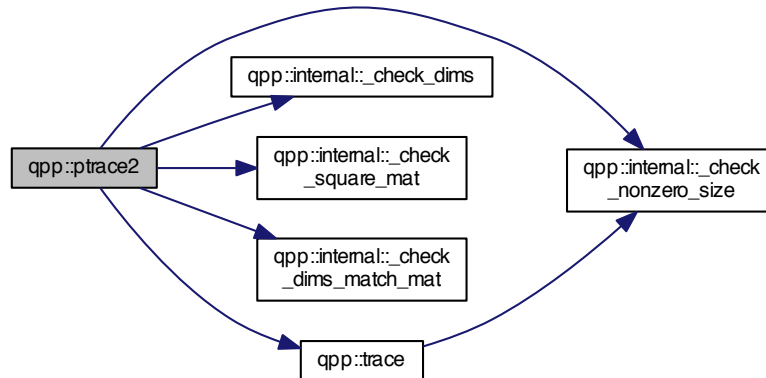
## Parameters

$A$	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:



**5.1.2.61** `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

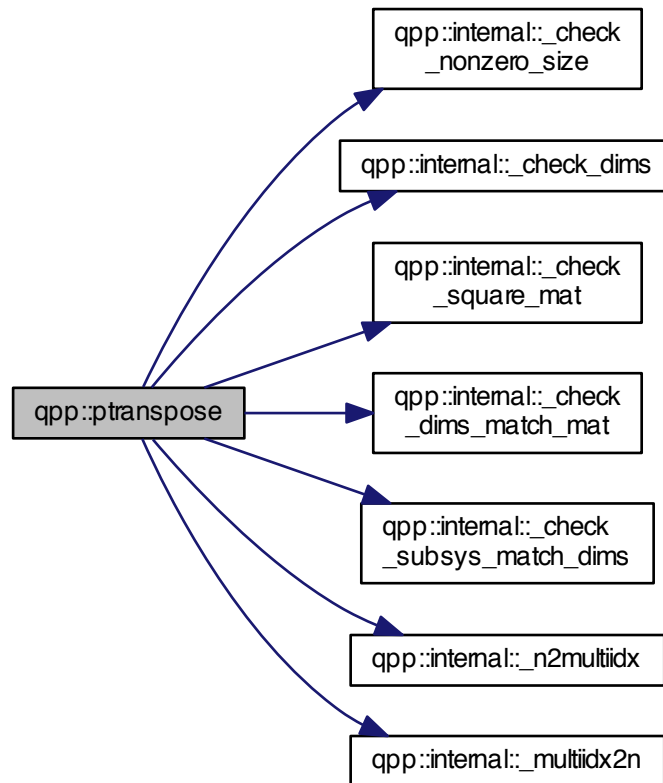
**Parameters**

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

**Returns**

Partial transpose  $(\cdot)^{T_{subsys}}$  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:



5.1.2.62 `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.

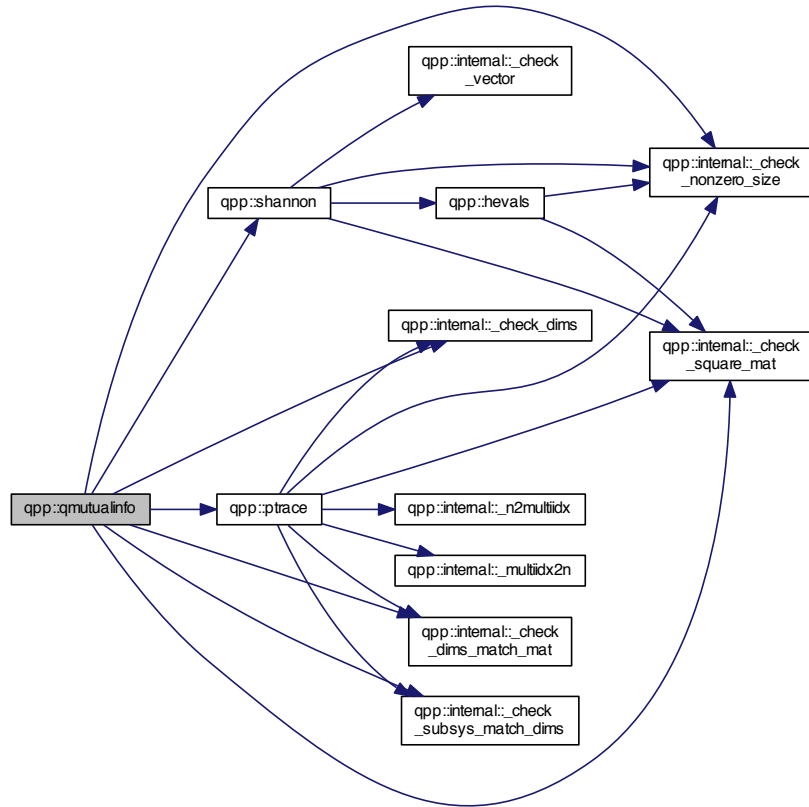
Parameters

<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:



5.1.2.63 `template<typename Derived> Derived qpp::rand ( std::size_t rows, std::size_t cols, double a = 0, double b = 1 )`

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

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

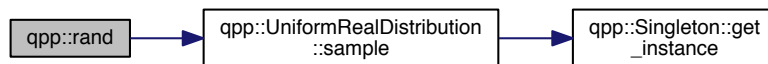
Here is the call graph for this function:





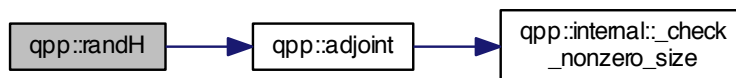
### 5.1.2.66 `double qpp::rand ( double a = 0, double b = 1 )`

Here is the call graph for this function:



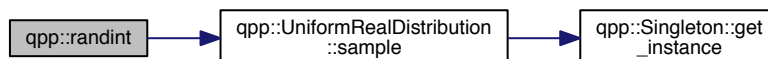
### 5.1.2.67 `cmat qpp::randH ( std::size_t D )`

Here is the call graph for this function:



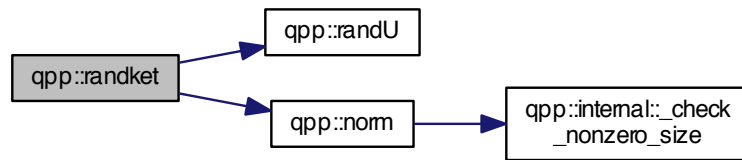
### 5.1.2.68 `long long qpp::randint ( long long a, long long b )`

Here is the call graph for this function:



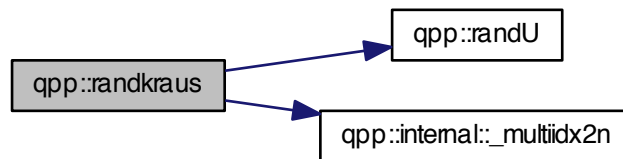
### 5.1.2.69 ket qpp::randket ( std::size\_t *D* )

Here is the call graph for this function:



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

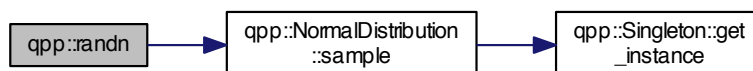
Here is the call graph for this function:



### 5.1.2.71 template<typename Derived > Derived qpp::randn ( std::size\_t *rows*, std::size\_t *cols*, double *mean* = 0, double *sigma* = 1 )

### 5.1.2.72 template<> dmat qpp::randn ( std::size\_t *rows*, std::size\_t *cols*, double *mean*, double *sigma* )

Here is the call graph for this function:



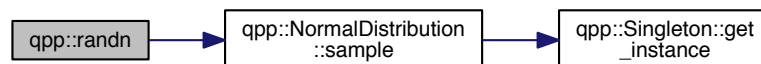
5.1.2.73 `template<> cmat qpp::randn ( std::size_t rows, std::size_t cols, double mean, double sigma )`

Here is the call graph for this function:



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

Here is the call graph for this function:



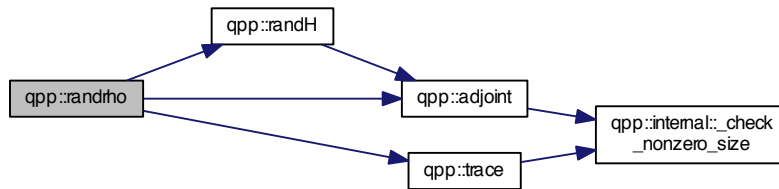
5.1.2.75 `std::vector<std::size_t> qpp::randperm ( std::size_t n )`

Here is the call graph for this function:



#### 5.1.2.76 `cmat qpp::randrho ( std::size_t D )`

Here is the call graph for this function:



#### 5.1.2.77 `cmat qpp::randU ( std::size_t D )`

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

Here is the call graph for this function:



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

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

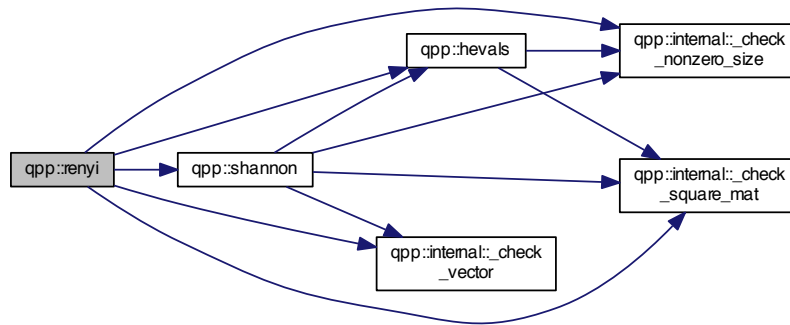
Parameters

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

## Returns

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

Here is the call graph for this function:



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

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

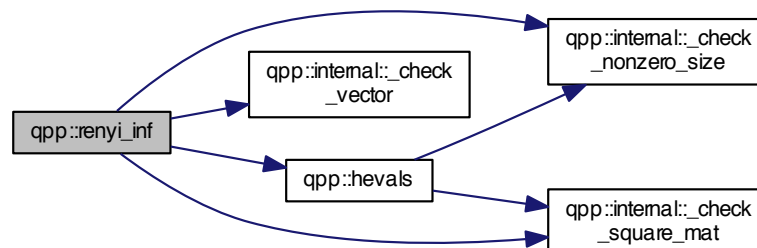
## Parameters

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

## Returns

Renyi-  $\infty$  entropy (min entropy), with the logarithm in base 2

Here is the call graph for this function:



#### 5.1.2.81 `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)

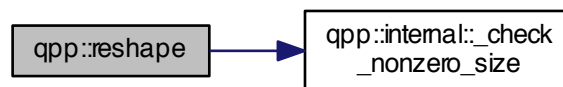
## Parameters

<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:



5.1.2.82 `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

5.1.2.83 `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)

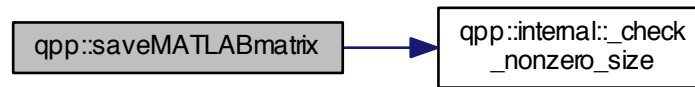
5.1.2.84 `template<> void qpp::saveMATLABmatrix ( const Eigen::MatrixBase< dmat > & A, const std::string & mat_file, const std::string & var_name, const std::string & mode )`

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:



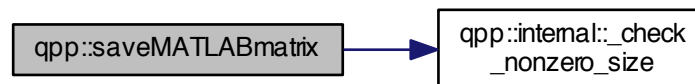
**5.1.2.85** `template<> void qpp::saveMATLABmatrix ( const Eigen::MatrixBase< cmat > & A, const std::string & mat_file, const std::string & var_name, const std::string & mode )`

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:



**5.1.2.86** `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\(\)](#)



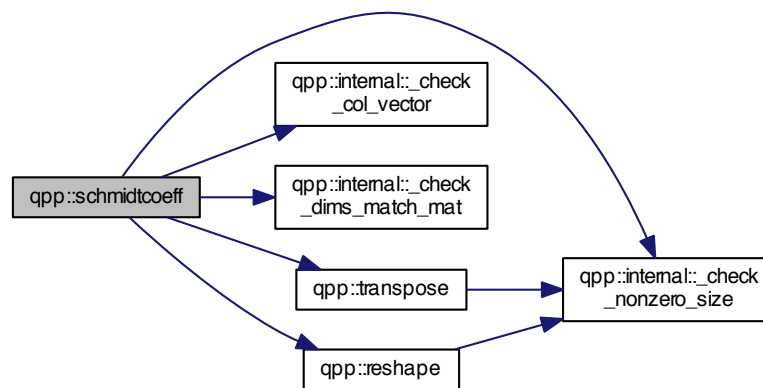
## Parameters

<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:



5.1.2.87 `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\(\)`](#)

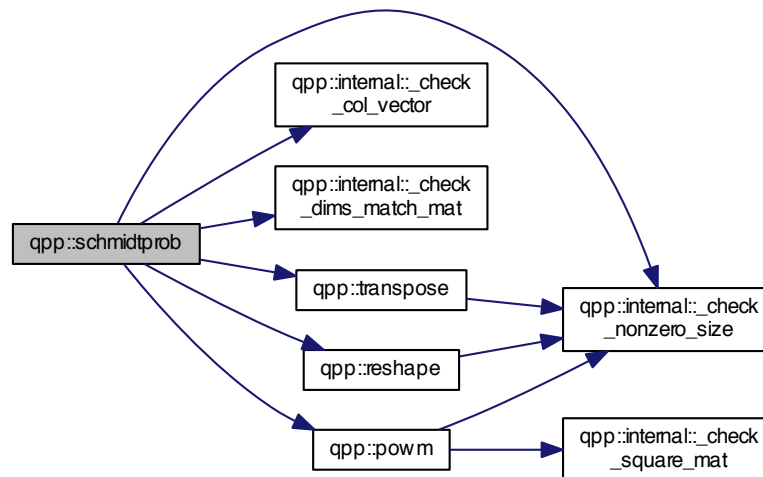
## Parameters

<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:



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

Schmidt basis on Alice's side.

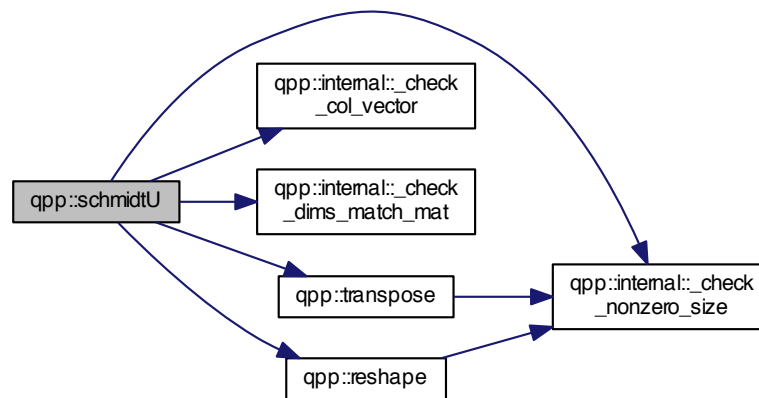
**Parameters**

$A$	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:



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

Schmidt basis on Bob's side.

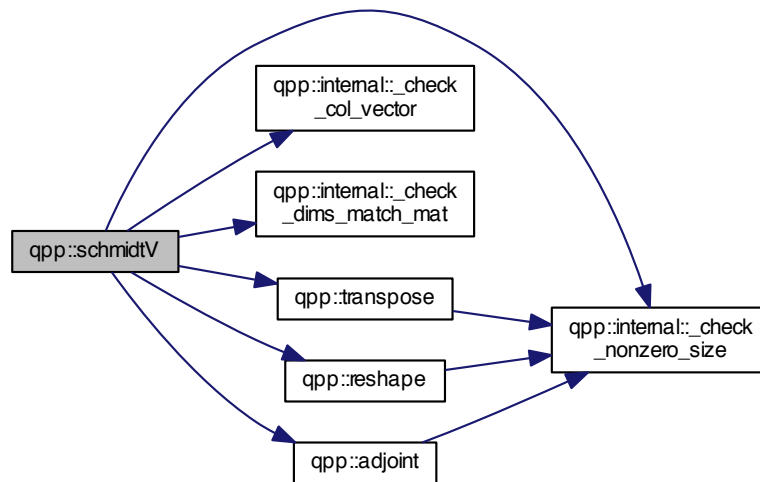
## Parameters

<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:



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

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

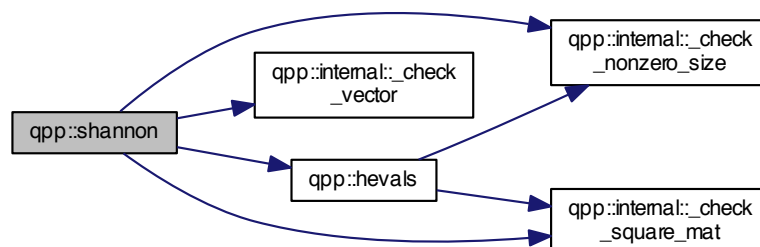
##### Parameters

$A$	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:



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

Matrix sin.

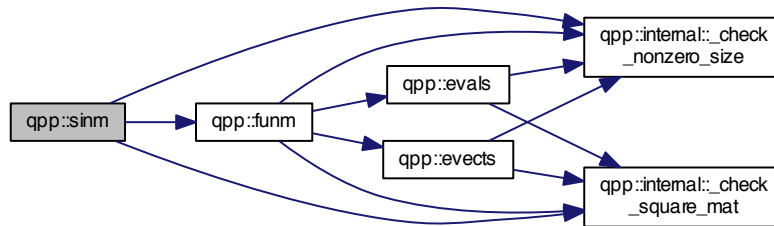
## Parameters

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

## Returns

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

Here is the call graph for this function:



5.1.2.92 `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  $A^0 = I$

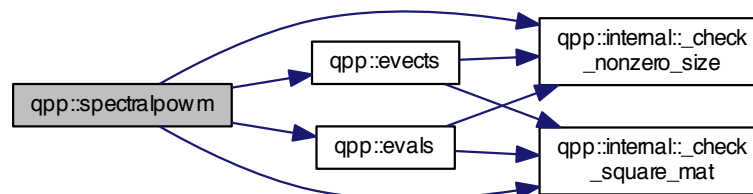
## Parameters

$A$	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:



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

Matrix square root.

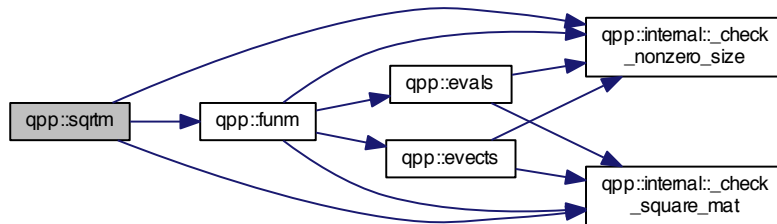
## Parameters

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

## Returns

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

Here is the call graph for this function:



#### 5.1.2.94 `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:



#### 5.1.2.95 `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.

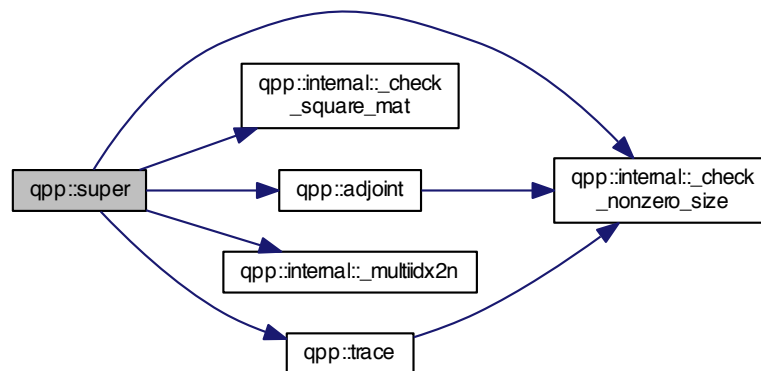
## Parameters

<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:



5.1.2.96 `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

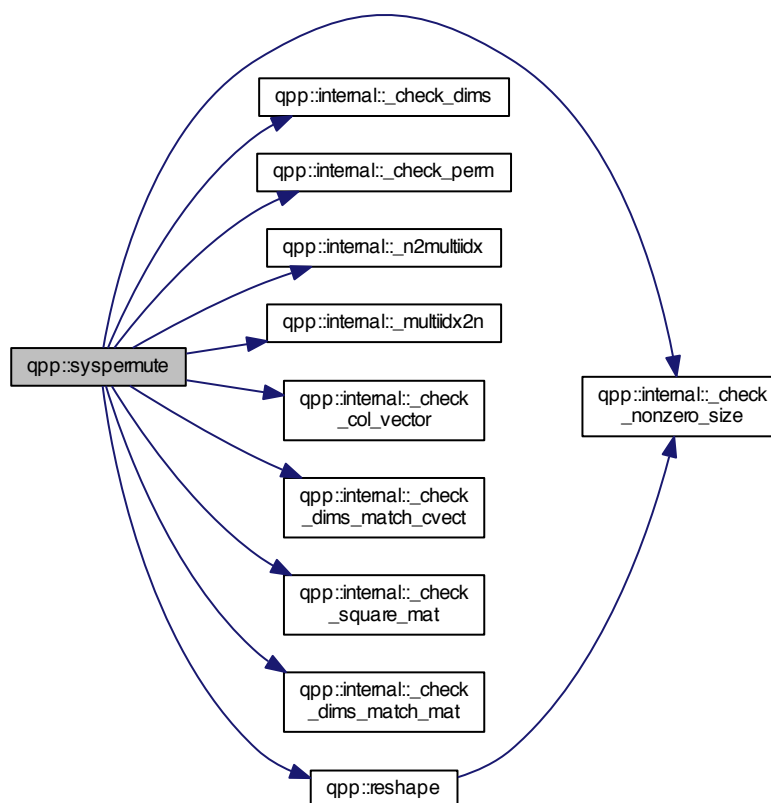
<i>A</i>	Eigen expression
<i>perm</i>	Permutation
<i>dims</i>	Subsystems' dimensions



## Returns

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

Here is the call graph for this function:



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

Trace.

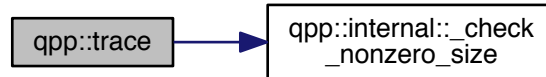
## Parameters

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

**Returns**

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

Here is the call graph for this function:



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

Transpose.

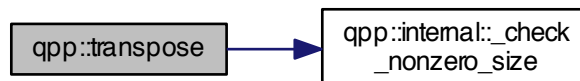
**Parameters**

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

**Returns**

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

Here is the call graph for this function:



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

Tsallis-  $\alpha$  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**

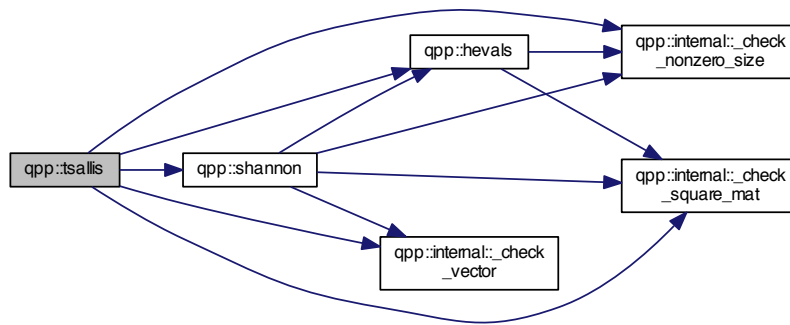
<i>alpha</i>	Non-negative real number
--------------	--------------------------

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

#### Returns

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

Here is the call graph for this function:



### 5.1.3 Variable Documentation

#### 5.1.3.1 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  $qpp::ct::chop$ .

#### 5.1.3.2 constexpr double qpp::ee = 2.718281828459045235360287471352662497

Base of natural logarithm,  $e$ .

#### 5.1.3.3 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
```

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

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

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

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

Maximum number of qubits.

Used internally to statically allocate arrays (for speed reasons)

5.1.3.6 constexpr double qpp::pi = 3.141592653589793238462643383279502884

$\pi$

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

[qpp::RandomDevices](#) Singleton

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

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

[qpp::States](#) const Singleton

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

## 5.2 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)

### 5.2.1 Detailed Description

Internal functions, do not modify or use them directly

### 5.2.2 Function Documentation

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

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

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

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

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

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

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

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

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

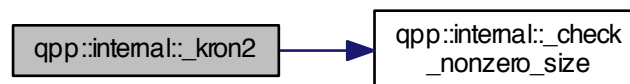
5.2.2.10 `template<typename Derived > bool qpp::internal::_check_square_mat ( const Eigen::MatrixBase< Derived > & A )`

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

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

5.2.2.13 `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:



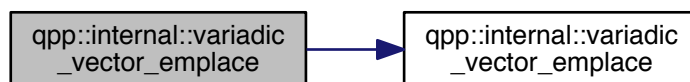
5.2.2.14 `std::size_t qpp::internal::_multiidx2n ( const std::size_t * midx, std::size_t numdims, const std::size_t * dims )`

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

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

5.2.2.17 `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 6

# Class Documentation

### 6.1 qpp::DiscreteDistribution Class 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)`
- `std::size_t sample ()`
- `std::vector< double > probabilities () const`

#### Protected Attributes

- `std::discrete_distribution  
< std::size_t > \_d`

#### 6.1.1 Constructor & Destructor Documentation

6.1.1.1 `template<typename InputIterator > qpp::DiscreteDistribution::DiscreteDistribution ( InputIterator first, InputIterator last ) [inline]`

6.1.1.2 `qpp::DiscreteDistribution::DiscreteDistribution ( std::initializer_list< double > weights ) [inline]`

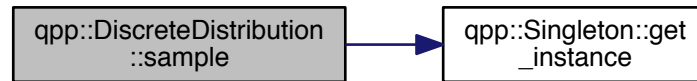
6.1.1.3 `qpp::DiscreteDistribution::DiscreteDistribution ( std::vector< double > weights ) [inline]`

#### 6.1.2 Member Function Documentation

6.1.2.1 `std::vector<double> qpp::DiscreteDistribution::probabilities ( ) const [inline]`

### 6.1.2.2 `std::size_t qpp::DiscreteDistribution::sample ( )` `[inline]`

Here is the call graph for this function:



## 6.1.3 Member Data Documentation

### 6.1.3.1 `std::discrete_distribution<std::size_t> qpp::DiscreteDistribution::_d` `[protected]`

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

- `include/classes/stat.h`

## 6.2 `qpp::DiscreteDistributionAbsSquare` Class 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)
- `std::size_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`  
`< std::size_t > _d`



### 6.2.1 Constructor & Destructor Documentation

6.2.1.1 `template<typename InputIterator > qpp::DiscreteDistributionAbsSquare::DiscreteDistributionAbsSquare ( InputIterator first, InputIterator last ) [inline]`

6.2.1.2 `qpp::DiscreteDistributionAbsSquare::DiscreteDistributionAbsSquare ( std::initializer_list< cplx > amplitudes ) [inline]`

6.2.1.3 `qpp::DiscreteDistributionAbsSquare::DiscreteDistributionAbsSquare ( std::vector< cplx > amplitudes ) [inline]`

6.2.1.4 `template<typename Derived > qpp::DiscreteDistributionAbsSquare::DiscreteDistributionAbsSquare ( const Eigen::MatrixBase< Derived > & V ) [inline]`

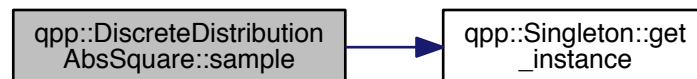
### 6.2.2 Member Function Documentation

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

6.2.2.2 `std::vector<double> qpp::DiscreteDistributionAbsSquare::probabilities ( ) const [inline]`

6.2.2.3 `std::size_t qpp::DiscreteDistributionAbsSquare::sample ( ) [inline]`

Here is the call graph for this function:



### 6.2.3 Member Data Documentation

6.2.3.1 `std::discrete_distribution<std::size_t> qpp::DiscreteDistributionAbsSquare::_d [protected]`

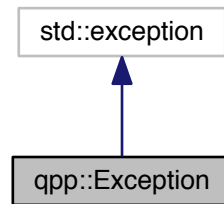
The documentation for this class was generated from the following file:

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

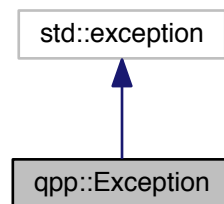
## 6.3 qpp::Exception Class Reference

```
#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`

### 6.3.1 Member Enumeration Documentation

#### 6.3.1.1 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

### 6.3.2 Constructor & Destructor Documentation

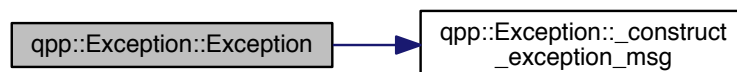
6.3.2.1 `qpp::Exception::Exception ( const std::string & where, const Type & type )` `[inline]`

Here is the call graph for this function:



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

Here is the call graph for this function:



### 6.3.3 Member Function Documentation

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

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

### 6.3.4 Member Data Documentation

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

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

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

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

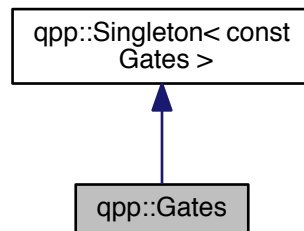
The documentation for this class was generated from the following file:

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

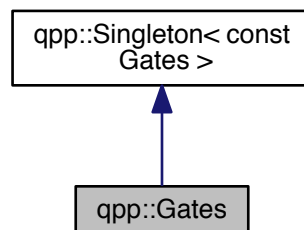
## 6.4 qpp::Gates Class Reference

```
#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

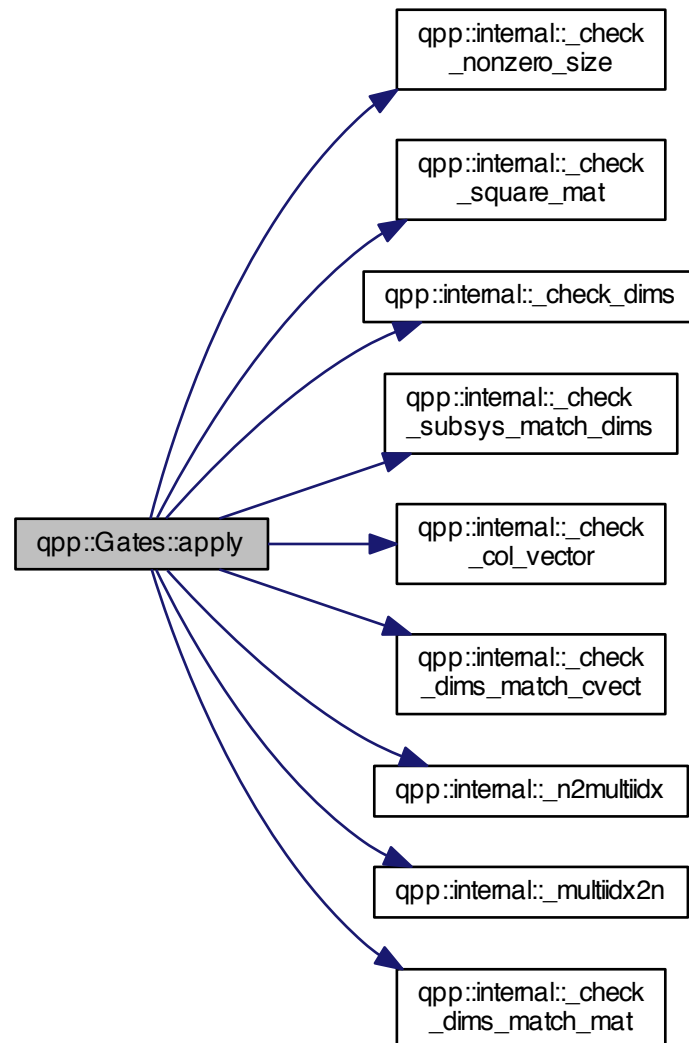
### 6.4.1 Constructor & Destructor Documentation

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

### 6.4.2 Member Function Documentation

6.4.2.1 `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]`

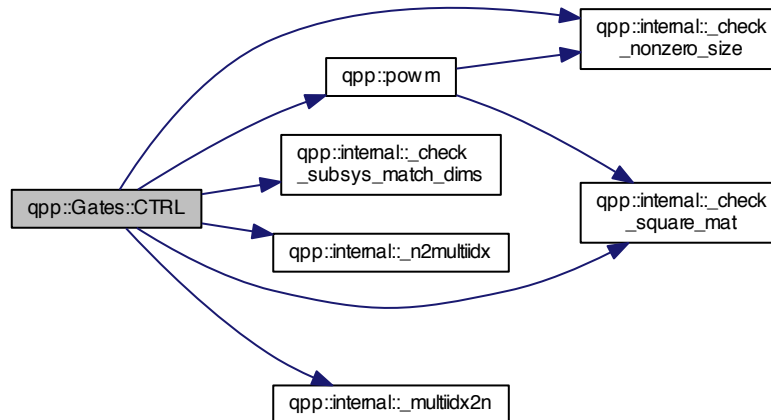
Here is the call graph for this function:



6.4.2.2 `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]`

6.4.2.3 `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:



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

Here is the call graph for this function:



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

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



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

Here is the call graph for this function:



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

Here is the call graph for this function:



### 6.4.3 Friends And Related Function Documentation

#### 6.4.3.1 `friend class Singleton< const Gates > [friend]`

### 6.4.4 Member Data Documentation

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 6.5 qpp::NormalDistribution Class Reference

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

### Public Member Functions

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

### Protected Attributes

- `std::normal_distribution _d`

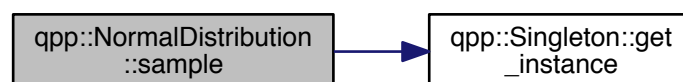
#### 6.5.1 Constructor & Destructor Documentation

6.5.1.1 `qpp::NormalDistribution::NormalDistribution ( double mean = 0, double sigma = 1 )` `[inline]`

#### 6.5.2 Member Function Documentation

6.5.2.1 `double qpp::NormalDistribution::sample ( )` `[inline]`

Here is the call graph for this function:



#### 6.5.3 Member Data Documentation

6.5.3.1 `std::normal_distribution qpp::NormalDistribution::_d` `[protected]`

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

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

## 6.6 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

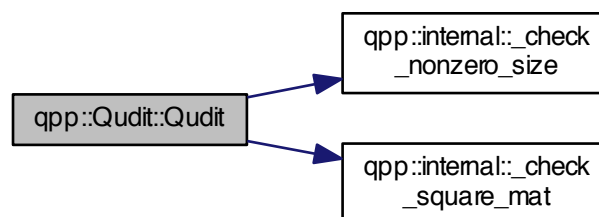
### Private Attributes

- [cmat \\_rho](#)
- [std::size\\_t \\_D](#)

### 6.6.1 Constructor & Destructor Documentation

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

Here is the call graph for this function:



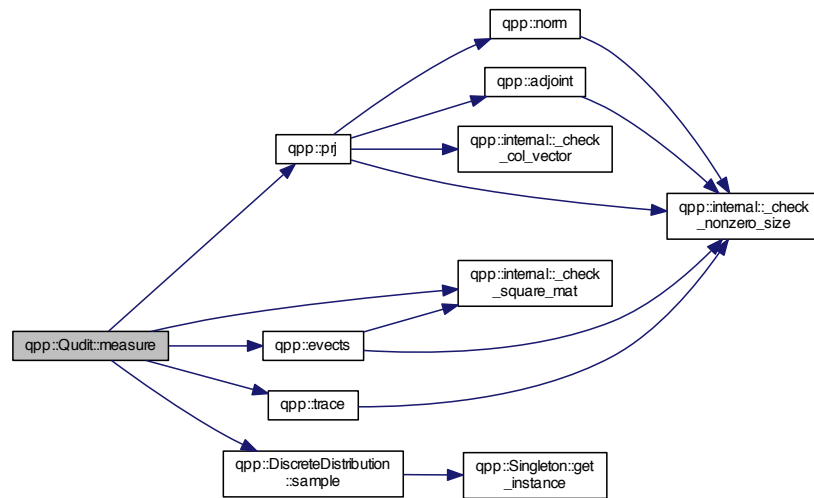
### 6.6.2 Member Function Documentation

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

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

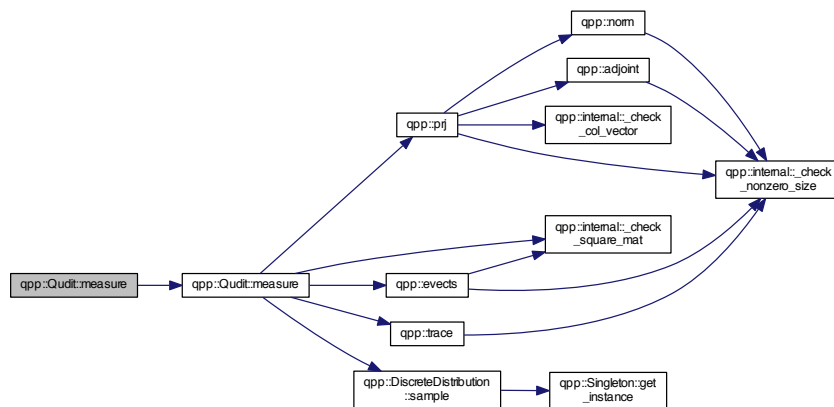
### 6.6.2.3 `std::size_t qpp::Qudit::measure ( const cmat & U, bool destructive = false ) [inline]`

Here is the call graph for this function:



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

Here is the call graph for this function:



## 6.6.3 Member Data Documentation

### 6.6.3.1 `std::size_t qpp::Qudit::_D [private]`

### 6.6.3.2 `cmat qpp::Qudit::_rho [private]`

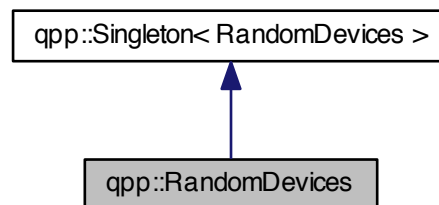
The documentation for this class was generated from the following file:

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

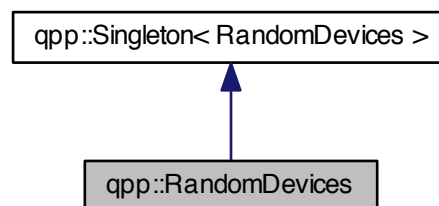
## 6.7 qpp::RandomDevices Class Reference

```
#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

### 6.7.1 Constructor & Destructor Documentation

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

### 6.7.2 Friends And Related Function Documentation

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

### 6.7.3 Member Data Documentation

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

6.7.3.2 `std::mt19937 qpp::RandomDevices::_rng`

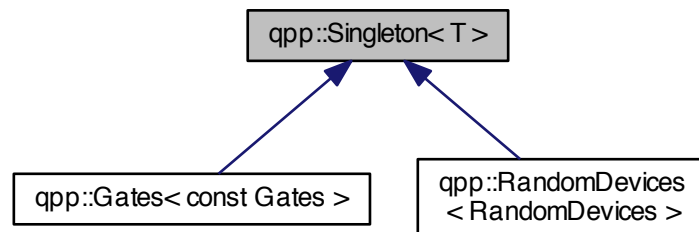
The documentation for this class was generated from the following file:

- `include/classes/randevs.h`

## 6.8 `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`

### 6.8.1 Constructor & Destructor Documentation

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

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

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

### 6.8.2 Member Function Documentation

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

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

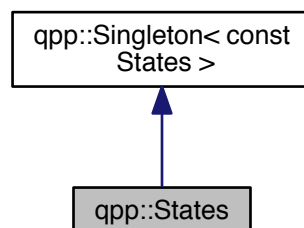
The documentation for this class was generated from the following file:

- `include/classes/singleton.h`

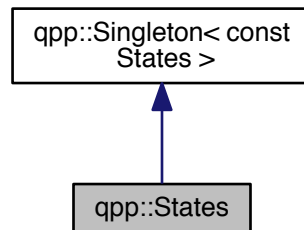
## 6.9 qpp::States Class Reference

```
#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

### 6.9.1 Constructor & Destructor Documentation

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

### 6.9.2 Friends And Related Function Documentation

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

### 6.9.3 Member Data Documentation

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 6.10 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)`

### 6.10.1 Constructor & Destructor Documentation

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

### 6.10.2 Member Function Documentation

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

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

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

### 6.10.3 Friends And Related Function Documentation

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

### 6.10.4 Member Data Documentation

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

6.10.4.2 `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](#)

## 6.11 qpp::UniformIntDistribution Class Reference

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

### Public Member Functions

- [UniformIntDistribution](#) (int a=0, int b=1)
- int [sample](#) ()

### Protected Attributes

- std::uniform\_int\_distribution [\\_d](#)

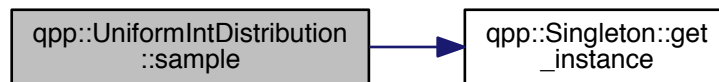
#### 6.11.1 Constructor & Destructor Documentation

6.11.1.1 `qpp::UniformIntDistribution::UniformIntDistribution ( int a = 0, int b = 1 )` `[inline]`

#### 6.11.2 Member Function Documentation

6.11.2.1 `int qpp::UniformIntDistribution::sample ( )` `[inline]`

Here is the call graph for this function:



#### 6.11.3 Member Data Documentation

6.11.3.1 `std::uniform_int_distribution qpp::UniformIntDistribution::_d` `[protected]`

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

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

## 6.12 qpp::UniformRealDistribution Class Reference

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

### Public Member Functions

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

## Protected Attributes

- `std::uniform_real_distribution _d`

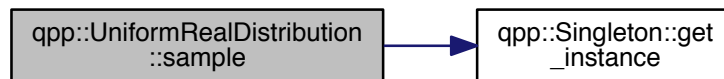
## 6.12.1 Constructor & Destructor Documentation

6.12.1.1 `qpp::UniformRealDistribution::UniformRealDistribution ( double a = 0, double b = 1 )` `[inline]`

## 6.12.2 Member Function Documentation

6.12.2.1 `double qpp::UniformRealDistribution::sample ( )` `[inline]`

Here is the call graph for this function:



## 6.12.3 Member Data Documentation

6.12.3.1 `std::uniform_real_distribution qpp::UniformRealDistribution::_d` `[protected]`

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

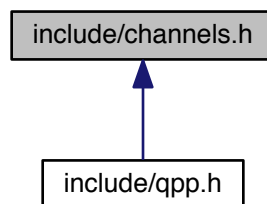
- `include/classes/stat.h`

## Chapter 7

# File Documentation

### 7.1 include/channels.h File Reference

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



### Namespaces

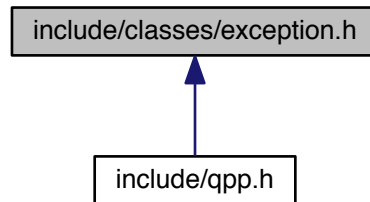
- [qpp](#)

### 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.*

## 7.2 include/classes/exception.h File Reference

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



### Classes

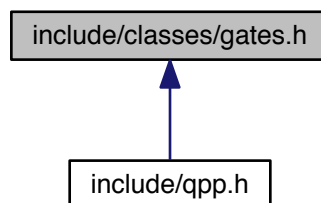
- class [qpp::Exception](#)

### Namespaces

- [qpp](#)

## 7.3 include/classes/gates.h File Reference

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



### Classes

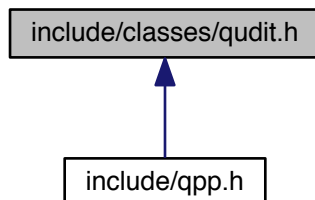
- class [qpp::Gates](#)

### Namespaces

- [qpp](#)

## 7.4 include/classes/qudit.h File Reference

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



### Classes

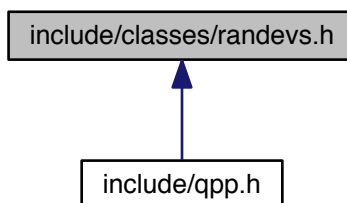
- class [qpp::Qudit](#)

### Namespaces

- [qpp](#)

## 7.5 include/classes/randevs.h File Reference

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



### Classes

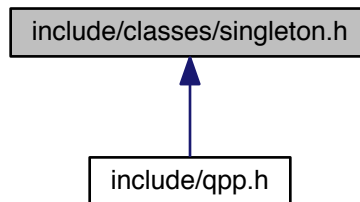
- class [qpp::RandomDevices](#)

### Namespaces

- [qpp](#)

## 7.6 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\(Foo\)](#)

#### 7.6.1 Macro Definition Documentation

##### 7.6.1.1 `#define CLASS_CONST_SINGLETON( Foo )`

###### Value:

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

##### 7.6.1.2 `#define CLASS_SINGLETON( Foo )`

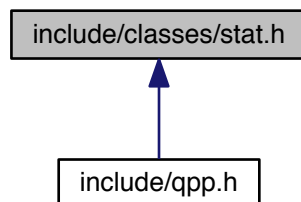
###### Value:

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



## 7.7 include/classes/stat.h File Reference

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



### Classes

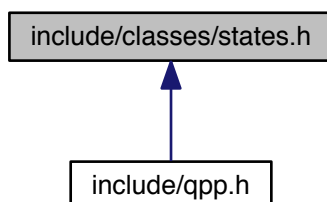
- class [qpp::NormalDistribution](#)
- class [qpp::UniformRealDistribution](#)
- class [qpp::UniformIntDistribution](#)
- class [qpp::DiscreteDistribution](#)
- class [qpp::DiscreteDistributionAbsSquare](#)

### Namespaces

- [qpp](#)

## 7.8 include/classes/states.h File Reference

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



### Classes

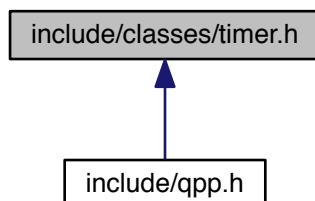
- class [qpp::States](#)

## Namespaces

- [qpp](#)

## 7.9 include/classes/timer.h File Reference

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



## Classes

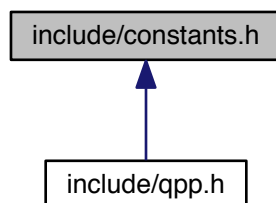
- class [qpp::Timer](#)

## Namespaces

- [qpp](#)

## 7.10 include/constants.h File Reference

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



## Namespaces

- [qpp](#)

## Functions

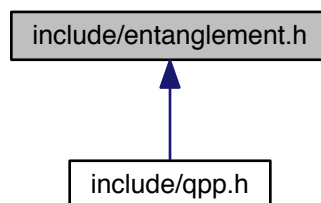
- constexpr std::complex< double > [qpp::operator""\\_i](#) (unsigned long long int x)  
*User-defined literal for complex  $i = \sqrt{-1}$  (integer overload)*
- constexpr std::complex< double > [qpp::operator""\\_i](#) (long double x)  
*User-defined literal for complex  $i = \sqrt{-1}$  (real overload)*
- 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  
 $\pi$
- constexpr double [qpp::ee](#) = 2.718281828459045235360287471352662497  
*Base of natural logarithm,  $e$ .*

## 7.11 include/entanglement.h File Reference

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



## Namespaces

- [qpp](#)

## Functions

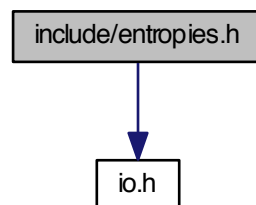
- template<typename Derived >  
cmat [qpp::schmidtcoeff](#) (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size\_t > &dims)  
*Schmidt coefficients of the bi-partite pure state A.*

- `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.*

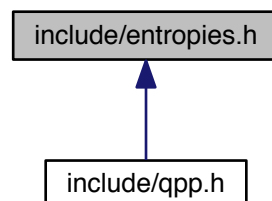
## 7.12 include/entropies.h File Reference

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

Include dependency graph for entropies.h:



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



## Namespaces

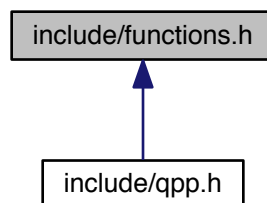
- [qpp](#)

## 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-  $\alpha$  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-  $\infty$  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-  $\alpha$  entropy of the probability distribution/density matrix A, for  $\alpha \geq 0$*
- `template<typename Derived >`  
`double qpp::qmutualinfo (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &subsysA, const std::vector< std::size_t > &subsysB, const std::vector< std::size_t > &dims)`  
*Quantum mutual information between 2 subsystems of a composite system.*

## 7.13 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)

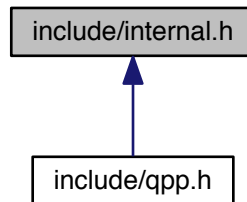
*Partial transpose.*

- `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.*



## 7.14 include/internal.h File Reference

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



### Namespaces

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

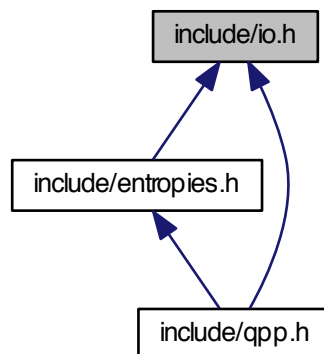
### Functions

- 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)`

## 7.15 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)`

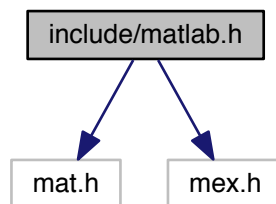
*Saves Eigen expression to a binary file (internal format) in double precision.*

- `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.*

## 7.16 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](#))*

## 7.17 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 <numeric>
#include <ostream>
#include <random>
#include <stdexcept>
#include <string>
#include <sstream>
#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

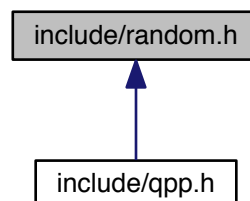
- [qpp](#)

## 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*

## 7.18 include/random.h File Reference

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



## Namespaces

- [qpp](#)

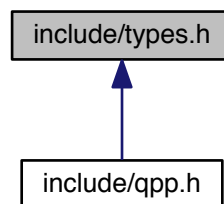
## Functions

- template<typename Derived >  
Derived [qpp::rand](#) (std::size\_t rows, std::size\_t cols, double a=0, double b=1)
- template<>  
dmat [qpp::rand](#) (std::size\_t rows, std::size\_t cols, double a, double b)
- template<>  
cmat [qpp::rand](#) (std::size\_t rows, std::size\_t cols, double a, double b)
- double [qpp::rand](#) (double a=0, double b=1)
- long long [qpp::randint](#) (long long a, long long b)

- `template<typename Derived >`  
Derived `qpp::randn` (`std::size_t` rows, `std::size_t` cols, double mean=0, double sigma=1)
- `template<>`  
`dmat qpp::randn` (`std::size_t` rows, `std::size_t` cols, double mean, double sigma)
- `template<>`  
`cmat qpp::randn` (`std::size_t` rows, `std::size_t` cols, double mean, double sigma)
- double `qpp::randn` (double mean=0, double sigma=1)
- `cmat qpp::randU` (`std::size_t` D)
- `cmat qpp::randV` (`std::size_t` Din, `std::size_t` Dout)
- `std::vector< cmat > qpp::randkraus` (`std::size_t` n, `std::size_t` D)
- `cmat qpp::randH` (`std::size_t` D)
- `ket qpp::randket` (`std::size_t` D)
- `cmat qpp::randrho` (`std::size_t` D)
- `std::vector< std::size_t > qpp::randperm` (`std::size_t` n)

## 7.19 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, 16  
adjoint  
    qpp, 16  
anticomm  
    qpp, 17  
  
bra  
    qpp, 15  
  
CUSTOM\_EXCEPTION  
    qpp::Exception, 79  
channel  
    qpp, 17, 19  
choi  
    qpp, 20  
choi2kraus  
    qpp, 21  
chop  
    qpp, 71  
cmat  
    qpp, 15  
comm  
    qpp, 22  
compperm  
    qpp, 22  
conjugate  
    qpp, 24  
cosm  
    qpp, 24  
cplx  
    qpp, 15  
cwise  
    qpp, 25  
  
DIMS\_INVALID  
    qpp::Exception, 79  
DIMS\_MISMATCH\_CVECTOR  
    qpp::Exception, 79  
DIMS\_MISMATCH\_MATRIX  
    qpp::Exception, 79  
DIMS\_MISMATCH\_RVECTOR  
    qpp::Exception, 79  
DIMS\_MISMATCH\_VECTOR  
    qpp::Exception, 79  
DIMS\_NOT\_EQUAL  
    qpp::Exception, 79  
det  
    qpp, 25  
disp  
    qpp, 26, 27  
displn  
    qpp, 27–29  
dmat  
    qpp, 15  
  
ee  
    qpp, 71  
entanglement  
    qpp, 29  
eps  
    qpp, 71  
evals  
    qpp, 30  
evects  
    qpp, 31  
expandout  
    qpp, 31  
expm  
    qpp, 32  
  
funm  
    qpp, 33  
  
gconcurrency  
    qpp, 33  
grams  
    qpp, 34, 35  
gt  
    qpp, 71  
  
hevals  
    qpp, 36  
hevects  
    qpp, 36  
  
inverse  
    qpp, 37  
invperm  
    qpp, 37  
  
ket  
    qpp, 16  
kron  
    qpp, 38, 39  
kronpow  
    qpp, 40  
  
load  
    qpp, 40  
logdet

- qpp, 41
- logm
  - qpp, 42
- MATRIX\_NOT\_CVECTOR
  - qpp::Exception, 79
- MATRIX\_NOT\_RVECTOR
  - qpp::Exception, 79
- MATRIX\_NOT\_SQUARE
  - qpp::Exception, 79
- MATRIX\_NOT\_SQUARE\_OR\_CVECTOR
  - qpp::Exception, 79
- MATRIX\_NOT\_SQUARE\_OR\_RVECTOR
  - qpp::Exception, 79
- MATRIX\_NOT\_SQUARE\_OR\_VECTOR
  - qpp::Exception, 79
- MATRIX\_NOT\_VECTOR
  - qpp::Exception, 79
- maxn
  - qpp, 71
- mket
  - qpp, 42, 43
- multiidx2n
  - qpp, 44
- n2multiidx
  - qpp, 44
- NOT\_BIPARTITE
  - qpp::Exception, 79
- NOT\_QUBIT\_GATE
  - qpp::Exception, 79
- NOT\_QUBIT\_SUBSYS
  - qpp::Exception, 79
- norm
  - qpp, 45
- OUT\_OF\_RANGE
  - qpp::Exception, 79
- omega
  - qpp, 45
- PERM\_INVALID
  - qpp::Exception, 79
- pi
  - qpp, 71
- powm
  - qpp, 46
- prj
  - qpp, 46
- ptrace
  - qpp, 47
- ptrace1
  - qpp, 48
- ptrace2
  - qpp, 49
- ptranspose
  - qpp, 50
- qmutualinfo
  - qpp, 51
- qpp, 9
  - absm, 16
  - adjoint, 16
  - anticomm, 17
  - bra, 15
  - channel, 17, 19
  - choi, 20
  - choi2kraus, 21
  - chop, 71
  - cmat, 15
  - comm, 22
  - compperm, 22
  - conjugate, 24
  - cosm, 24
  - cplx, 15
  - cwise, 25
  - det, 25
  - disp, 26, 27
  - displn, 27–29
  - dmat, 15
  - ee, 71
  - entanglement, 29
  - eps, 71
  - evals, 30
  - evects, 31
  - expandout, 31
  - expm, 32
  - funm, 33
  - gconcurrence, 33
  - grams, 34, 35
  - gt, 71
  - hevals, 36
  - hevects, 36
  - inverse, 37
  - invperm, 37
  - ket, 16
  - kron, 38, 39
  - kronpow, 40
  - load, 40
  - logdet, 41
  - logm, 42
  - maxn, 71
  - mket, 42, 43
  - multiidx2n, 44
  - n2multiidx, 44
  - norm, 45
  - omega, 45
  - pi, 71
  - powm, 46
  - prj, 46
  - ptrace, 47
  - ptrace1, 48
  - ptrace2, 49
  - ptranspose, 50
  - qmutualinfo, 51
  - rand, 52
  - randint, 53



- randket, 53
- randkraus, 54
- randn, 54, 55
- randperm, 55
- randrho, 55
- rdevs, 72
- renyi, 56
- reshape, 57
- save, 59
- schmidtcoeff, 60
- schmidtprob, 61
- shannon, 64
- sinm, 64
- spectralpowm, 66
- sqrtn, 66
- st, 72
- sum, 67
- super, 67
- syspermute, 68
- trace, 69
- transpose, 70
- tsallis, 70
- qpp::Exception
  - CUSTOM\_EXCEPTION, 79
  - DIMS\_INVALID, 79
  - DIMS\_MISMATCH\_CVECTOR, 79
  - DIMS\_MISMATCH\_MATRIX, 79
  - DIMS\_MISMATCH\_RVECTOR, 79
  - DIMS\_MISMATCH\_VECTOR, 79
  - DIMS\_NOT\_EQUAL, 79
  - MATRIX\_NOT\_CVECTOR, 79
  - MATRIX\_NOT\_RVECTOR, 79
  - MATRIX\_NOT\_SQUARE, 79
  - MATRIX\_NOT\_SQUARE\_OR\_CVECTOR, 79
  - MATRIX\_NOT\_SQUARE\_OR\_RVECTOR, 79
  - MATRIX\_NOT\_SQUARE\_OR\_VECTOR, 79
  - MATRIX\_NOT\_VECTOR, 79
  - NOT\_BIPARTITE, 79
  - NOT\_QUBIT\_GATE, 79
  - NOT\_QUBIT\_SUBSYS, 79
  - OUT\_OF\_RANGE, 79
  - PERM\_INVALID, 79
  - SUBSYS\_MISMATCH\_DIMS, 79
  - TYPE\_MISMATCH, 79
  - UNDEFINED\_TYPE, 79
  - UNKNOWN\_EXCEPTION, 79
  - ZERO\_SIZE, 79
- rand
  - qpp, 52
- randint
  - qpp, 53
- randket
  - qpp, 53
- randkraus
  - qpp, 54
- randn
  - qpp, 54, 55
- randperm
  - qpp, 55
- randrho
  - qpp, 55
- rdevs
  - qpp, 72
- renyi
  - qpp, 56
- reshape
  - qpp, 57
- SUBSYS\_MISMATCH\_DIMS
  - qpp::Exception, 79
- save
  - qpp, 59
- schmidtcoeff
  - qpp, 60
- schmidtprob
  - qpp, 61
- shannon
  - qpp, 64
- sinm
  - qpp, 64
- spectralpowm
  - qpp, 66
- sqrtn
  - qpp, 66
- st
  - qpp, 72
- sum
  - qpp, 67
- super
  - qpp, 67
- syspermute
  - qpp, 68
- TYPE\_MISMATCH
  - qpp::Exception, 79
- trace
  - qpp, 69
- transpose
  - qpp, 70
- tsallis
  - qpp, 70
- UNDEFINED\_TYPE
  - qpp::Exception, 79
- UNKNOWN\_EXCEPTION
  - qpp::Exception, 79
- ZERO\_SIZE
  - qpp::Exception, 79