quantum++ 0.1

Generated by Doxygen 1.8.7

Fri Nov 14 2014 14:55:19

# **Contents**

1	Qua	ntum++	- A C++1	11 q	uan	ıtum	1 CO	mp	utir	ng	libı	rary	/									1
2	Nam	espace	Index																			3
	2.1	Names	pace List	t											 				 			3
3	Hier	archical	Index																			5
	3.1	Class I	Hierarchy												 				 			5
4	Clas	s Index																				7
	4.1	Class I	_ist												 				 			7
5	File	Index																				9
	5.1	File Lis	t												 				 			9
6	Nam	espace	Documer	nta	ıtior	1																11
	6.1	qpp Na	ımespace	Re	efere	ence	٠								 				 			11
		6.1.1	Typedef I	Do	cum	enta	atior	ı							 				 			20
			6.1.1.1	b	ra .										 				 			20
			6.1.1.2	С	mat										 				 			21
			6.1.1.3	С	plx .										 				 			21
			6.1.1.4	d	lmat										 				 			21
			6.1.1.5	D	OynC	ColVe	ect								 				 			21
			6.1.1.6	D	) Dyn <b>l</b>	∕lat .									 				 			21
			6.1.1.7	D	) ynF	low\	Vect								 				 			21
			6.1.1.8	k	et .										 				 		 	21
		6.1.2	Function	n Do	ocur	nent	tatio	n .							 				 			21
			6.1.2.1	a	ıbsm	١									 				 		 	21
			6.1.2.2	а	ıbssı	<b>q</b>									 				 			22
			6.1.2.3	а	ıbssı	<b>q</b>									 				 			22
			6.1.2.4	a	ıdjoiı	nt .									 				 			22
			6.1.2.5	a	ıntic	omn	n								 				 		 	22
			6.1.2.6	а	ıpply	<i>,</i>									 				 			23
			6.1.2.7	а	ıpply	<i>,</i>									 				 			23

iv CONTENTS

6.1.2.8	applyCTRL	23
6.1.2.9	applyCTRL	24
6.1.2.10	channel	24
6.1.2.11	channel	24
6.1.2.12	channel	25
6.1.2.13	choi	25
6.1.2.14	choi2kraus	25
6.1.2.15	comm	26
6.1.2.16	compperm	26
6.1.2.17	concurrence	26
6.1.2.18	conjugate	26
6.1.2.19	contfrac2x	27
6.1.2.20	contfrac2x	27
6.1.2.21	cosm	27
6.1.2.22	cwise	27
6.1.2.23	det	28
6.1.2.24	disp	28
6.1.2.25	disp	28
6.1.2.26	disp	28
6.1.2.27	disp	29
6.1.2.28	disp	29
6.1.2.29	entanglement	29
6.1.2.30	evals	30
6.1.2.31	evects	30
6.1.2.32	expm	30
6.1.2.33	funm	30
6.1.2.34	gcd	31
6.1.2.35	gcd	31
6.1.2.36	gconcurrence	31
6.1.2.37	grams	31
6.1.2.38	grams	32
6.1.2.39	grams	32
6.1.2.40	hevals	32
6.1.2.41	hevects	32
6.1.2.42	inverse	33
6.1.2.43	invperm	33
6.1.2.44	kron	33
6.1.2.45	kron	33
6.1.2.46	kron	34
6.1.2.47	kron	34

CONTENTS

6.1.2.48	kronpow	34
6.1.2.49	lcm	34
6.1.2.50	lcm	35
6.1.2.51	load	35
6.1.2.52	loadMATLABmatrix	35
6.1.2.53	loadMATLABmatrix	35
6.1.2.54	loadMATLABmatrix	36
6.1.2.55	logdet	36
6.1.2.56	logm	36
6.1.2.57	lognegativity	37
6.1.2.58	measure	37
6.1.2.59	measure	37
6.1.2.60	measure	38
6.1.2.61	measure	38
6.1.2.62	measure	39
6.1.2.63	measure	39
6.1.2.64	measure	39
6.1.2.65	measure	40
6.1.2.66	measure	40
6.1.2.67	mket	40
6.1.2.68	mket	41
6.1.2.69	mprj	42
6.1.2.70	mprj	42
6.1.2.71	multiidx2n	42
6.1.2.72	n2multiidx	43
6.1.2.73	negativity	44
6.1.2.74	norm	44
6.1.2.75	omega	44
6.1.2.76	operator""""_i	44
6.1.2.77	operator"""_i	45
6.1.2.78	powm	45
6.1.2.79	prj	45
6.1.2.80	prod	45
6.1.2.81	prod	45
6.1.2.82	ptrace	46
6.1.2.83	ptrace	46
6.1.2.84	ptrace1	46
6.1.2.85	ptrace2	47
6.1.2.86	ptranspose	47
6.1.2.87	ptranspose	47

vi CONTENTS

6.1.2.88	qmutualinfo	48
6.1.2.89	rand	49
6.1.2.90	rand	49
6.1.2.91	rand	49
6.1.2.92	rand	50
6.1.2.93	randH	50
6.1.2.94	randint	50
6.1.2.95	randket	50
6.1.2.96	randkraus	51
6.1.2.97	randn	51
6.1.2.98	randn	51
6.1.2.99	randn	52
6.1.2.100	randn	52
6.1.2.101	randperm	52
6.1.2.102	2 randrho	52
6.1.2.103	BrandU	53
6.1.2.104	FrandV	53
6.1.2.105	orenyi	53
6.1.2.106	Sireshape	53
6.1.2.107	<sup>7</sup> rho2pure	54
6.1.2.108	B save	54
6.1.2.109	9 saveMATLABmatrix	54
6.1.2.110	saveMATLABmatrix	54
6.1.2.111	saveMATLABmatrix	55
6.1.2.112	2 schatten	55
6.1.2.113	3 schmidtcoeff	55
6.1.2.114	schmidtprob	56
	schmidtU	56
6.1.2.116	S schmidtV	56
6.1.2.117	shannon	56
6.1.2.118	3 sinm	57
	9 spectralpowm	57
6.1.2.120	sqrtm	57
6.1.2.121	sum	57
	2 sum	58
	3 super	58
	I svals	58
	5 svdU	58
	S svdV	59
6.1.2.127	<sup>7</sup> syspermute	59

CONTENTS vii

		6.1.2.128 syspermute	59
		6.1.2.129 trace	59
		6.1.2.130 transpose	60
		6.1.2.131 tsallis	60
		6.1.2.132 x2contfrac	60
	6.1.3	Variable Documentation	60
		6.1.3.1 chop	60
		6.1.3.2 codes	61
		6.1.3.3 ee	61
		6.1.3.4 eps	61
		6.1.3.5 gt	61
		6.1.3.6 infty	61
		6.1.3.7 init	61
		6.1.3.8 maxn	61
		6.1.3.9 pi	61
		6.1.3.10 rdevs	61
		6.1.3.11 st	61
6.2	qpp::ex	perimental Namespace Reference	62
	6.2.1	Detailed Description	62
6.3	qpp::in	ternal Namespace Reference	62
	6.3.1	Detailed Description	63
	6.3.2	Function Documentation	63
		6.3.2.1 _check_col_vector	63
		6.3.2.2 _check_dims	63
		6.3.2.3 _check_dims_match_cvect	63
		6.3.2.4 _check_dims_match_mat	63
		6.3.2.5 _check_dims_match_rvect	63
		6.3.2.6 _check_eq_dims	63
		6.3.2.7 _check_nonzero_size	63
		6.3.2.8 _check_perm	63
		6.3.2.9 _check_row_vector	63
		6.3.2.10 _check_square_mat	63
		6.3.2.11 _check_subsys_match_dims	63
		6.3.2.12 _check_vector	63
		6.3.2.13 _kron2	63
		6.3.2.14 _multiidx2n	63
		6.3.2.15 _n2multiidx	63
		6.3.2.16 variadic_vector_emplace	63
		6.3.2.17 variadic_vector_emplace	63

viii CONTENTS

7	Clas	s Docu	mentation	65
	7.1	qpp::C	odes Class Reference	65
		7.1.1	Detailed Description	66
		7.1.2	Member Enumeration Documentation	66
			7.1.2.1 Type	66
		7.1.3	Constructor & Destructor Documentation	66
			7.1.3.1 Codes	66
		7.1.4	Member Function Documentation	66
			7.1.4.1 codeword	66
		7.1.5	Friends And Related Function Documentation	67
			7.1.5.1 internal::Singleton < const Codes >	67
	7.2	qpp::E	xception Class Reference	67
		7.2.1	Detailed Description	69
		7.2.2	Member Enumeration Documentation	69
			7.2.2.1 Type	69
		7.2.3	Constructor & Destructor Documentation	70
			7.2.3.1 Exception	70
			7.2.3.2 Exception	70
		7.2.4	Member Function Documentation	70
			7.2.4.1 _construct_exception_msg	70
			7.2.4.2 what	70
		7.2.5	Member Data Documentation	70
			7.2.5.1 _custom	70
			7.2.5.2 _msg	71
			7.2.5.3 _type	71
			7.2.5.4 _where	71
	7.3	qpp::G	ates Class Reference	71
		7.3.1	Detailed Description	73
		7.3.2	Constructor & Destructor Documentation	73
			7.3.2.1 Gates	73
		7.3.3	Member Function Documentation	73
			7.3.3.1 CTRL	73
			7.3.3.2 expandout	73
			7.3.3.3 Fd	74
			7.3.3.4 ld	74
			7.3.3.5 Rn	74
			7.3.3.6 Xd	75
			7.3.3.7 Zd	75
		7.3.4	Friends And Related Function Documentation	75
			7.3.4.1 internal::Singleton < const Gates >	75

CONTENTS

	7.3.5	Member Data Documentation	75
		7.3.5.1 CNOTab	75
		7.3.5.2 CNOTba	75
		7.3.5.3 CZ	75
		7.3.5.4 FRED	75
		7.3.5.5 H	76
		7.3.5.6 ld2	76
		7.3.5.7 S	76
		7.3.5.8 SWAP	76
		7.3.5.9 T	76
		7.3.5.10 TOF	76
		7.3.5.11 X	76
		7.3.5.12 Y	76
		7.3.5.13 Z	76
7.4	qpp::In	Class Reference	76
	7.4.1	Detailed Description	77
	7.4.2	Constructor & Destructor Documentation	78
		7.4.2.1 Init	78
		7.4.2.2 $\sim$ Init	78
	7.4.3	Friends And Related Function Documentation	78
		7.4.3.1 internal::Singleton< const Init >	78
7.5	qpp::in	rnal::IOManipEigen Class Reference	78
	7.5.1	Constructor & Destructor Documentation	78
		7.5.1.1 IOManipEigen	78
		7.5.1.2 IOManipEigen	78
	7.5.2	Friends And Related Function Documentation	78
		7.5.2.1 operator<<	79
	7.5.3	Member Data Documentation	79
		7.5.3.1 _A	79
		7.5.3.2 _chop	79
7.6	qpp::in	rnal::IOManipPointer< PointerType > Class Template Reference	79
	7.6.1	Constructor & Destructor Documentation	80
		7.6.1.1 IOManipPointer	80
		7.6.1.2 IOManipPointer	80
	7.6.2	Member Function Documentation	80
		7.6.2.1 operator=	80
	7.6.3	Friends And Related Function Documentation	80
		7.6.3.1 operator <<	80
	7.6.4	Member Data Documentation	80
		7.6.4.1 _end	80

CONTENTS

		7.6.4.2 _n	80
		7.6.4.3 _p	80
		7.6.4.4 _separator	80
		7.6.4.5 _start	80
7.7	qpp::int	ternal::IOManipRange< InputIterator > Class Template Reference	80
	7.7.1	Constructor & Destructor Documentation	81
		7.7.1.1 IOManipRange	81
	7.7.2	Friends And Related Function Documentation	81
		7.7.2.1 operator<<	82
	7.7.3	Member Data Documentation	82
		7.7.3.1 _end	82
		7.7.3.2 _first	82
		7.7.3.3 _last	82
		7.7.3.4 _separator	82
		7.7.3.5 _start	82
7.8	qpp::Ra	andomDevices Class Reference	82
	7.8.1	Detailed Description	83
	7.8.2	Constructor & Destructor Documentation	83
		7.8.2.1 RandomDevices	83
	7.8.3	Friends And Related Function Documentation	84
		7.8.3.1 internal::Singleton < RandomDevices >	84
	7.8.4	Member Data Documentation	84
		7.8.4.1 _rd	84
		7.8.4.2 _rng	84
7.9	qpp::int	ternal::Singleton< T > Class Template Reference	84
	7.9.1	Detailed Description	84
	7.9.2	Constructor & Destructor Documentation	85
		7.9.2.1 Singleton	85
		7.9.2.2 ~Singleton	85
		7.9.2.3 Singleton	85
	7.9.3	Member Function Documentation	85
		7.9.3.1 get_instance	85
		7.9.3.2 operator=	85
7.10	qpp::St	ates Class Reference	85
	7.10.1	Detailed Description	87
	7.10.2	Constructor & Destructor Documentation	87
		7.10.2.1 States	87
	7.10.3	Friends And Related Function Documentation	87
		7.10.3.1 internal::Singleton < const States >	87
	7.10.4	Member Data Documentation	87

CONTENTS xi

	7.10.4.1 b00	87
	7.10.4.2 b01	87
	7.10.4.3 b10	88
	7.10.4.4 b11	88
	7.10.4.5 GHZ	88
	7.10.4.6 pb00	88
	7.10.4.7 pb01	88
	7.10.4.8 pb10	88
	7.10.4.9 pb11	88
	7.10.4.10 pGHZ	88
	7.10.4.11 pW	88
	7.10.4.12 px0	88
	7.10.4.13 px1	88
	7.10.4.14 py0	88
	7.10.4.15 py1	89
	7.10.4.16 pz0	89
	7.10.4.17 pz1	89
	7.10.4.18 W	89
	7.10.4.19 x0	89
	7.10.4.20 x1	89
	7.10.4.21 y0	89
	7.10.4.22 y1	89
	7.10.4.23 z0	89
	7.10.4.24 z1	89
7.11 qpp::T	imer Class Reference	89
7.11.1	Detailed Description	90
7.11.2	Constructor & Destructor Documentation	90
	7.11.2.1 Timer	90
7.11.3	Member Function Documentation	90
	7.11.3.1 seconds	90
	7.11.3.2 tic	90
	7.11.3.3 toc	91
7.11.4	Friends And Related Function Documentation	91
	7.11.4.1 operator<<	91
7.11.5	Member Data Documentation	91
	7.11.5.1 _end	91
	7.11.5.2 _start	91
File Docum	entation	93
	e/classes/codes.h File Reference	93
J. I IIICIUUR	Columbia Col	55

8

xii CONTENTS

8.2	include/classes/exception.h File Reference	94
8.3	include/classes/gates.h File Reference	94
8.4	include/classes/init.h File Reference	95
8.5	include/classes/random_devices.h File Reference	95
8.6	include/classes/singleton.h File Reference	96
8.7	include/classes/states.h File Reference	97
8.8	include/classes/timer.h File Reference	97
8.9	include/constants.h File Reference	98
8.10	include/entanglement.h File Reference	99
8.11	include/entropies.h File Reference	100
8.12	include/experimental/test.h File Reference	100
8.13	include/functions.h File Reference	101
8.14	include/internal/functions.h File Reference	104
8.15	include/input_output.h File Reference	105
8.16	include/instruments.h File Reference	106
8.17	include/internal/classes/iomanip.h File Reference	108
8.18	include/MATLAB/matlab.h File Reference	108
8.19	include/number_theory.h File Reference	109
8.20	include/operations.h File Reference	110
8.21	include/qpp.h File Reference	112
8.22	include/random.h File Reference	113
8.23	include/types.h File Reference	114
8.24	mainpage.dox File Reference	115

Index

116

## **Chapter 1**

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

```
Version

0.1

Author

Vlad Gheorghiu (vgheorgh@gmail.com)

Copyright

(c) 2013 - 2014 Vlad Gheorghiu (vgheorgh@gmail.com)
```

See the file example.cpp for some usage examples.

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

# Chapter 2

# Namespace Index

2.1 Namespace Lis	2.1	Nar	nesp	ace	Lis
-------------------	-----	-----	------	-----	-----

Here	ic a	lict	of all	namespaces	with	hrief	descriptions
11010	10 0	ເມວເ	UI all	Hallicspaces	VVILII	וסווט	ucscriptions.

qpp	 - 11
qpp::experimental	 62
qpp::internal	 62

Namespace Index

# **Chapter 3**

## **Hierarchical Index**

### 3.1 Class Hierarchy

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

std::exception	
qpp::Exception	67
qpp::internal::IOManipEigen	78
qpp::internal::IOManipPointer< PointerType >	79
qpp::internal::IOManipRange< InputIterator >	80
$qpp::internal::Singleton < T > \dots \dots$	84
qpp::internal::Singleton < const Codes >	84
qpp::Codes	65
qpp::internal::Singleton < const Gates >	84
qpp::Gates	71
$qpp :: internal :: Singleton < const \; Init > \dots $	84
qpp::Init	76
qpp::internal::Singleton < const States >	84
qpp::States	85
qpp::internal::Singleton< RandomDevices >	84
qpp::RandomDevices	82
gpp: Timer	20

6 **Hierarchical Index** 

# **Chapter 4**

## **Class Index**

### 4.1 Class List

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

qpp::Codes	
Const Singleton class that defines quantum error correcting codes	65
qpp::Exception	
Generates custom exceptions, used when validating function parameters	67
qpp::Gates	
Const Singleton class that implements most commonly used gates	71
qpp::Init	
Const Singleton class that performs additional initializations/cleanups	76
qpp::internal::IOManipEigen	78
qpp::internal::IOManipPointer< PointerType >	79
qpp::internal::IOManipRange< InputIterator >	80
qpp::RandomDevices	
Singeleton class that manages the source of randomness in the library	82
qpp::internal::Singleton< T >	
Singleton policy class, used internally to implement the singleton pattern via CRTP (Curiously	
recurring template pattern)	84
qpp::States	
Const Singleton class that implements most commonly used states	85
qpp::Timer	
Measures time	89

8 Class Index

# **Chapter 5**

## File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

include/constants.h
include/entanglement.h
include/entropies.h
include/functions.h
include/input_output.h
include/instruments.h
include/number_theory.h
include/operations.h
include/qpp.h
include/random.h
include/types.h
include/classes/codes.h
include/classes/exception.h
include/classes/gates.h
include/classes/init.h
include/classes/random_devices.h
include/classes/singleton.h
include/classes/states.h
include/classes/timer.h
include/experimental/test.h
include/internal/functions.h
include/internal/classes/iomanip.h
include/MATLAR/matlab h

10 File Index

### **Chapter 6**

### **Namespace Documentation**

### 6.1 qpp Namespace Reference

### **Namespaces**

- · experimental
- internal

#### Classes

· class Codes

const Singleton class that defines quantum error correcting codes

class Exception

Generates custom exceptions, used when validating function parameters.

class Gates

const Singleton class that implements most commonly used gates

· class Init

const Singleton class that performs additional initializations/cleanups

· class RandomDevices

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

class States

const Singleton class that implements most commonly used states

· class Timer

Measures time.

### **Typedefs**

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

Complex number in double precision.

```
• template<typename Scalar >
```

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

Dynamic Eigen matrix over the field specified by Scalar.

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

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

Dynamic Eigen column vector over the field specified by Scalar.

```
• template<typename Scalar >
```

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

Dynamic Eigen row vector over the field specified by Scalar.

using ket = DynColVect< cplx >

Complex (double precision) dynamic Eigen column vector.

• using bra = DynRowVect< cplx >

Complex (double precision) dynamic Eigen row vector.

using cmat = DynMat< cplx >

Complex (double precision) dynamic Eigen matrix.

using dmat = DynMat< double >

Real (double precision) dynamic Eigen matrix.

#### **Functions**

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

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

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

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

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

D-th root of unity.

• template<typename Derived >

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

Schmidt coefficients of the bi-partite pure state A.

template<typename Derived >

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

Schmidt basis on Alice's side.

• template<typename Derived >

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

Schmidt basis on Bob's side.

• template<typename Derived >

 $\label{lem:decomposition} \begin{tabular}{ll} DynColVect< double > schmidtprob (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size \leftarrow \_t > &dims) \end{tabular}$ 

Schmidt probabilities of the bi-partite pure state A.

template<typename Derived >

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

Entanglement of the bi-partite pure state A.

template<typename Derived >

double gconcurrence (const Eigen::MatrixBase< Derived > &A)

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

• template<typename Derived >

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

Negativity of the bi-partite mixed state A.

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

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

Logarithmic negativity of the bi-partite mixed state A.

template<typename Derived >

double concurrence (const Eigen::MatrixBase< Derived > &A)

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

template<typename Derived >

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

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

template<typename Derived >

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

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

    template<typename Derived >

  Derived::Scalar trace (const Eigen::MatrixBase< Derived > &A)
      Trace.

    template<typename Derived >

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

    template<typename Derived >

  Derived::Scalar sum (const Eigen::MatrixBase< Derived > &A)
      Element-wise sum of A.
• template<typename Derived >
  Derived::Scalar prod (const Eigen::MatrixBase< Derived > &A)
      Element-wise product of A.
• template<typename Derived >
  double norm (const Eigen::MatrixBase< Derived > &A)
      Frobenius norm.

    template<typename Derived >

  DynColVect< cplx > evals (const Eigen::MatrixBase< Derived > &A)
      Eigenvalues.

    template<typename Derived >

  cmat evects (const Eigen::MatrixBase< Derived > &A)
      Eigenvectors.

    template<typename Derived >

  {\color{red} {\sf DynColVect}{<}}\ {\color{blue} {\sf double}} > {\color{blue} {\sf hevals}}\ ({\color{blue} {\sf const}}\ {\color{blue} {\sf Eigen}} :: {\color{blue} {\sf MatrixBase}{<}}\ {\color{blue} {\sf Derived}} > {\color{blue} \&A})
      Hermitian eigenvalues.

    template<typename Derived >

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

    template<typename Derived >

  DynColVect< double > svals (const Eigen::MatrixBase< Derived > &A)
      Singular values.
```

Kronecker product.

```
• template<typename Derived >
  cmat svdU (const Eigen::MatrixBase< Derived > &A)
     Left singular vectors.
• template<typename Derived >
  cmat svdV (const Eigen::MatrixBase< Derived > &A)
     Right singular vectors.
• template<typename Derived >
  cmat funm (const Eigen::MatrixBase< Derived > &A, cplx(*f)(const cplx &))
     Functional calculus f(A)

    template<typename Derived >

  cmat sqrtm (const Eigen::MatrixBase< Derived > &A)
     Matrix square root.

    template<typename Derived >

  cmat absm (const Eigen::MatrixBase< Derived > &A)
     Matrix absolut value.

    template<typename Derived >

  cmat expm (const Eigen::MatrixBase< Derived > &A)
     Matrix exponential.
• template<typename Derived >
  cmat logm (const Eigen::MatrixBase< Derived > &A)
     Matrix logarithm.

    template<typename Derived >

  cmat sinm (const Eigen::MatrixBase< Derived > &A)
     Matrix sin.
• template<typename Derived >
  cmat cosm (const Eigen::MatrixBase< Derived > &A)
     Matrix cos.
• template<typename Derived >
  cmat spectralpowm (const Eigen::MatrixBase< Derived > &A, const cplx z)
     Matrix power.
\bullet \ \ \text{template}{<} \text{typename Derived} >
  DynMat< typename Derived::Scalar > powm (const Eigen::MatrixBase< Derived > &A, std::size_t n)
     Matrix power.
• template<typename Derived >
  double schatten (const Eigen::MatrixBase< Derived > &A, std::size_t p)
     Schatten norm.
• template<typename OutputScalar , typename Derived >
  DynMat< OutputScalar > cwise (const Eigen::MatrixBase< Derived > &A, OutputScalar(*f)(const typename
  Derived::Scalar &))
     Functor.
template<typename T >
  DynMat< typename T::Scalar > kron (const T &head)
     Kronecker product.
• template<typename T , typename... Args>
  DynMat< typename T::Scalar > kron (const T &head, const Args &...tail)
     Kronecker product.
template<typename Derived >
  DynMat< typename Derived::Scalar > kron (const std::vector< Derived > &As)
     Kronecker product.

    template<typename Derived >

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

```
    template<typename Derived >

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

    template<typename Derived >

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

    template<typename Derived1, typename Derived2 >

  DynMat< typename Derived1::Scalar > comm (const Eigen::MatrixBase< Derived1 > &A, const Eigen::←
  MatrixBase < Derived2 > &B)
     Commutator.

    template<typename Derived1 , typename Derived2 >

  DynMat< typename Derived1::Scalar > anticomm (const Eigen::MatrixBase< Derived1 > &A, const Eigen ←
  ::MatrixBase< Derived2 > &B)
     Anti-commutator.

    template<typename Derived >

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

    template<typename Derived >

  DynMat< typename Derived::Scalar > grams (const std::vector< Derived > &Vs)
     Gram-Schmidt orthogonalization.

    template<typename Derived >

  DynMat< typename Derived::Scalar > grams (const std::initializer_list< Derived > &Vs)
      Gram-Schmidt orthogonalization.
template<typename Derived >
  DynMat< typename Derived::Scalar > grams (const Eigen::MatrixBase< Derived > &A)
      Gram-Schmidt orthogonalization.
• std::vector< std::size_t > n2multiidx (std::size_t n, const std::vector< std::size_t > &dims)
     Non-negative integer index to multi-index.

    std::size_t multiidx2n (const std::vector < std::size_t > &midx, const std::vector < std::size_t > &dims)

     Multi-index to non-negative integer index.

    ket mket (const std::vector< std::size t > &mask, const std::vector< std::size t > &dims)

     Multi-partite qudit ket.

    ket mket (const std::vector < std::size_t > &mask, std::size_t d=2)

     Multi-partite qudit ket.

    cmat mprj (const std::vector < std::size_t > &mask, const std::vector < std::size_t > &dims)

     Projector onto multi-partite qudit ket.

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

     Projector onto multi-partite qudit ket.

    template<typename InputIterator >

  std::vector< double > abssq (InputIterator first, InputIterator last)
     Computes the absolut values squared of a range of complex numbers.

    template<typename Derived >

  std::vector< double > abssq (const Eigen::MatrixBase< Derived > &V)
      Computes the absolut values squared of a column vector.

    template<typename InputIterator >

  auto sum (InputIterator first, InputIterator last) -> typename InputIterator::value type
     Element-wise sum of a range.
• template<typename InputIterator >
  auto prod (InputIterator first, InputIterator last) -> typename InputIterator::value type
```

Element-wise product of a range.

```
• template<typename Derived >
  DynColVect< typename
  Derived::Scalar > rho2pure (const Eigen::MatrixBase < Derived > &A)
     Finds the pure state representation of a matrix proportional to a projector onto a pure state.

    template<typename Derived >

  internal::IOManipEigen disp (const Eigen::MatrixBase< Derived > &A, double chop=qpp::chop)
      Eigen expression ostream manipulator.
• internal::IOManipEigen disp (cplx z, double chop=qpp::chop)
     Complex number ostream manipulator.
• template<typename InputIterator >
  internal::IOManipRange
  < InputIterator > disp (const InputIterator &first, const InputIterator &last, const std::string &separator, const
  std::string &start="[", const std::string &end="]")
     Range ostream manipulator.

    template<typename Container >

  internal::IOManipRange
  < typename
  Container::const_iterator > disp (const Container &c, const std::string &separator, const std::string &start="[",
  const std::string &end="]")
     Standard container ostream manipulator. The container must support std::begin(), std::end() and forward iteration.
 template<typename PointerType >
  internal::IOManipPointer
  < PointerType > disp (const PointerType *p, std::size t n, const std::string &separator, const std::string
  &start="[", const std::string &end="]")
      C-style pointer ostream manipulator.
• template<typename Derived >
  void save (const Eigen::MatrixBase< Derived > &A, const std::string &fname)
     Saves Eigen expression to a binary file (internal format) in double precision.
• template<typename Derived >
  DynMat< typename Derived::Scalar > load (const std::string &fname)
     Loads Eigen matrix from a binary file (internal format) in double precision.

    template<typename Derived >

  std::tuple < std::size t,
  std::vector< double >
  , std::vector< cmat >> measure (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks,
  const std::vector< std::size_t > &subsys, const std::vector< std::size_t > &dims)
     Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.

    template<typename Derived >

  std::tuple < std::size t,
  std::vector< double >
  , std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const std::initializer list< cmat
  > &Ks, const std::vector< std::size_t > &subsys, const std::vector< std::size_t > &dims)
     Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.
• template<typename Derived >
  std::tuple < std::size t,
  std::vector< double >
  , std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks,
  const std::vector< std::size_t > &subsys, const std::size_t d=2)
     Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.
• template<typename Derived >
  std::tuple < std::size t,
  std::vector< double >
  , std::vector< cmat >> measure (const Eigen::MatrixBase< Derived > &A, const std::initializer list< cmat
  > &Ks, const std::vector< std::size_t > &subsys, const std::size_t d=2)
     Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.
```

```
• template<typename Derived >
  std::tuple < std::size t,
  std::vector< double >
  , std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const cmat &U, const std↔
  ::vector< std::size_t > &subsys, const std::vector< std::size_t > &dims)
     Measures the part subsys of the multi-partite state A in the orthonormal basis specified by the unitary matrix U.

    template<typename Derived >

  std::tuple < std::size t.
  std::vector< double >
  , std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const cmat &U, const std↔
  ::vector< std::size_t > &subsys, const std::size_t d=2)
     Measures the part subsys of the multi-partite state A in the orthonormal basis specified by the unitary matrix U.

    template<typename Derived >

  std::tuple < std::size t,
  std::vector< double >
  , std::vector < cmat > > measure (const Eigen::MatrixBase < Derived > &A, const std::vector < cmat > &Ks)
     Measures the state A using the set of Kraus operators Ks.

    template<typename Derived >

  std::tuple < std::size t.
  std::vector< double >
  , std::vector< cmat >> measure (const Eigen::MatrixBase< Derived > &A, const std::initializer_list< cmat
  > &Ks)
     Measures the state A using the set of Kraus operators Ks.

    template<typename Derived >

  std::tuple < std::size t,
  std::vector< double >
  , std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const cmat &U)
     Measures the state A in the orthonormal basis specified by the unitary matrix U.
template<typename Derived >
  Derived loadMATLABmatrix (const std::string &mat file, const std::string &var name)
     Loads an Eigen dynamic matrix from a MATLAB .mat file, generic version.
template<>
  dmat loadMATLABmatrix (const std::string &mat_file, const std::string &var_name)
     Loads an Eigen dynamic matrix from a MATLAB .mat file, specialization for double matrices (qpp::dmat)
template<>
  cmat loadMATLABmatrix (const std::string &mat_file, const std::string &var_name)
     Loads an Eigen dynamic matrix from a MATLAB .mat file, specialization for complex matrices (qpp::cmat)

    template<typename Derived >

  void saveMATLABmatrix (const Eigen::MatrixBase< Derived > &A, const std::string &mat file, const std↔
  ::string &var_name, const std::string &mode)
     Saves an Eigen dynamic matrix to a MATLAB .mat file, generic version.
• template<>
  void saveMATLABmatrix (const Eigen::MatrixBase < dmat > &A, const std::string &mat_file, const std::string
  &var name, const std::string &mode)
     Saves an Eigen dynamic matrix to a MATLAB .mat file, specialization for double matrices (qpp::dmat)
• template<>
  void saveMATLABmatrix (const Eigen::MatrixBase < cmat > &A, const std::string &mat_file, const std::string
  &var_name, const std::string &mode)
     Saves an Eigen dynamic matrix to a MATLAB .mat file, specialization for complex matrices (qpp::cmat)

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

     Simple continued fraction expansion.

    double contfrac2x (const std::vector< int > &cf, std::size t n)

      Real representation of a simple continued fraction.

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

Real representation of a simple continued fraction.

• std::size\_t gcd (std::size\_t m, std::size\_t n)

Greatest common divisor of two non-negative integers.

std::size t gcd (const std::vector < std::size t > &ns)

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

• std::size\_t lcm (std::size\_t m, std::size\_t n)

Least common multiple of two positive integers.

std::size\_t lcm (const std::vector< std::size\_t > &ns)

Least common multiple of a list of positive integers.

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

Inverse permutation.

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

Compose permutations.

• template<typename Derived1 , typename Derived2 >

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

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

• template<typename Derived1 , typename Derived2 >

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

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

• template<typename Derived1 , typename Derived2 >

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

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

template<typename Derived1, typename Derived2 >

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

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

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

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

• template<typename Derived >

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

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

template<typename Derived >

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

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

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

Superoperator matrix representation.

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

Choi matrix representation.

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

Extracts orthogonal Kraus operators from Choi matrix.

template<typename Derived >

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

Partial trace.

template<typename Derived >

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

Partial trace.

template<typename Derived >

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

Partial trace.

template<typename Derived >

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

Partial trace.

• template<typename Derived >

Partial transpose.

template<typename Derived >

 $\label{localize} \mbox{ DynMat} < \mbox{ typename Derived::Scalar} > \mbox{ptranspose (const Eigen::MatrixBase} < \mbox{ Derived} > \mbox{\&A, const std} \\ ::vector < \mbox{std::size\_t} > \mbox{\&subsys, std::size\_t} \mbox{ derived} > \mbox{\&A, const std} \\ ::vector < \mbox{std::size\_t} > \mbox{\&subsys, std::size\_t} \mbox{ derived} > \mbox{\&A, const std} \\ ::vector < \mbox{std::size\_t} > \mbox{\&subsys, std::size\_t} \mbox{ derived} > \mbox{\&A, const std} \\ ::vector < \mbox{std::size\_t} > \mbox{\&subsys, std::size\_t} \\ ::vector < \mbox{\&subsys, std::size\_t} > \mbox{\&subsys, std::size\_t} > \mbox{\&subsys, std::size\_t} \\ ::vector < \mbox{\&subsys, std::size\_t} > \mb$ 

Partial transpose.

• template<typename Derived >

System permutation.

• template<typename Derived >

DynMat< typename Derived::Scalar > syspermute (const Eigen::MatrixBase< Derived > &A, const std ← ::vector< std::size\_t > &perm, std::size\_t d=2)

System permutation.

template<typename Derived >

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

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

template<>

dmat rand (std::size t rows, std::size t cols, double a, double b)

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

template<>

cmat rand (std::size\_t rows, std::size\_t cols, double a, double b)

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

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

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

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

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

• template<typename Derived >

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

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

template<>

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

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

• template<>

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

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

• double randn (double mean=0, double sigma=1)

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

cmat randU (std::size t D)

Generates a random unitary matrix.

cmat randV (std::size t Din, std::size t Dout)

Generates a random isometry matrix.

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

Generates a set of random Kraus operators.

cmat randH (std::size t D)

Generates a random Hermitian matrix.

ket randket (std::size t D)

Generates a random normalized ket (pure state vector)

cmat randrho (std::size\_t D)

Generates a random density matrix.

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

Generates a random uniformly distributed permutation.

#### **Variables**

• constexpr double chop = 1e-10

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

constexpr double eps = 1e-12

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

• constexpr std::size\_t maxn = 64

Maximum number of qubits.

constexpr double pi = 3.141592653589793238462643383279502884

 $\pi$ 

• constexpr double ee = 2.718281828459045235360287471352662497

Base of natural logarithm, e.

constexpr std::size\_t infty = -1

Used to denote infinity.

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

qpp::Init const Singleton

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

qpp::Codes const Singleton

• const Gates & gt = Gates::get\_instance()

qpp::Gates const Singleton

• const States & st = States::get instance()

qpp::States const Singleton

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

qpp::RandomDevices Singleton

#### 6.1.1 Typedef Documentation

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

Complex (double precision) dynamic Eigen row vector.

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

Complex (double precision) dynamic Eigen matrix.

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

Complex number in double precision.

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

Real (double precision) dynamic Eigen matrix.

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

Dynamic Eigen column vector over the field specified by Scalar.

Example:

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

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

Dynamic Eigen matrix over the field specified by Scalar.

Example:

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

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

Dynamic Eigen row vector over the field specified by Scalar.

Example:

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

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

Complex (double precision) dynamic Eigen column vector.

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

Matrix absolut value.

#### **Parameters**

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

#### Returns

Matrix absolut value of A

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

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

#### **Parameters**

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

#### Returns

Real vector consisting of the range's absolut values squared

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

Computes the absolut values squared of a column vector.

#### **Parameters**

V	Eigen expression

#### Returns

Real vector consisting of the absolut values squared

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

### Adjoint.

### **Parameters**

Α	Eigen expression

#### Returns

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

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

### Anti-commutator.

Anti-commutator  $\{A,B\} = AB + BA$ . Both A and B must be Eigen expressions over the same scalar field.

#### **Parameters**

Α	Eigen expression
В	Eigen expression

#### Returns

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

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

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

#### Note

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

#### **Parameters**

state	Eigen expression
Α	Eigen expression
subsys	Subsystem indexes where the gate A is applied
dims	Dimensions of the multi-partite system

#### Returns

Gate A applied to the part subsys of state

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

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

#### Note

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

#### **Parameters**

state	Eigen expression
Α	Eigen expression
subsys	Subsystem indexes where the gate A is applied
d	Subsystem dimensions

#### Returns

Gate A applied to the part subsys of state

6.1.2.8 template < typename Derived1 , typename Derived2 > DynMat < typename Derived1::Scalar > qpp::applyCTRL ( const Eigen::MatrixBase < Derived2 > & A, const std::vector < std::size\_t > & ctrl, const std::vector < std::size\_t > & dims )

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

#### Note

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

#### **Parameters**

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

#### Returns

CTRL-A gate applied to the part subsys of state

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

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

#### Note

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

#### **Parameters**

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

#### Returns

CTRL-A gate applied to the part subsys of state

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

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

### **Parameters**

rho	Eigen expression
Ks	Set of Kraus operators

#### Returns

Output density matrix after the action of the channel

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

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

rho	Eigen expression
Ks	Set of Kraus operators
subsys	Subsystem indexes where the Kraus operators Ks are applied
dims	Dimensions of the multi-partite system

#### Returns

Output density matrix after the action of the channel

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

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

#### **Parameters**

rho	Eigen expression
Ks	Set of Kraus operators
subsys	Subsystem indexes where the Kraus operators Ks are applied
d	Subsystem dimensions

#### Returns

Output density matrix after the action of the channel

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

Choi matrix representation.

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

Note

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

#### **Parameters**

Ks	Set of Kraus operators

## Returns

Choi matrix representation

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

Extracts orthogonal Kraus operators from Choi matrix.

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

Note

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

Α	Choi matrix
---	-------------

## Returns

Set of Kraus operators

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

## Commutator.

Commutator [A,B] = AB - BA. Both A and B must be Eigen expressions over the same scalar field.

#### **Parameters**

Α	Eigen expression
В	Eigen expression

## Returns

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

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

Compose permutations.

## **Parameters**

perm	Permutation
sigma	Permutation

## Returns

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

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

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

Parameters

Α	Eigen expression

## Returns

Wootters concurrence

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

Complex conjugate.

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

## Returns

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

6.1.2.19 double qpp::contfrac2x ( const std::vector< int > & cf, std::size\_t n )

Real representation of a simple continued fraction.

## **Parameters**

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

#### Returns

Real representation of the simple continued fraction

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

Real representation of a simple continued fraction.

## **Parameters**

# Returns

Real representation of the simple continued fraction

 $6.1.2.21 \quad template < typename \ Derived > cmat \ qpp::cosm \ ( \ const \ Eigen::MatrixBase < Derived > \& \ \textit{A} \ )$ 

Matrix cos.

**Parameters** 

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

# Returns

Matrix cosine of A

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

Functor.

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

## Returns

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

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

## Determinant.

#### **Parameters**

Α	Eigen expression

## Returns

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

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

Eigen expression ostream manipulator.

#### **Parameters**

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

## Returns

Instance of qpp::internal::internal::IOManipEigen

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

Complex number ostream manipulator.

## Parameters

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

# Returns

Instance of qpp::internal::internal::IOManipEigen

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

Range ostream manipulator.

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

### **Returns**

Instance of qpp::internal::internal::IOManipRange

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

## **Parameters**

X	Container
separator	Separator
start	Left marking
end	Right marking

## Returns

Instance of qpp::internal::IOManipRange

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

C-style pointer ostream manipulator.

### **Parameters**

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

### Returns

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

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

Entanglement of the bi-partite pure state A.

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

## See also

qpp::shannon()

Α	Eigen expression
dims	Dimensions of the bi-partite system

## Returns

Entanglement, with the logarithm in base 2

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

Eigenvalues.

**Parameters** 

Α	Eigen expression
, ,	Eigen expression

#### Returns

Eigenvalues of A, as a complex dynamic column vector

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

Eigenvectors.

**Parameters** 

Α	Eigen expression

## Returns

Eigenvectors of A, as columns of a complex matrix

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

Matrix exponential.

**Parameters** 

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

# Returns

Matrix exponential of A

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

Functional calculus f(A)

Parameters

Α	Eigen expression

f Pointer-to-function from complex to complex

Returns

f(A)

6.1.2.34 std::size\_t qpp::gcd ( std::size\_t m, std::size\_t n )

Greatest common divisor of two non-negative integers.

#### **Parameters**

т	Non-negative integer
n	Non-negative integer

#### Returns

Greatest common divisor of m and n

6.1.2.35 std::size\_t qpp::gcd ( const std::vector < std::size\_t > & ns )

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

**Parameters** 

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

## Returns

Greatest common divisor of all numbers in ns

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

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

Note

Both local dimensions must be equal

Uses qpp::logdet() to avoid overflows

See also

qpp::logdet()

## Parameters

Α	Eigen expression

## Returns

G-concurrence

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

Gram-Schmidt orthogonalization.

Vs std::vector of Eigen expressions as column vectors

## Returns

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

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

Gram-Schmidt orthogonalization.

**Parameters** 

Vs std::initializer\_list of Eigen expressions as column vectors

#### Returns

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

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

Gram-Schmidt orthogonalization.

**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 as A

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

Hermitian eigenvalues.

**Parameters** 

A Eigen expression

### Returns

Eigenvalues of Hermitian A, as a real dynamic column vector

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

Hermitian eigenvectors.

**Parameters** 

A Eigen expression

Returns

Eigenvectors of Hermitian A, as columns of a complex matrix

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

Inverse.

**Parameters** 

A | Eigen expression

Returns

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

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

Inverse permutation.

**Parameters** 

perm | Permutation

Returns

Inverse of the permutation perm

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

Kronecker product.

Used to stop the recursion for the variadic template version of <a href="app::kron()">app::kron()</a>

**Parameters** 

head Eigen expression

Returns

Its argument head

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

Kronecker product.

**Parameters** 

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

## Returns

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

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

Kronecker product.

#### **Parameters**

As	std::vector of Eigen expressions

## Returns

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

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

Kronecker product.

#### **Parameters**

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

## Returns

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

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

Kronecker power.

## **Parameters**

Α	Eigen expression
n	Non-negative integer

#### Returns

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

6.1.2.49 std::size\_t qpp::lcm ( std::size\_t m, std::size\_t n )

Least common multiple of two positive integers.

т	Positive integer
n	Positive integer

## Returns

Least common multiple of m and n

6.1.2.50 std::size\_t qpp::lcm ( const std::vector< std::size\_t > & ns )

Least common multiple of a list of positive integers.

#### **Parameters**

ns	List of positive integers

#### Returns

Least common multiple of all numbers in ns

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

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

The template parameter cannot be automatically deduced and must be explicitly provided, depending on the scalar field of the matrix that is being loaded.

#### Example:

```
// loads a previously saved Eigen dynamic complex matrix from "input.bin"
auto mat = load<cmat>("input.bin");
```

## See also

qpp::loadMATLABmatrix()

#### **Parameters**

Α	Eigen expression
fname	Output file name

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

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

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

6.1.2.53 template<> dmat qpp::loadMATLABmatrix ( const std::string & mat\_file, const std::string & var\_name )
[inline]

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

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

Example:

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

## Note

If var name is a complex matrix, only the real part is loaded

#### **Parameters**

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

### Returns

Eigen double dynamic matrix (qpp::dmat)

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

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

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

## Example:

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

## **Parameters**

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

## Returns

Eigen complex dynamic matrix (qpp::cmat)

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

Logarithm of the determinant.

Useful when the determinant overflows/underflows

## **Parameters**

Α	Eigen expression

#### Returns

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

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

Matrix logarithm.

A	Eigen expression
/ ·	Ligen expression

## Returns

Matrix logarithm of A

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

Logarithmic negativity of the bi-partite mixed state A.

#### **Parameters**

Α	Eigen expression
dims	Dimensions of the bi-partite system

#### Returns

Logarithmic negativity, with the logarithm in base 2

Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.

### Note

The dimension of all Ks must match the dimension of subsys.

## **Parameters**

Α	Eigen expression
Ks	Set of Kraus operators
subsys	Subsystem indexes that are measured
dims	Dimensions of the multi-partite system

#### Returns

Tuple consisiting of the result of the measurement, the vector of outcome probabilities and the vector of postmeasurement normalized states

6.1.2.59 template < typename Derived > std::tuple < std::size\_t, std::vector < double >, std::vector < cmat > > qpp::measure ( const Eigen::MatrixBase < Derived > & A, const std::initializer\_list < cmat > & Ks, const std::vector < std::size\_t > & subsys, const std::vector < std::size\_t > & dims )

Measures the part *subsys* of the multi-partite state vector or density matrix *A* using the set of Kraus operators *Ks*.

## Note

The dimension of all Ks must match the dimension of subsys.

Α	Eigen expression
subsys	Subsystem indexes that are measured
dims	Dimensions of the multi-partite system
Ks	Set of Kraus operators

#### Returns

Tuple consisiting of the result of the measurement, the vector of outcome probabilities and the vector of postmeasurement normalized states

6.1.2.60 template < typename Derived > std::tuple < std::size\_t, std::vector < double >, std::vector < cmat > > qpp::measure ( const Eigen::MatrixBase < Derived > & A, const std::vector < cmat > & Ks, const std::vector < std::size\_t > & subsys, const std::size\_t d = 2)

Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.

#### Note

The dimension of all Ks must match the dimension of subsys.

#### **Parameters**

Α	Eigen expression
subsys	Subsystem indexes that are measured
d	Subsystem dimensions
Ks	Set of Kraus operators

### Returns

Tuple consisiting of the result of the measurement, the vector of outcome probabilities and the vector of post-measurement normalized states

6.1.2.61 template < typename Derived > std::tuple < std::size\_t, std::vector < double >, std::vector < cmat > > qpp::measure ( const Eigen::MatrixBase < Derived > & A, const std::initializer\_list < cmat > & Ks, const std::vector < std::size\_t > & subsys, const std::size\_t d = 2)

Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.

# Note

The dimension of all Ks must match the dimension of subsys.

## **Parameters**

A	Eigen expression
subsys	Subsystem indexes that are measured
d	Subsystem dimensions
Ks	Set of Kraus operators

## Returns

Tuple consisiting of the result of the measurement, the vector of outcome probabilities and the vector of post-measurement normalized states

6.1.2.62 template < typename Derived > std::tuple < std::size\_t, std::vector < double >, std::vector < cmat > > qpp::measure ( const Eigen::MatrixBase < Derived > & A, const cmat & U, const std::vector < std::size\_t > & subsys, const std::vector < std::size\_t > & dims )

Measures the part subsys of the multi-partite state A in the orthonormal basis specified by the unitary matrix U.

#### Note

The dimension of *U* must match the dimension of *subsys*.

#### **Parameters**

Α	Eigen expression
subsys	Subsystem indexes that are measured
dims	Dimensions of the multi-partite system
U	Unitary matrix whose columns represent the measurement basis vectors

#### Returns

Tuple consisiting of the result of the measurement, the vector of outcome probabilities and the vector of postmeasurement normalized states

6.1.2.63 template < typename Derived > std::tuple < std::size\_t, std::vector < double >, std::vector < cmat > > qpp::measure ( const Eigen::MatrixBase < Derived > & A, const cmat & U, const std::vector < std::size\_t > & subsys, const std::size\_t d = 2)

Measures the part subsys of the multi-partite state A in the orthonormal basis specified by the unitary matrix U.

## Note

The dimension of *U* must match the dimension of *subsys*.

### **Parameters**

Α	Eigen expression
subsys	Subsystem indexes that are measured
d	Subsystem dimensions
U	Unitary matrix whose columns represent the measurement basis vectors

## Returns

Tuple consisiting of the result of the measurement, the vector of outcome probabilities and the vector of postmeasurement normalized states

6.1.2.64 template<typename Derived > std::tuple<std::size\_t, std::vector<double>, std::vector<cmat> > qpp::measure ( const Eigen::MatrixBase< Derived > & A, const std::vector< cmat > & Ks )

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

#### **Parameters**

Α	Eigen expression

17	0 + (1/
KS	Set of Kraus operators
713	Cot of Mado opolatoro

## Returns

Tuple consisiting of the result of the measurement, the vector of outcome probabilities and the vector of postmeasurement normalized states

6.1.2.65 template<typename Derived > std::tuple<std::size\_t, std::vector<double>, std::vector<cmat> > qpp::measure ( const Eigen::MatrixBase< Derived > & A, const std::initializer\_list< cmat > & Ks )

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

#### **Parameters**

Α	Eigen expression
Ks	Set of Kraus operators

#### Returns

Tuple consisiting of the result of the measurement, the vector of outcome probabilities and the vector of post-measurement normalized states

6.1.2.66 template < typename Derived > std::tuple < std::size\_t, std::vector < double > , std::vector < cmat > > qpp::measure ( const Eigen::MatrixBase < Derived > & A, const cmat & U)

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

## **Parameters**

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

### Returns

Tuple consisiting of the result of the measurement, the vector of outcome probabilities and the vector of post-measurement normalized states

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

Multi-partite qudit ket.

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

#### **Parameters**

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

### Returns

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

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

Multi-partite qudit ket.

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

mask	std::vector of non-negative integers
d	Subsystem dimensions

#### Returns

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

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

Projector onto multi-partite qudit ket.

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

#### **Parameters**

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

#### Returns

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

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

Projector onto multi-partite qudit ket.

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

### **Parameters**

mask	std::vector of non-negative integers
d	Subsystem dimensions

## Returns

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

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

Multi-index to non-negative integer index.

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

## **Parameters**

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

## Returns

Non-negative integer index

 $6.1.2.72 \quad \text{std::vector} < \text{std::size\_t} > \text{qpp::n2multiidx ( std::size\_t } \textit{n, const std::vector} < \text{std::size\_t} > \text{\& dims )}$ 

Non-negative integer index to multi-index.

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

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

## Returns

Multi-index of the same size as dims

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

Negativity of the bi-partite mixed state A.

### **Parameters**

Α	Eigen expression
dims	Dimensions of the bi-partite system

## Returns

Negativity

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

Frobenius norm.

## **Parameters**

Α	Eigen expression

## Returns

Frobenius norm of A, as a real number

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

D-th root of unity.

## **Parameters**

D	Non-negative integer

# Returns

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

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

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

## Example:

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

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

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

Example:

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

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

Matrix power.

Explicitly multiplies the matrix *A* with itself *n* times. By convention  $A^0 = I$ .

## **Parameters**

Α	Eigen expression
n	Non-negative integer

#### Returns

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

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

Projector.

Normalized projector onto state vector

**Parameters** 

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

## Returns

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

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

Element-wise product of A.

**Parameters** 

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

### Returns

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

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

Element-wise product of a range.

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

## Returns

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

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

Partial trace.

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

#### **Parameters**

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

## Returns

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

Partial trace.

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

## **Parameters**

Α	Eigen expression
subsys	Subsystem indexes
d	Subsystem dimensions

# Returns

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

6.1.2.84 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** 

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

#### Returns

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

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

#### Partial trace.

#### **Parameters**

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

## Returns

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

## Partial transpose.

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

## **Parameters**

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

### Returns

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

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

## Partial transpose.

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

## Parameters

Α	Eigen expression
subsys	Subsystem indexes
d	Subsystem dimensions

# Returns

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

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

Quantum mutual information between 2 subsystems of a composite system.

Α	Eigen expression
subsysA	Indexes of the first subsystem
subsysB	Indexes of the second subsystem
dims	Dimensions of the multi-partite system

#### Returns

Mutual information between the 2 subsystems

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

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

If complex, then both real and imaginary parts are uniformly distributed in [a, b)

This is the generic version that always throws *qpp::Exception::Type::UNDEFINED\_TYPE*. It is specialized only for *qpp::dmat* and *qpp::cmat* 

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

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

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

### Example:

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

# **Parameters**

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

#### Returns

Random real matrix

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

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

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

## Example:

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

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

## Returns

Random complex matrix

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

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

#### **Parameters**

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

#### Returns

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

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

Generates a random Hermitian matrix.

## **Parameters**

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

## Returns

Random Hermitian matrix

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

# **Parameters**

а	Beginning of the interval, belongs to it
Ь	End of the interval, belongs to it

## Returns

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

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

Generates a random normalized ket (pure state vector)

-		
ſ	D	Dimension of the Hilbert space

## Returns

Random normalized ket

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

Generates a set of random Kraus operators.

Note

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

#### **Parameters**

N	Number of Kraus operators
D	Dimension of the Hilbert space

## Returns

Set of *N* Kraus operators satisfying the closure condition

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

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

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

This is the generic version that always throws *qpp::Exception::Type::UNDEFINED\_TYPE*. It is specialized only for *qpp::dmat* and *qpp::cmat* 

6.1.2.98 template<> dmat qpp::randn ( std::size\_t rows, std::size\_t cols, double mean, double sigma ) [inline]

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

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

## Example:

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

## Parameters

rows	Number of rows of the random generated matrix
cols	Number of columns of the random generated matrix
mean	Mean

sigma	Standard deviation

#### Returns

Random real matrix

6.1.2.99 template<> cmat qpp::randn ( std::size\_t rows, std::size\_t cols, double mean, double sigma ) [inline]

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

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

#### Example:

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

## **Parameters**

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

#### Returns

Random complex matrix

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

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

#### **Parameters**

mean	Mean
sigma	Standard deviation

## Returns

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

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

Generates a random uniformly distributed permutation.

Uses Knuth's shuffle method (as implemented by std::shuffle), so that all permutations are equally probable

# **Parameters**

n	Size of the permutation
	·

# Returns

Random permutation of size n

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

Generates a random density matrix.

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

## Returns

Random density matrix

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

Generates a random unitary matrix.

**Parameters** 

D	Dimension of the Hilbert space

# Returns

Random unitary

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

Generates a random isometry matrix.

#### **Parameters**

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

## Returns

Random isometry matrix

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

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

## **Parameters**

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

## Returns

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

6.1.2.106 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)

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

#### Returns

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

6.1.2.107 template<typename Derived > DynColVect<typename Derived::Scalar> qpp::rho2pure ( const Eigen::MatrixBase< Derived > & A )

Finds the pure state representation of a matrix proportional to a projector onto a pure state.

#### Note

No purity check is done, the input state A must have rank one, otherwise the function returs the first non-zero eigenvector of A

#### **Parameters**

Α	Eigen expression, assumed to be proportional to a projector onto a pure state, i.e. A is
	assumed to have rank one

#### Returns

The unique non-zero eigenvector of A, as a dynamic column vector over the same scalar field as A

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

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

## See also

qpp::saveMATLABmatrix()

### **Parameters**

Α	Eigen expression
fname	Output file name

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

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

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

6.1.2.110 template<> void qpp::saveMATLABmatrix ( const Eigen::MatrixBase< dmat > & A, const std::string & mat\_file, const std::string & var\_name, const std::string & mode ) [inline]

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

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

6.1.2.111 template<> void qpp::saveMATLABmatrix ( const Eigen::MatrixBase< cmat > & A, const std::string & mat\_file, const std::string & war\_name, const std::string & mode ) [inline]

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

#### **Parameters**

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

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

## Schatten norm.

#### **Parameters**

Α	Eigen expression
р	Integer, greater or equal to 1

### Returns

Schatten-p norm of A, as a real number

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

Schmidt coefficients of the bi-partite pure state A.

Note

The sum of the squares of the Schmidt coefficients equals 1

## See also

qpp::schmidtprob()

## **Parameters**

Α	Eigen expression
dims	Dimensions of the bi-partite system

# Returns

Schmidt coefficients of A, as a complex dynamic column vector

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

Schmidt probabilities of the bi-partite pure state A.

Defined as the squares of the Schmidt coefficients. The sum of the Schmidt probabilities equals 1.

#### See also

qpp::schmidtcoeff()

## **Parameters**

Α	Eigen expression
dims	Dimensions of the bi-partite system

#### Returns

Schmidt probabilites of A, as a real dynamic column vector

6.1.2.115 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**

Α	Eigen expression
dims	Dimensions of the bi-partite system

# Returns

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

6.1.2.116 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**

A	Eigen expression
dims	Dimensions of the bi-partite system

## Returns

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

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

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

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

## Returns

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

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

Matrix sin.

**Parameters** 

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

## Returns

Matrix sine of A

6.1.2.119 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** 

Α	Eigen expression
Z	Complex number

## Returns

Matrix power  $A^z$ 

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

Matrix square root.

**Parameters** 

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

## Returns

Matrix square root of A

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

Element-wise sum of A.

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

## Returns

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

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

Element-wise sum of a range.

#### **Parameters**

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

## Returns

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

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

Superoperator matrix representation.

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

## **Parameters**

Ks	Set of Kraus operators
----	------------------------

## Returns

Superoperator matrix representation

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

Singular values.

#### **Parameters**

Α	Eigen expression

## Returns

Singular values of A, as a real dynamic column vector

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

Left singular vectors.

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

## Returns

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

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

Right singular vectors.

#### **Parameters**

Α	Eigen expression

#### Returns

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

6.1.2.127 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**

Α	Eigen expression
perm	Permutation
dims	Dimensions of the multi-partite system

# Returns

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

6.1.2.128 template<typename Derived > DynMat<typename Derived::Scalar> qpp::syspermute ( const Eigen::MatrixBase< Derived > & A, const std::vector< std::size\_t > & perm, std::size\_t d = 2 )

System permutation.

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

#### **Parameters**

Α	Eigen expression
perm	Permutation
d	Subsystem dimensions

# Returns

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

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

Trace.

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

#### Returns

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

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

Transpose.

## **Parameters**

Α	Eigen expression

## Returns

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

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

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

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

Α	Eigen expression, representing a probability distribution (real dynamic column vector) or a density matrix (complex dynamic matrix)
alpha	Non-negative real number

### Returns

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

6.1.2.132 std::vector<int> qpp::x2contfrac ( double x, std::size\_t n, std::size\_t cut = 1e5 )

Simple continued fraction expansion.

#### **Parameters**

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

### Returns

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

# 6.1.3 Variable Documentation

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

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

```
6.1.3.2 const Codes& qpp::codes = Codes::get_instance()
qpp::Codes const Singleton
Initializes the codes, see the class qpp::Codes
6.1.3.3 constexpr double qpp::ee = 2.718281828459045235360287471352662497
Base of natural logarithm, e.
6.1.3.4 constexpr double qpp::eps = 1e-12
Used to decide whether a number or expression in double precision is zero or not.
Example:
if(std::abs(x) < qpp::eps) // x is zero</pre>
6.1.3.5 const Gates& qpp::gt = Gates::get_instance()
qpp::Gates const Singleton
Initializes the gates, see the class qpp::Gates
6.1.3.6 constexpr std::size_t qpp::infty = -1
Used to denote infinity.
6.1.3.7 const Init& qpp::init = Init::get_instance()
qpp::Init const Singleton
Additional initializations/cleanups
6.1.3.8 constexpr std::size_t qpp::maxn = 64
Maximum number of qubits.
Used internally to allocate arrays on the stack (for speed reasons)
6.1.3.9 constexpr double qpp::pi = 3.141592653589793238462643383279502884
\pi
6.1.3.10 RandomDevices& qpp::rdevs = RandomDevices::get_instance()
qpp::RandomDevices Singleton
Initializes the random devices, see the class qpp::RandomDevices
6.1.3.11 const States& qpp::st = States::get_instance()
qpp::States const Singleton
Initializes the states, see the class qpp::States
```

# 6.2 qpp::experimental Namespace Reference

#### 6.2.1 Detailed Description

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

# 6.3 qpp::internal Namespace Reference

#### Classes

- class IOManipEigen
- class IOManipPointer
- class IOManipRange
- · class Singleton

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

#### **Functions**

- void \_n2multiidx (std::size\_t n, std::size\_t numdims, const std::size\_t \*dims, std::size\_t \*result)
- std::size\_t \_multiidx2n (const std::size\_t \*midx, std::size\_t numdims, const std::size\_t \*dims)
- template<typename Derived >

bool <u>\_check\_square\_mat</u> (const Eigen::MatrixBase< Derived > &A)

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

bool <u>\_check\_vector</u> (const Eigen::MatrixBase< Derived > &A)

template<typename Derived >

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

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

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

- template<typename T >
  - bool <u>\_check\_nonzero\_size</u> (const T &x)
- bool <u>\_check\_dims</u> (const std::vector < std::size\_t > &dims)
- $\bullet \ \ \mathsf{template}{<}\mathsf{typename} \ \mathsf{Derived}>$

bool \_check\_dims\_match\_mat (const std::vector< std::size\_t > &dims, const Eigen::MatrixBase< Derived > &A)

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

 $\label{local_bool} \begin{tabular}{ll} bool\_check\_dims\_match\_cvect (const std::vector < std::size\_t > \&dims, const Eigen::MatrixBase < Derived > \&V) \end{tabular}$ 

• template<typename Derived >

 $\label{local_check_dims_match_rvect} bool\_check\_dims\_match\_rvect \ (const \ std::vector < std::size\_t > \&dims, \ const \ Eigen::MatrixBase < Derived > \&V)$ 

- 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 >

 $\begin{tabular}{ll} DynMat &< typename \ Derived 1 :: Scalar > \_kron 2 \ (const \ Eigen :: MatrixBase &< Derived 1 > \&A, \ const \ Eigen :: $$ MatrixBase &< Derived 2 > \&B ) \end{tabular}$ 

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

void variadic vector emplace (std::vector< T > &)

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

# 6.3.1 Detailed Description

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

- 6.3.2 Function Documentation
- 6.3.2.1 template < typename Derived > bool qpp::internal::\_check\_col\_vector ( const Eigen::MatrixBase < Derived > & A )
- 6.3.2.2 bool qpp::internal::\_check\_dims ( const std::vector < std::size\_t > & dims )
- 6.3.2.3 template<typename Derived > bool qpp::internal::\_check\_dims\_match\_cvect ( const std::vector< std::size\_t > & dims, const Eigen::MatrixBase< Derived > & V )
- 6.3.2.4 template<typename Derived > bool qpp::internal::\_check\_dims\_match\_mat ( const std::vector< std::size\_t > & dims, const Eigen::MatrixBase< Derived > & A )
- 6.3.2.5 template<typename Derived > bool qpp::internal::\_check\_dims\_match\_rvect ( const std::vector< std::size\_t > & dims, const Eigen::MatrixBase< Derived > & V )
- 6.3.2.6 bool qpp::internal::\_check\_eq\_dims ( const std::vector < std::size\_t > & dims, std::size\_t dim )
- 6.3.2.7 template < typename T > bool qpp::internal::\_check\_nonzero\_size ( const T & x )
- 6.3.2.8 bool gpp::internal::\_check\_perm ( const std::vector < std::size\_t > & perm )
- 6.3.2.9 template < typename Derived > bool qpp::internal::\_check\_row\_vector ( const Eigen::MatrixBase < Derived > & A )
- $6.3.2.10 \quad template < type name\ Derived > bool\ qpp::internal::\_check\_square\_mat\ (\ const\ Eigen::MatrixBase < Derived > \&\ A\ )$
- 6.3.2.11 bool qpp::internal::\_check\_subsys\_match\_dims ( const std::vector< std::size\_t > & subsys, const std::vector< std::size\_t > & dims )
- 6.3.2.12 template < typename Derived > bool qpp::internal::\_check\_vector ( const Eigen::MatrixBase < Derived > & A )
- 6.3.2.14 std::size\_t qpp::internal::\_multiidx2n ( const std::size\_t \* midx, std::size\_t numdims, const std::size\_t \* dims ) [inline]
- 6.3.2.15 void qpp::internal::\_n2multiidx ( std::size\_t n, std::size\_t
- 6.3.2.16 template<typename T > void qpp::internal::variadic\_vector\_emplace ( std::vector< T > & )
- $\begin{array}{ll} \textbf{6.3.2.17} & \textbf{template} \negthinspace < \negthinspace \textbf{typename} \ \textbf{T} \ , \ \textbf{typename...} \ \textbf{Args} \negthinspace > \negthinspace \textbf{void} \ \textbf{qpp::internal::variadic\_vector\_emplace} \ ( \\ \textbf{std::vector} \negthinspace < \negthinspace \textbf{T} \negthinspace > \negthinspace \textbf{\&} \ \textit{v}, \ \textbf{First} \ \&\& \ \textit{first}, \ \textbf{Args} \ \&\&... \ \textit{args} \ ) \end{array}$

Namespace	Docume	entation

# **Chapter 7**

# **Class Documentation**

# 7.1 qpp::Codes Class Reference

const Singleton class that defines quantum error correcting codes

#include <codes.h>

Inheritance diagram for qpp::Codes:



Collaboration diagram for qpp::Codes:



# **Public Types**

enum Type { Type::FIVE\_QUBIT = 1, Type::SEVEN\_QUBIT\_STEANE, Type::NINE\_QUBIT\_SHOR }
 Code types, add more codes here if needed.

## **Public Member Functions**

ket codeword (Type type, std::size\_t i) const
 Returns the codeword of the specified code.

#### **Private Member Functions**

• Codes ()

Default constructor.

#### **Friends**

class internal::Singleton < const Codes >

#### **Additional Inherited Members**

## 7.1.1 Detailed Description

const Singleton class that defines quantum error correcting codes

#### 7.1.2 Member Enumeration Documentation

```
7.1.2.1 enum qpp::Codes::Type [strong]
```

Code types, add more codes here if needed.

See also

```
qpp::Codes::codeword()
```

#### **Enumerator**

```
FIVE_QUBIT [[5,1,3]] qubit code
SEVEN_QUBIT_STEANE [[7,1,3]] Steane qubit code
NINE_QUBIT_SHOR [[9,1,3]] Shor qubit code
```

## 7.1.3 Constructor & Destructor Documentation

```
7.1.3.1 qpp::Codes::Codes( ) [inline],[private]
```

Default constructor.

## 7.1.4 Member Function Documentation

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

Returns the codeword of the specified code.

#### **Parameters**

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

## Returns

i-th codeword of the code type

## 7.1.5 Friends And Related Function Documentation

**7.1.5.1 friend class internal::Singleton**< **const Codes** > [friend]

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

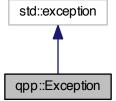
• include/classes/codes.h

# 7.2 qpp::Exception Class Reference

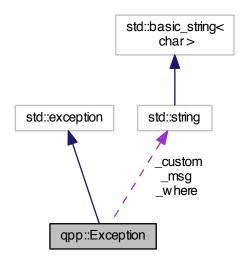
Generates custom exceptions, used when validating function parameters.

#include <exception.h>

Inheritance diagram for qpp::Exception:



Collaboration diagram for qpp::Exception:



# **Public Types**

enum Type {

Type::UNKNOWN\_EXCEPTION = 1, Type::ZERO\_SIZE, Type::MATRIX\_NOT\_SQUARE, Type::MATRIX\_← NOT\_CVECTOR,

Type::MATRIX\_NOT\_RVECTOR, Type::MATRIX\_NOT\_VECTOR, Type::MATRIX\_NOT\_SQUARE\_OR\_C↔ VECTOR, Type::MATRIX\_NOT\_SQUARE\_OR\_RVECTOR,

Type::MATRIX\_NOT\_SQUARE\_OR\_VECTOR, Type::MATRIX\_MISMATCH\_SUBSYS, Type::DIMS\_INVA← LID, Type::DIMS\_NOT\_EQUAL,

Type::DIMS\_MISMATCH\_MATRIX, Type::DIMS\_MISMATCH\_CVECTOR, Type::DIMS\_MISMATCH\_RVE← CTOR, Type::DIMS MISMATCH VECTOR,

Type::SUBSYS\_MISMATCH\_DIMS, Type::NOT\_QUBIT\_GATE, Type::NOT\_QUBIT\_SUBSYS, Type::NOT← \_BIPARTITE,

Type::NO\_CODEWORD, Type::PERM\_INVALID, Type::OUT\_OF\_RANGE, Type::TYPE\_MISMATCH, Type::UNDEFINED\_TYPE, Type::CUSTOM\_EXCEPTION }

Exception types, add more exceptions here if needed.

#### **Public Member Functions**

• Exception (const std::string &where, const Type &type)

Constructs an exception.

• Exception (const std::string &where, const std::string &custom)

Constructs an exception.

• virtual const char \* what () const noexceptoverride

Overrides std::exception::what()

#### **Private Member Functions**

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

Constructs the exception's description from its type.

#### **Private Attributes**

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

### 7.2.1 Detailed Description

Generates custom exceptions, used when validating function parameters.

Customize this class if more exceptions are needed

#### 7.2.2 Member Enumeration Documentation

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

Exception types, add more exceptions here if needed.

#### See also

```
qpp:Exception::_construct_exception_msg()
```

#### Enumerator

UNKNOWN EXCEPTION Unknown exception

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

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

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

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

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

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

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

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

**MATRIX\_MISMATCH\_SUBSYS** Matrix size mismatch subsystem sizes (e.g. in qpp::apply(), or qpp← ::channel())

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 of 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 of std::vector<std::size\_t> is not equal to the number of columns of Eigen::Matrix (assumed to be a column vector)

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

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

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

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

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

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

OUT\_OF\_RANGE Parameter out of range

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

UNDEFINED\_TYPE Templated function not defined for this type

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

#### 7.2.3 Constructor & Destructor Documentation

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

Constructs an exception.

#### **Parameters**

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

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

Constructs an exception.

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

#### **Parameters**

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

## 7.2.4 Member Function Documentation

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

Constructs the exception's description from its type.

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

Returns

Exception's description

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

Overrides std::exception::what()

Returns

Exception's description

#### 7.2.5 Member Data Documentation

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

- **7.2.5.2 std::string qpp::Exception::\_msg** [private]
- 7.2.5.3 Type qpp::Exception::\_type [private]
- **7.2.5.4** std::string qpp::Exception::\_where [private]

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

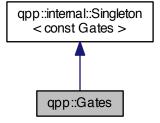
• include/classes/exception.h

# 7.3 qpp::Gates Class Reference

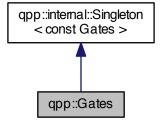
const Singleton class that implements most commonly used gates

#include <gates.h>

Inheritance diagram for qpp::Gates:



Collaboration diagram for qpp::Gates:



# **Public Member Functions**

cmat Rn (double theta, std::vector< double > n) const
 Rotation of theta about the 3-dimensional real unit vector n.

```
    cmat Zd (std::size_t D) const

          Generalized Z gate for qudits.
    • cmat Fd (std::size t D) const
          Fourier transform gate for qudits.

    cmat Xd (std::size_t D) const

          Generalized X gate for qudits.
    • template<typename Derived = Eigen::MatrixXcd>
      Derived Id (std::size_t D) const
          Identity gate.
    • template<typename Derived >
      DynMat< typename Derived::Scalar > CTRL (const Eigen::MatrixBase< Derived > &A, const std::vector<
      std::size_t > &ctrl, const std::vector< std::size_t > &subsys, std::size_t n, std::size_t d=2) const
          Generates the multi-partite multiple-controlled-A gate in matrix form.
    template<typename Derived >
      DynMat< typename Derived::Scalar > expandout (const Eigen::MatrixBase< Derived > &A, std::size t pos,
      const std::vector< std::size_t > &dims) const
          Expands out.
Public Attributes
    • cmat Id2 {cmat::Identity(2, 2)}
          Identity gate.

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

          Hadamard gate.

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

          Pauli Sigma-X gate.

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

          Pauli Sigma-Y gate.

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

          Pauli Sigma-Z gate.
    • cmat S {cmat::Zero(2, 2)}
          S gate.
    cmat T {cmat::Zero(2, 2)}
          T gate.

    cmat CNOTab {cmat::Identity(4, 4)}

          Controlled-NOT control target gate.

    cmat CZ {cmat::Identity(4, 4)}

          Controlled-Phase gate.
    cmat CNOTba {cmat::Zero(4, 4)}
          Controlled-NOT target control gate.

    cmat SWAP {cmat::Identity(4, 4)}

          SWAP gate.

    cmat TOF {cmat::Identity(8, 8)}

          Toffoli gate.

    cmat FRED {cmat::ldentity(8, 8)}

          Fredkin gate.
```

# **Private Member Functions**

• Gates ()

Initializes the gates.

#### **Friends**

class internal::Singleton < const Gates >

#### **Additional Inherited Members**

# 7.3.1 Detailed Description

const Singleton class that implements most commonly used gates

#### 7.3.2 Constructor & Destructor Documentation

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

Initializes the gates.

#### 7.3.3 Member Function Documentation

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

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

Note

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

#### **Parameters**

Α	Eigen expression
ctrl	Control subsystem indexes
subsys	Subsystem indexes where the gate A is applied
n	Total number of subsystes
d	Subsystem dimensions

#### Returns

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

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

## Expands out.

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

#### **Parameters**

Α	Eigen expression
pos	Position

dims Dimensions of the multi-partite system

#### Returns

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

7.3.3.3 cmat qpp::Gates::Fd ( std::size\_t D ) const [inline]

Fourier transform gate for qudits.

Note

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

#### **Parameters**

D	Dimension of the Hilbert space

#### Returns

Fourier transform gate for qudits

7.3.3.4 template<typename Derived = Eigen::MatrixXcd> Derived qpp::Gates::Id ( std::size\_t D ) const [inline] Identity gate.

Note

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

#### **Parameters**

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

## Returns

Identity gate

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

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

#### **Parameters**

theta	Rotation angle
n	3-dimensional real unit vector

## Returns

Rotation gate

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

Generalized X gate for qudits.

Note

```
Defined as X = \sum_j |j \oplus 1\rangle\langle j|
```

#### **Parameters**

```
D Dimension of the Hilbert space
```

#### Returns

Generalized X gate for qudits

7.3.3.7 cmat qpp::Gates::Zd ( std::size\_t D ) const [inline]

Generalized Z gate for qudits.

Note

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

#### **Parameters**

D Dimension of the Hilbert space

#### Returns

Generalized Z gate for qudits

- 7.3.4 Friends And Related Function Documentation
- **7.3.4.1** friend class internal::Singleton < const Gates > [friend]
- 7.3.5 Member Data Documentation
- 7.3.5.1 cmat qpp::Gates::CNOTab {cmat::Identity(4, 4)}

Controlled-NOT control target gate.

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

Controlled-NOT target control gate.

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

Controlled-Phase gate.

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

Fredkin gate.

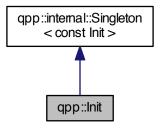
```
7.3.5.5 cmat qpp::Gates::H {cmat::Zero(2, 2)}
Hadamard gate.
7.3.5.6 cmat qpp::Gates::ld2 {cmat::ldentity(2, 2)}
Identity gate.
7.3.5.7 cmat qpp::Gates::S {cmat::Zero(2, 2)}
S gate.
7.3.5.8 cmat qpp::Gates::SWAP {cmat::Identity(4, 4)}
SWAP gate.
7.3.5.9 cmat qpp::Gates::T {cmat::Zero(2, 2)}
T gate.
7.3.5.10 cmat qpp::Gates::TOF {cmat::Identity(8, 8)}
Toffoli gate.
7.3.5.11 cmat qpp::Gates::X {cmat::Zero(2, 2)}
Pauli Sigma-X gate.
7.3.5.12 cmat qpp::Gates::Y {cmat::Zero(2, 2)}
Pauli Sigma-Y gate.
7.3.5.13 cmat qpp::Gates::Z {cmat::Zero(2, 2)}
Pauli Sigma-Z gate.
The documentation for this class was generated from the following file:
    • include/classes/gates.h
```

# 7.4 qpp::Init Class Reference

const Singleton class that performs additional initializations/cleanups

#include <init.h>

Inheritance diagram for qpp::Init:



Collaboration diagram for qpp::Init:



# **Public Member Functions**

• Init ()

Additional initializations.

# **Private Member Functions**

• ∼Init ()

Cleanups.

# **Friends**

- class internal::Singleton < const Init >

# **Additional Inherited Members**

# 7.4.1 Detailed Description

const Singleton class that performs additional initializations/cleanups

#### 7.4.2 Constructor & Destructor Documentation

```
7.4.2.1 qpp::Init::Init() [inline]
```

Additional initializations.

```
7.4.2.2 qpp::Init::~Init() [inline], [private]
```

Cleanups.

#### 7.4.3 Friends And Related Function Documentation

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

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

· include/classes/init.h

# 7.5 qpp::internal::IOManipEigen Class Reference

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

### **Public Member Functions**

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

### **Private Attributes**

- cmat \_A
- · double \_chop

#### **Friends**

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

## 7.5.1 Constructor & Destructor Documentation

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

### 7.5.2 Friends And Related Function Documentation

- 7.5.2.1 template<typename charT, typename traits > std::basic\_ostream<charT, traits>& operator<< (
  std::basic\_ostream< charT, traits > & os, const IOManipEigen & rhs) [friend]
- 7.5.3 Member Data Documentation
- **7.5.3.1 cmat qpp::internal::IOManipEigen::\_A** [private]
- **7.5.3.2** double qpp::internal::IOManipEigen::\_chop [private]

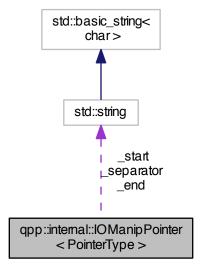
The documentation for this class was generated from the following file:

• include/internal/classes/iomanip.h

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

#include <iomanip.h>

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



#### **Public Member Functions**

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

## **Private Attributes**

- const PointerType \* p
- std::size\_t \_n

- std::string \_separator
- std::string start
- · std::string \_end

#### **Friends**

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

#### 7.6.1 Constructor & Destructor Documentation

- 7.6.1.1 template < typename PointerType > qpp::internal::IOManipPointer < PointerType >::IOManipPointer ( const PointerType \* p, const std::string & separator, const std::string & start = " [ ", const std::string & end = " ] " ) [inline], [explicit]
- 7.6.1.2 template<typename PointerType> qpp::internal::IOManipPointer< PointerType>::IOManipPointer( const IOManipPointer<< PointerType> & ) [default]
- 7.6.2 Member Function Documentation
- 7.6.2.1 template<typename PointerType> IOManipPointer& qpp::internal::IOManipPointer< PointerType
  >::operator= ( const IOManipPointer< PointerType > & ) [ default ]
- 7.6.3 Friends And Related Function Documentation
- 7.6.3.1 template<typename PointerType> template<typename charT , typename traits > std::basic\_ostream<charT, traits>& operator<< ( std::basic\_ostream< charT, traits > & os, const IOManipPointer< PointerType > & rhs )
  [friend]
- 7.6.4 Member Data Documentation
- 7.6.4.1 template<typename PointerType> std::string qpp::internal::IOManipPointer< PointerType>::\_end [private]
- 7.6.4.2 template<typename PointerType> std::size\_t qpp::internal::IOManipPointer< PointerType >::\_n [private]
- 7.6.4.3 template<typename PointerType> const PointerType\* qpp::internal::IOManipPointer< PointerType >::\_p [private]
- 7.6.4.4 template<typename PointerType> std::string qpp::internal::IOManipPointer< PointerType >::\_separator [private]
- 7.6.4.5 template<typename PointerType> std::string qpp::internal::IOManipPointer< PointerType >::\_start [private]

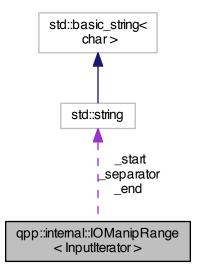
The documentation for this class was generated from the following file:

• include/internal/classes/iomanip.h

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

#include <iomanip.h>

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



#### **Public Member Functions**

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

### **Private Attributes**

- · InputIterator\_first
- InputIterator \_last
- std::string \_separator
- std::string \_start
- std::string \_end

## Friends

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

# 7.7.1 Constructor & Destructor Documentation

### 7.7.2 Friends And Related Function Documentation

7.7.2.1 template < typename InputIterator > template < typename charT , typename traits > std::basic\_ostream < charT, traits > & os, const IOManipRange < InputIterator > & rhs ) [friend]

#### 7.7.3 Member Data Documentation

- 7.7.3.1 template<typename InputIterator > std::string qpp::internal::IOManipRange< InputIterator >::\_end [private]
- 7.7.3.2 template<typename InputIterator > InputIterator qpp::internal::IOManipRange< InputIterator >::\_first [private]
- 7.7.3.3 template<typename InputIterator > InputIterator qpp::internal::IOManipRange< InputIterator >::\_last [private]
- 7.7.3.4 template<typename InputIterator > std::string qpp::internal::IOManipRange< InputIterator >::\_separator [private]
- 7.7.3.5 template<typename InputIterator > std::string qpp::internal::IOManipRange< InputIterator >::\_start [private]

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

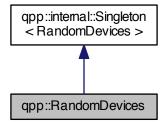
• include/internal/classes/iomanip.h

# 7.8 qpp::RandomDevices Class Reference

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

#include <random\_devices.h>

Inheritance diagram for qpp::RandomDevices:



Collaboration diagram for qpp::RandomDevices:



## **Public Attributes**

std::mt19937 \_rng

Mersenne twister random number generator.

#### **Private Member Functions**

· RandomDevices ()

Initializes and seeds the random number generators.

# **Private Attributes**

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

## **Friends**

class internal::Singleton < RandomDevices >

# **Additional Inherited Members**

## 7.8.1 Detailed Description

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

It consists of a wrapper around an std::mt19937 Mersenne twister random number generator engine and an std⇔ ::random\_device engine. The latter is used to seed the Mersenne twister. The class also seeds the standard std::srand C number generator, as it is used by Eigen.

#### 7.8.2 Constructor & Destructor Documentation

**7.8.2.1 qpp::RandomDevices::RandomDevices()** [inline], [private]

Initializes and seeds the random number generators.

#### 7.8.3 Friends And Related Function Documentation

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

#### 7.8.4 Member Data Documentation

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

```
7.8.4.2 std::mt19937 qpp::RandomDevices::_rng
```

Mersenne twister random number generator.

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

• include/classes/random\_devices.h

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

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

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

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

• static T & get\_instance ()

#### **Protected Member Functions**

- Singleton ()
- virtual  $\sim$ Singleton ()
- Singleton (const Singleton &)=delete
- Singleton & operator= (const Singleton &)=delete

# 7.9.1 Detailed Description

```
template<typename T>class qpp::internal::Singleton< T>
```

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

To implement a singleton, derive your class from qpp::internal::Singleton, make qpp::internal::Singleton a friend of your class, then declare the constructor of your class as private. To get an instance, use the static member function qpp::internal::Singleton::get\_instance(), which returns a reference to your newly created singleton (thread-safe in C++11).

#### Example:

```
class MySingleton: public qpp::internal::Singleton<MySingleton>{
         friend class qpp::internal::Singleton<MySingleton>;
public:
         // Declare all public members here
private:
         MySingleton()
```

```
{
      // Implement the constructor here
   }
};

MySingleton& mySingleton = MySingleton::get_instance(); // Get an instance
```

#### See also

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

- 7.9.2 Constructor & Destructor Documentation
- 7.9.2.1 template < typename T > qpp::internal::Singleton < T >::Singleton( ) [inline], [protected]
- 7.9.2.2 template<typename T> virtual qpp::internal::Singleton< T>::~Singleton( ) [inline], [protected], [virtual]
- 7.9.2.3 template<typename T> qpp::internal::Singleton < T >::Singleton ( const Singleton < T > & ) [protected], [delete]
- 7.9.3 Member Function Documentation
- 7.9.3.1 template < typename T > static T& qpp::internal::Singleton < T >::get\_instance( ) [inline], [static]
- 7.9.3.2 template<typename T> Singleton& qpp::internal::Singleton< T>::operator=( const Singleton< T>& ) [protected], [delete]

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

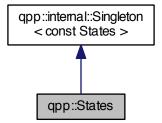
• include/classes/singleton.h

# 7.10 qpp::States Class Reference

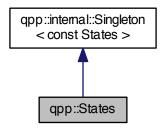
const Singleton class that implements most commonly used states

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

Inheritance diagram for qpp::States:



Collaboration diagram for qpp::States:



#### **Public Attributes**

```
    ket x0 {ket::Zero(2)}
        Pauli Sigma-X 0-eigenstate |+>
    ket x1 {ket::Zero(2)}
        Pauli Sigma-X 1-eigenstate |->
    ket y0 {ket::Zero(2)}
        Pauli Sigma-Y 0-eigenstate |y+>
```

ket y1 {ket::Zero(2)}

Pauli Sigma-Y 1-eigenstate |y->

ket z0 {ket::Zero(2)}

Pauli Sigma-Z 0-eigenstate | 0>

ket z1 {ket::Zero(2)}

Pauli Sigma-Z 1-eigenstate | 1>

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

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

• cmat px1 {cmat::Zero(2, 2)}

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

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

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

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

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

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

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

cmat pz1 {cmat::Zero(2, 2)}

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

ket b00 {ket::Zero(4)}

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

ket b01 {ket::Zero(4)}

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

ket b10 {ket::Zero(4)}

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

ket b11 {ket::Zero(4)}

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

cmat pb00 {cmat::Zero(4, 4)}

Projector onto the Bell-00 state.

• cmat pb01 {cmat::Zero(4, 4)}

Projector onto the Bell-01 state.

cmat pb10 {cmat::Zero(4, 4)}

Projector onto the Bell-10 state.

cmat pb11 {cmat::Zero(4, 4)}

Projector onto the Bell-11 state.

ket GHZ {ket::Zero(8)}

GHZ state.

ket W {ket::Zero(8)}

W state.

cmat pGHZ {cmat::Zero(8, 8)}

Projector onto the GHZ state.

cmat pW {cmat::Zero(8, 8)}

Projector onto the W state.

#### **Private Member Functions**

• States ()

#### **Friends**

class internal::Singleton < const States >

# **Additional Inherited Members**

# 7.10.1 Detailed Description

const Singleton class that implements most commonly used states

## 7.10.2 Constructor & Destructor Documentation

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

Initialize the states

## 7.10.3 Friends And Related Function Documentation

 $\textbf{7.10.3.1} \quad \textbf{friend class internal::Singleton} < \textbf{const States} > \quad \texttt{[friend]}$ 

## 7.10.4 Member Data Documentation

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

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

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

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

```
7.10.4.3 ket qpp::States::b10 {ket::Zero(4)}
Bell-10 state (following the convention in Nielsen and Chuang)
7.10.4.4 ket qpp::States::b11 {ket::Zero(4)}
Bell-11 state (following the convention in Nielsen and Chuang)
7.10.4.5 ket qpp::States::GHZ {ket::Zero(8)}
GHZ state.
7.10.4.6 cmat qpp::States::pb00 {cmat::Zero(4, 4)}
Projector onto the Bell-00 state.
7.10.4.7 cmat qpp::States::pb01 {cmat::Zero(4, 4)}
Projector onto the Bell-01 state.
7.10.4.8 cmat qpp::States::pb10 {cmat::Zero(4, 4)}
Projector onto the Bell-10 state.
7.10.4.9 cmat qpp::States::pb11 {cmat::Zero(4, 4)}
Projector onto the Bell-11 state.
7.10.4.10 cmat qpp::States::pGHZ {cmat::Zero(8, 8)}
Projector onto the GHZ state.
7.10.4.11 cmat qpp::States::pW {cmat::Zero(8, 8)}
Projector onto the W state.
7.10.4.12 cmat qpp::States::px0 {cmat::Zero(2, 2)}
Projector onto the Pauli Sigma-X 0-eigenstate |+><+|.
7.10.4.13 cmat qpp::States::px1 {cmat::Zero(2, 2)}
Projector onto the Pauli Sigma-X 1-eigenstate |-><-|.
7.10.4.14 cmat qpp::States::py0 {cmat::Zero(2, 2)}
Projector onto the Pauli Sigma-Y 0-eigenstate |y+><y+|.
```

```
7.10.4.15 cmat qpp::States::py1 {cmat::Zero(2, 2)}
Projector onto the Pauli Sigma-Y 1-eigenstate |y-><y-|.
7.10.4.16 cmat qpp::States::pz0 {cmat::Zero(2, 2)}
Projector onto the Pauli Sigma-Z 0-eigenstate |0><0|.
7.10.4.17 cmat qpp::States::pz1 {cmat::Zero(2, 2)}
Projector onto the Pauli Sigma-Z 1-eigenstate |1><1|.
7.10.4.18 ket qpp::States::W {ket::Zero(8)}
W state.
7.10.4.19 ket qpp::States::x0 {ket::Zero(2)}
Pauli Sigma-X 0-eigenstate |+>
7.10.4.20 ket qpp::States::x1 {ket::Zero(2)}
Pauli Sigma-X 1-eigenstate |->
7.10.4.21 ket qpp::States::y0 {ket::Zero(2)}
Pauli Sigma-Y 0-eigenstate |y+>
7.10.4.22 ket qpp::States::y1 {ket::Zero(2)}
Pauli Sigma-Y 1-eigenstate |y->
7.10.4.23 ket qpp::States::z0 {ket::Zero(2)}
Pauli Sigma-Z 0-eigenstate |0>
7.10.4.24 ket qpp::States::z1 {ket::Zero(2)}
Pauli Sigma-Z 1-eigenstate |1>
The documentation for this class was generated from the following file:
```

• include/classes/states.h

# 7.11 qpp::Timer Class Reference

#### Measures time.

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

#### **Public Member Functions**

• Timer ()

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

• void tic ()

Resets the chronometer.

• const Timer & toc ()

Stops the chronometer.

• double seconds () const

Time passed in seconds.

## **Protected Attributes**

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

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

#### **Friends**

```
    template<typename charT, typename traits >
    std::basic_ostream< charT,
    traits > & operator<< (std::basic_ostream< charT, traits > &os, const Timer &rhs)
    Overload for std::ostream operators.
```

#### 7.11.1 Detailed Description

Measures time.

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

## 7.11.2 Constructor & Destructor Documentation

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

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

# 7.11.3 Member Function Documentation

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

Time passed in seconds.

Returns

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

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

Resets the chronometer.

Resets the starting/ending point to the current time

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

Stops the chronometer.

Set the current time as the ending point

#### Returns

Current instance

#### 7.11.4 Friends And Related Function Documentation

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

Overload for std::ostream operators.

#### **Parameters**

OS	Output stream
rhs	Timer instance

#### Returns

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

## 7.11.5 Member Data Documentation

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

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

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

• include/classes/timer.h

# **Chapter 8**

# **File Documentation**

# 8.1 include/classes/codes.h File Reference

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



## Classes

class qpp::Codes

const Singleton class that defines quantum error correcting codes

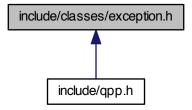
# **Namespaces**

• qpp

94 File Documentation

# 8.2 include/classes/exception.h File Reference

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



# Classes

· class qpp::Exception

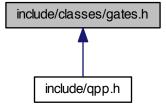
Generates custom exceptions, used when validating function parameters.

# **Namespaces**

qpp

# 8.3 include/classes/gates.h File Reference

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



# **Classes**

class qpp::Gates

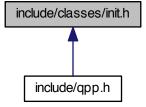
const Singleton class that implements most commonly used gates

# **Namespaces**

• qpp

# 8.4 include/classes/init.h File Reference

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



## Classes

• class qpp::Init

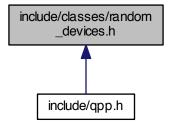
const Singleton class that performs additional initializations/cleanups

# **Namespaces**

• qpp

# 8.5 include/classes/random\_devices.h File Reference

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



96 File Documentation

# Classes

class qpp::RandomDevices

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

# **Namespaces**

qpp

# 8.6 include/classes/singleton.h File Reference

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



# Classes

class qpp::internal::Singleton< T >

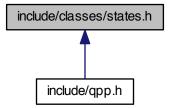
Singleton policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern)

# **Namespaces**

- qpp
- · qpp::internal

# 8.7 include/classes/states.h File Reference

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



## Classes

• class qpp::States

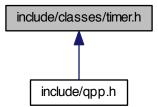
const Singleton class that implements most commonly used states

# **Namespaces**

• qpp

# 8.8 include/classes/timer.h File Reference

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



# **Classes**

• class qpp::Timer

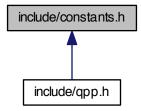
Measures time.

## **Namespaces**

qpp

## 8.9 include/constants.h File Reference

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



## **Namespaces**

qpp

## **Functions**

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

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

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

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

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

D-th root of unity.

#### **Variables**

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

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

• constexpr double qpp::eps = 1e-12

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

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

Maximum number of qubits.

• constexpr double qpp::pi = 3.141592653589793238462643383279502884

π

constexpr double qpp::ee = 2.718281828459045235360287471352662497

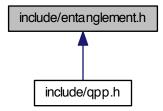
Base of natural logarithm, e.

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

Used to denote infinity.

# 8.10 include/entanglement.h File Reference

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



## **Namespaces**

qpp

## **Functions**

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

Schmidt coefficients of the bi-partite pure state A.

template<typename Derived >

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

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.

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

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

Schmidt probabilities of the bi-partite pure state A.

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

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

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

• template<typename Derived >

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

Negativity of the bi-partite mixed state A.

template<typename Derived >

 $\label{local-poly-to-double-qpp::lognegativity} \mbox{(const Eigen::MatrixBase} < \mbox{Derived} > \&\mbox{A, const std::vector} < \mbox{std::size\_t} > \&\mbox{dims})$ 

Logarithmic negativity of the bi-partite mixed state A.

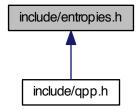
template<typename Derived >

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

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

# 8.11 include/entropies.h File Reference

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



## **Namespaces**

• qpp

#### **Functions**

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

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

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

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

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

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

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

Quantum mutual information between 2 subsystems of a composite system.

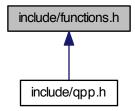
# 8.12 include/experimental/test.h File Reference

## **Namespaces**

- qpp::experimental
- qpp

## 8.13 include/functions.h File Reference

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



## **Namespaces**

qpp

## **Functions**

```
    template<typename Derived >

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

    template<typename Derived >

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

    template<typename Derived >

  DynMat< typename Derived::Scalar > qpp::adjoint (const Eigen::MatrixBase< Derived > &A)
• 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 <a href="mailto:qpp::det">qpp::det</a> (const Eigen::MatrixBase</a> Derived > &A)
      Determinant.

    template<typename Derived >

  Derived::Scalar qpp::logdet (const Eigen::MatrixBase< Derived > &A)
      Logarithm of the determinant.

    template<typename Derived >

  Derived::Scalar qpp::sum (const Eigen::MatrixBase< Derived > &A)
      Element-wise sum of A.
• template<typename Derived >
  Derived::Scalar <a href="mailto:open:prod">open:prod</a> (const Eigen::MatrixBase</a> Derived > &A)
      Element-wise product of A.

    template<typename Derived >
```

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

```
Frobenius norm.
• template<typename Derived >
  DynColVect< cplx > qpp::evals (const Eigen::MatrixBase< Derived > &A)
     Eigenvalues.

    template<typename Derived >

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

    template<typename Derived >

  cmat qpp::hevects (const Eigen::MatrixBase< Derived > &A)
     Hermitian eigenvectors.
• template<typename Derived >
  DynColVect< double > qpp::svals (const Eigen::MatrixBase< Derived > &A)
     Singular values.
template<typename Derived >
  cmat qpp::svdU (const Eigen::MatrixBase< Derived > &A)
     Left singular vectors.

    template<typename Derived >

  cmat qpp::svdV (const Eigen::MatrixBase< Derived > &A)
     Right singular vectors.

    template<typename Derived >

  cmat qpp::funm (const Eigen::MatrixBase< Derived > &A, cplx(*f)(const cplx &))
     Functional calculus f(A)
template<typename Derived >
  cmat qpp::sqrtm (const Eigen::MatrixBase< Derived > &A)
     Matrix square root.
• template<typename Derived >
  cmat qpp::absm (const Eigen::MatrixBase< Derived > &A)
     Matrix absolut value.

    template<typename Derived >

  cmat qpp::expm (const Eigen::MatrixBase< Derived > &A)
     Matrix exponential.

    template<typename Derived >

  cmat qpp::logm (const Eigen::MatrixBase< Derived > &A)
     Matrix logarithm.
• template<typename Derived >
  cmat qpp::sinm (const Eigen::MatrixBase< Derived > &A)
     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 Derived >

  double qpp::schatten (const Eigen::MatrixBase< Derived > &A, std::size_t p)
     Schatten norm.
```

 template<typename OutputScalar , typename Derived > DynMat< OutputScalar > qpp::cwise (const Eigen::MatrixBase< Derived > &A, OutputScalar(\*f)(const typename Derived::Scalar &)) Functor. template<typename T > DynMat< typename T::Scalar > qpp::kron (const T &head) Kronecker product.  $\bullet \ \ template{<} typename\ T\ ,\ typename...\ Args{>}$ DynMat< typename T::Scalar > qpp::kron (const T &head, const Args &...tail) Kronecker product. template<typename Derived > DynMat< typename Derived::Scalar > qpp::kron (const std::vector< Derived > &As) Kronecker product. template<typename Derived > DynMat< typename Derived::Scalar > qpp::kron (const std::initializer list< Derived > &As) Kronecker product. template<typename Derived > DynMat< typename Derived::Scalar > qpp::kronpow (const Eigen::MatrixBase< Derived > &A, std::size\_t n) Kronecker power. template<typename Derived > DynMat< typename Derived::Scalar > qpp::reshape (const Eigen::MatrixBase< Derived > &A, std::size\_t rows, std::size\_t cols) Reshape. template<typename Derived1 , typename Derived2 > DynMat< typename Derived1::Scalar > qpp::comm (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B) Commutator. template<typename Derived1 , typename Derived2 > DynMat< typename Derived1::Scalar > qpp::anticomm (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B) Anti-commutator. template<typename Derived > DynMat< typename Derived::Scalar > qpp::prj (const Eigen::MatrixBase< Derived > &V) Projector. template < typename Derived > DynMat< typename Derived::Scalar > qpp::grams (const std::vector< Derived > &Vs) Gram-Schmidt orthogonalization. template<typename Derived > DynMat< typename Derived::Scalar > qpp::grams (const std::initializer\_list< Derived > &Vs) Gram-Schmidt orthogonalization. • template<typename Derived > DynMat< typename Derived::Scalar > qpp::grams (const Eigen::MatrixBase< Derived > &A) Gram-Schmidt orthogonalization. std::vector < std::size t > qpp::n2multiidx (std::size t n, const std::vector < std::size t > &dims) Non-negative integer index to multi-index. std::size t qpp::multiidx2n (const std::vector< std::size t > &midx, const std::vector< std::size t > &dims) Multi-index to non-negative integer index. ket qpp::mket (const std::vector< std::size\_t > &mask, const std::vector< std::size\_t > &dims) Multi-partite qudit ket. ket qpp::mket (const std::vector< std::size\_t > &mask, std::size\_t d=2) Multi-partite qudit ket. cmat qpp::mprj (const std::vector < std::size\_t > &mask, const std::vector < std::size\_t > &dims)

Projector onto multi-partite qudit ket.

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

Projector onto multi-partite qudit ket.

• template<typename InputIterator >

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

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

• template<typename Derived >

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

Computes the absolut values squared of a column vector.

• template<typename InputIterator >

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

Element-wise sum of a range.

• template<typename InputIterator >

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

Element-wise product of a range.

• template<typename Derived >

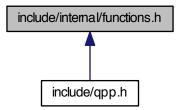
DynColVect< typename

 $\label{eq:const_equal} \mbox{Derived::Scalar} > \mbox{qpp::rho2pure (const Eigen::MatrixBase} < \mbox{Derived} > \&\mbox{A})$ 

Finds the pure state representation of a matrix proportional to a projector onto a pure state.

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

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



## **Namespaces**

- · qpp::internal
- qpp

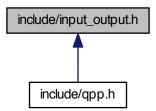
## **Functions**

- void qpp::internal::\_n2multiidx (std::size\_t n, std::size\_t numdims, const std::size\_t \*dims, std::size\_t \*result)
- std::size\_t app::internal::\_multiidx2n (const std::size\_t \*midx, std::size\_t numdims, const std::size\_t \*dims)
- template < typename Derived >
   bool qpp::internal::\_check\_square\_mat (const Eigen::MatrixBase < Derived > &A)
- template<typename Derived >
   bool qpp::internal::\_check\_vector (const Eigen::MatrixBase< Derived > &A)

- template < typename Derived >
   bool qpp::internal::\_check\_row\_vector (const Eigen::MatrixBase < Derived > &A)
- template<typename Derived >
   bool qpp::internal::\_check\_col\_vector (const Eigen::MatrixBase< Derived > &A)
- template<typename T >
   bool qpp::internal::\_check\_nonzero\_size (const T &x)
- bool qpp::internal:: check dims (const std::vector< std::size t > &dims)
- template<typename Derived >
   bool qpp::internal::\_check\_dims\_match\_mat (const std::vector< std::size\_t > &dims, const Eigen::Matrix
   Base< Derived > &A)
- template<typename Derived >
   bool qpp::internal::\_check\_dims\_match\_cvect (const std::vector< std::size\_t > &dims, const Eigen::Matrix
   Base< Derived > &V)
- template<typename Derived >
   bool qpp::internal::\_check\_dims\_match\_rvect (const std::vector< std::size\_t > &dims, const Eigen::Matrix
   Base< Derived > &V)
- bool qpp::internal::\_check\_eq\_dims (const std::vector < std::size\_t > &dims, std::size\_t dim)
- bool qpp::internal::\_check\_subsys\_match\_dims (const std::vector< std::size\_t > &subsys, const std
   ::vector< std::size\_t > &dims)
- bool qpp::internal::\_check\_perm (const std::vector< std::size\_t > &perm)
- template<typename Derived1 , typename Derived2 >
   DynMat< typename Derived1::Scalar > qpp::internal::\_kron2 (const Eigen::MatrixBase< Derived1 > &A,
   const Eigen::MatrixBase< Derived2 > &B)
- template<typename T >
   void qpp::internal::variadic\_vector\_emplace (std::vector< T > &)
- template<typename T, typename First, typename... Args>
   void qpp::internal::variadic\_vector\_emplace (std::vector< T > &v, First &&first, Args &&...args)

# 8.15 include/input\_output.h File Reference

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



## **Namespaces**

qpp

## **Functions**

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

Eigen expression ostream manipulator.

• internal::IOManipEigen qpp::disp (cplx z, double chop=qpp::chop)

Complex number ostream manipulator.

• template<typename InputIterator >

internal::IOManipRange

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

Range ostream manipulator.

• template<typename Container >

internal::IOManipRange

< typename

Container::const\_iterator > qpp::disp (const Container &c, const std::string &separator, const std::string &start="[", const std::string &end="]")

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

template<typename PointerType >

internal::IOManipPointer

< PointerType > qpp::disp (const PointerType \*p, std::size\_t n, const std::string &separator, const std::string &start="[", const std::string &end="]")

C-style pointer ostream manipulator.

• template<typename Derived >

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

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

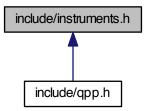
template<typename Derived >

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

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

## 8.16 include/instruments.h File Reference

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



#### **Namespaces**

qpp

#### **Functions**

```
    template<typename Derived >

  std::tuple < std::size_t,
  std::vector< double >
  , std::vector< cmat >> qpp::measure (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat >
  &Ks, const std::vector< std::size t > &subsys, const std::vector< std::size t > &dims)
     Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.
• template<typename Derived >
  std::tuple < std::size_t,
  std::vector< double >
  , std::vector< cmat >> qpp::measure (const Eigen::MatrixBase< Derived > &A, const std::initializer_list<
  cmat > &Ks, const std::vector< std::size_t > &subsys, const std::vector< std::size_t > &dims)
     Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.
template<typename Derived >
  std::tuple < std::size t,
  std::vector< double >
  , std::vector< cmat >> qpp::measure (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat >
  &Ks, const std::vector < std::size t > &subsys, const std::size t = 2)
     Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.
• template<typename Derived >
  std::tuple < std::size_t,
  std::vector< double >
  , std::vector< cmat >> qpp::measure (const Eigen::MatrixBase< Derived > &A, const std::initializer list<
  cmat > &Ks, const std::vector< std::size_t > &subsys, const std::size_t d=2)
     Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.
• template<typename Derived >
  std::tuple < std::size t,
  std::vector< double >
  , std::vector< cmat > > qpp::measure (const Eigen::MatrixBase< Derived > &A, const cmat &U, const
  std::vector< std::size t > &subsys, const std::vector< std::size t > &dims)
     Measures the part subsys of the multi-partite state A in the orthonormal basis specified by the unitary matrix U.

    template<typename Derived >

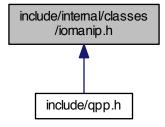
  std::tuple < std::size_t,
  std::vector< double >
  , std::vector < cmat > > qpp::measure (const Eigen::MatrixBase < Derived > &A, const cmat &U, const
  std::vector< std::size_t > &subsys, const std::size_t d=2)
     Measures the part subsys of the multi-partite state A in the orthonormal basis specified by the unitary matrix U.
• template<typename Derived >
  std::tuple < std::size t,
  std::vector< double >
  , std::vector< cmat >> qpp::measure (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat >
  &Ks)
     Measures the state A using the set of Kraus operators Ks.

    template<typename Derived >

  std::tuple < std::size t,
  std::vector< double >
  , std::vector < cmat > > qpp::measure (const Eigen::MatrixBase < Derived > &A, const std::initializer list <
  cmat > &Ks)
     Measures the state A using the set of Kraus operators Ks.
• template<typename Derived >
  std::tuple < std::size t,
  std::vector< double >
  , std::vector< cmat > > qpp::measure (const Eigen::MatrixBase< Derived > &A, const cmat &U)
     Measures the state A in the orthonormal basis specified by the unitary matrix U.
```

# 8.17 include/internal/classes/iomanip.h File Reference

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



#### **Classes**

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

## **Namespaces**

- qpp
- · qpp::internal

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

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

## **Namespaces**

• qpp

## **Functions**

- template < typename Derived >
   Derived qpp::loadMATLABmatrix (const std::string &mat\_file, const std::string &var\_name)
   Loads an Eigen dynamic matrix from a MATLAB .mat file, generic version.
- template<>
   dmat qpp::loadMATLABmatrix (const std::string &mat\_file, const std::string &var\_name)
   Loads an Eigen dynamic matrix from a MATLAB .mat file, specialization for double matrices (qpp::dmat)
- template<>
   cmat qpp::loadMATLABmatrix (const std::string &mat\_file, const std::string &var\_name)
   Loads an Eigen dynamic matrix from a MATLAB .mat file, specialization for complex matrices (qpp::cmat)

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

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

• template<>

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

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

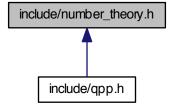
template<>

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

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

# 8.19 include/number\_theory.h File Reference

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



## **Namespaces**

• qpp

#### **Functions**

std::vector < int > qpp::x2contfrac (double x, std::size\_t n, std::size\_t cut=1e5)
 Simple continued fraction expansion.

double qpp::contfrac2x (const std::vector< int > &cf, std::size t n)

Real representation of a simple continued fraction.

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

Real representation of a simple continued fraction.

std::size\_t qpp::gcd (std::size\_t m, std::size\_t n)

Greatest common divisor of two non-negative integers.

std::size t <u>qpp::gcd</u> (const std::vector < std::size t > &ns)

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

std::size\_t qpp::lcm (std::size\_t m, std::size\_t n)

Least common multiple of two positive integers.

- std::size\_t qpp::lcm (const std::vector< std::size\_t > &ns)

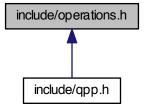
Least common multiple of a list of positive integers.

- std::vector< std::size\_t > qpp::invperm (const std::vector< std::size\_t > &perm)
   Inverse permutation.
- std::vector< std::size\_t > app::compperm (const std::vector< std::size\_t > aperm, const std::vector< std::size\_t

Compose permutations.

# 8.20 include/operations.h File Reference

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



#### **Namespaces**

qpp

#### **Functions**

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

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

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

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

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

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

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

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

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 subsys of the multi-partite density matrix rho.

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

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

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

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

Superoperator matrix representation.

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

Choi matrix representation.

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

Extracts orthogonal Kraus operators from Choi matrix.

• template<typename Derived >

Partial trace.

• template<typename Derived >

DynMat< typename Derived::Scalar > qpp::ptrace2 (const Eigen::MatrixBase< Derived > &A, const std $\leftarrow$  ::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::ptrace (const Eigen::MatrixBase< Derived > &A, const std↔ ::vector< std::size\_t > &subsys, std::size\_t d=2)

Partial trace.

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

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

Partial transpose.

template<typename Derived >

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

Partial transpose.

template<typename Derived >

 $\label{lem:def:def:def:def:DynMat} \begin{tabular}{ll} DynMat< typename Derived::Scalar > qpp::syspermute (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &dims) \end{tabular}$ 

System permutation.

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

System permutation.

# 8.21 include/qpp.h File Reference

```
#include <algorithm>
#include <chrono>
#include <cmath>
#include <complex>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <exception>
#include <fstream>
#include <functional>
#include <initializer_list>
#include <iomanip>
#include <iostream>
#include <iterator>
#include <limits>
#include <numeric>
#include <ostream>
#include <random>
#include <sstream>
#include <stdexcept>
#include <string>
#include <tuple>
#include <type_traits>
#include <utility>
#include <vector>
#include <Eigen/Dense>
#include <Eigen/SVD>
#include "constants.h"
#include "types.h"
#include "classes/exception.h"
#include "internal/classes/iomanip.h"
#include "input_output.h"
#include "classes/singleton.h"
#include "classes/init.h"
#include "internal/functions.h"
#include "functions.h"
#include "classes/codes.h"
#include "classes/gates.h"
#include "classes/states.h"
#include "classes/random_devices.h"
#include "operations.h"
#include "entropies.h"
#include "entanglement.h"
#include "random.h"
#include "classes/timer.h"
#include "instruments.h"
#include "number_theory.h"
```

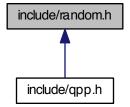
## **Namespaces**

qpp

#### **Variables**

## 8.22 include/random.h File Reference

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



#### **Namespaces**

qpp

#### **Functions**

```
    template<typename Derived >
        Derived qpp::rand (std::size_t rows, std::size_t cols, double a=0, double b=1)
        Generates a random matrix with entries uniformly distributed in the interval [a, b)
    template<>
        dmat qpp::rand (std::size_t rows, std::size_t cols, double a, double b)
        Generates a random real matrix with entries uniformly distributed in the interval [a, b), specialization for double matrices (qpp::dmat)
    template<>
        cmat qpp::rand (std::size_t rows, std::size_t cols, double a, double b)
        Generates a random complex matrix with entries (both real and imaginary) uniformly distributed in the interval [a, b), specialization for complex matrices (qpp::cmat)
```

Generated on Fri Nov 14 2014 14:55:19 for quantum++ by Doxygen

double <a href="mailto:qpp::rand">qpp::rand</a> (double a=0, double b=1)

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

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

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

• template<typename Derived >

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

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

template<>

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

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

template<>

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

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

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

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

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

Generates a random unitary matrix.

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

Generates a random isometry matrix.

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

Generates a set of random Kraus operators.

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

Generates a random Hermitian matrix.

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

Generates a random normalized ket (pure state vector)

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

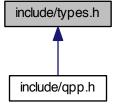
Generates a random density matrix.

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

Generates a random uniformly distributed permutation.

## 8.23 include/types.h File Reference

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



#### **Namespaces**

qpp

## **Typedefs**

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

# 8.24 mainpage.dox File Reference

# Index

absm	qpp::Exception, 69
qpp, 21	DIMS_MISMATCH_RVECTOR
abssq	qpp::Exception, 69
qpp, 22	DIMS_MISMATCH_VECTOR
adjoint	qpp::Exception, 69
qpp, 22	DIMS NOT EQUAL
anticomm	qpp::Exception, 69
qpp, 22	det
apply	qpp, 28
qpp, 23	disp
la va	qpp, 28, 29
bra	dmat
qpp, 20	qpp, 21
OUOTOM EVOERTION	
CUSTOM_EXCEPTION	66
qpp::Exception, 70	qpp, 61
channel	entanglement
qpp, 24, 25	qpp, 29
choi	eps
qpp, 25	qpp, 61
choi2kraus	evals
qpp, 25	qpp, 30
chop	evects
qpp, 60	qpp, <mark>30</mark>
cmat	expm
qpp, 20	qpp, 30
codes	<b>Ч</b> рр, <b>30</b>
	FIVE QUBIT
qpp, 60	qpp::Codes, 66
comm	funm
qpp, 26	qpp, 30
compperm	<b>Ч</b> рр, <b>30</b>
qpp, 26	gcd
concurrence	qpp, 31
qpp, 26	gconcurrence
conjugate	qpp, 31
qpp, 26	grams
contfrac2x	•
qpp, 27	qpp, 31, 32
cosm	gt care 64
qpp, 27	qpp, 61
cplx	hevals
qpp, 21	
cwise	qpp, 32
qpp, 27	hevects
<b>4ρρ</b> , <b>2</b> 7	qpp, 32
DIMS INVALID	infty
qpp::Exception, 69	qpp, 61
DIMS MISMATCH CVECTOR	init
<del>-</del>	
qpp::Exception, 69	qpp, 61
DIMS_MISMATCH_MATRIX	inverse

INDEX 117

qpp, 33	NOT_QUBIT_SUBSYS
invperm	qpp::Exception, 69
qpp, 33	negativity
	qpp, 44
ket	norm
qpp, 21	qpp, 44
kron	
qpp, 33, 34	OUT_OF_RANGE
kronpow	qpp::Exception, 70
qpp, 34	omega
la	qpp, 44
lcm	PERM INVALID
qpp, 34, 35 load	qpp::Exception, 70
qpp, 35	pi
logdet	qpp, 61
qpp, 36	powm
logm	qpp, 45
qpp, 36	prj
lognegativity	qpp, 45
qpp, 37	prod
	qpp, 45
MATRIX_MISMATCH_SUBSYS	ptrace
qpp::Exception, 69	qpp, 46
MATRIX_NOT_CVECTOR	ptrace1
qpp::Exception, 69	qpp, 46
MATRIX_NOT_RVECTOR	ptrace2
qpp::Exception, 69	qpp, 47
MATRIX_NOT_SQUARE	ptranspose
qpp::Exception, 69	qpp, 47
MATRIX_NOT_SQUARE_OR_CVECTOR	amutualinfo
qpp::Exception, 69	qmutualinfo
MATRIX_NOT_SQUARE_OR_RVECTOR	qpp, 47 qpp, 11
qpp::Exception, 69 MATRIX_NOT_SQUARE_OR_VECTOR	absm, 21
qpp::Exception, 69	abssq, 22
MATRIX_NOT_VECTOR	adjoint, 22
qpp::Exception, 69	anticomm, 22
maxn	apply, 23
qpp, 61	bra, <mark>20</mark>
measure	channel, 24, 25
qpp, 37–40	choi, 25
mket	choi2kraus, 25
qpp, 40	chop, 60
mprj	cmat, 20
qpp, 42	codes, 60
multiidx2n	comm, 26
qpp, 42	compperm, 26
0 11" 1	concurrence, 26
n2multiidx	conjugate, 26
qpp, 42	contfrac2x, 27
NINE_QUBIT_SHOR	cosm, 27
qpp::Codes, 66 NO CODEWORD	cplx, 21
qpp::Exception, 69	cwise, 27 det, 28
NOT BIPARTITE	det, 28 disp, 28, 29
qpp::Exception, 69	dmat, 21
NOT_QUBIT_GATE	ee, 61
qpp::Exception, 69	entanglement, 29
11 /	J - ·-···,

118 INDEX

eps, 61	spectralpowm, 57
evals, 30	sqrtm, 57
evects, 30	st, 61
expm, 30	sum, 57, 58
funm, 30	super, 58
gcd, 31	svals, 58
gconcurrence, 31	syspermute, 59
grams, 31, 32	trace, 59
gt, 61	transpose, 60
hevals, 32	tsallis, 60
hevects, 32	x2contfrac, 60
infty, 61	qpp::Codes
init, 61	FIVE_QUBIT, 66
inverse, 33	NINE_QUBIT_SHOR, 66
invperm, 33	SEVEN_QUBIT_STEANE, 66
ket, 21	qpp::Exception
kron, 33, 34	CUSTOM_EXCEPTION, 70
kronpow, 34	DIMS_INVALID, 69
lcm, 34, 35	DIMS_MISMATCH_CVECTOR, 69
load, 35	DIMS_MISMATCH_MATRIX, 69
logdet, 36	DIMS_MISMATCH_RVECTOR, 69
logm, 36	DIMS_MISMATCH_VECTOR, 69
lognegativity, 37	DIMS_NOT_EQUAL, 69
maxn, 61	MATRIX_MISMATCH_SUBSYS, 69
measure, 37–40	MATRIX_NOT_CVECTOR, 69
mket, 40	MATRIX_NOT_RVECTOR, 69
mprj, 42	MATRIX_NOT_SQUARE, 69
multiidx2n, 42	MATRIX_NOT_SQUARE_OR_CVECTOR, 69
n2multiidx, 42	MATRIX_NOT_SQUARE_OR_RVECTOR, 69
negativity, 44	MATRIX_NOT_SQUARE_OR_VECTOR, 69
norm, 44	MATRIX_NOT_VECTOR, 69
omega, 44	NO_CODEWORD, 69
pi, 61	NOT_BIPARTITE, 69
powm, 45	NOT_QUBIT_GATE, 69
prj, 45	NOT_QUBIT_SUBSYS, 69
prod, 45	OUT_OF_RANGE, 70
ptrace, 46	PERM_INVALID, 70
ptrace1, 46	SUBSYS_MISMATCH_DIMS, 69
ptrace2, 47	TYPE_MISMATCH, 70
ptranspose, 47	UNDEFINED_TYPE, 70
qmutualinfo, 47	UNKNOWN_EXCEPTION, 69
rand, 49, 50	ZERO_SIZE, 69
randint, 50	rand
randket, 50	qpp, 49, 50
randkraus, 51	randint
randn, 51, 52	qpp, 50
randperm, 52	randket
randrho, 52	qpp, 50
rdevs, 61	randkraus
renyi, 53	qpp, 51
reshape, 53	randn
rho2pure, 54	qpp, 51, 52
save, 54	randperm
schatten, 55	qpp, 52
schmidtcoeff, 55	randrho
schmidtprob, 55	qpp, 52
shannon, 56	rdevs
sinm, 57	
Jiiiii, Ji	qpp, 61

```
renyi
    qpp, 53
reshape
    qpp, 53
rho2pure
    qpp, 54
SEVEN_QUBIT_STEANE
    qpp::Codes, 66
SUBSYS_MISMATCH_DIMS
    qpp::Exception, 69
save
    qpp, <mark>54</mark>
schatten
    qpp, 55
schmidtcoeff
    qpp, 55
schmidtprob
    qpp, 55
shannon
    qpp, <mark>56</mark>
sinm
    qpp, 57
spectralpowm
    qpp, <mark>57</mark>
sqrtm
    qpp, 57
st
    qpp, 61
sum
    qpp, 57, 58
super
     qpp, 58
svals
    qpp, 58
syspermute
    qpp, 59
TYPE_MISMATCH
    qpp::Exception, 70
trace
    qpp, 59
transpose
    qpp, 60
tsallis
    qpp, 60
UNDEFINED_TYPE
     qpp::Exception, 70
UNKNOWN_EXCEPTION
    qpp::Exception, 69
x2contfrac
    qpp, 60
ZERO_SIZE
```

qpp::Exception, 69