Quantum++ v0.1

Generated by Doxygen 1.8.7

Tue Dec 9 2014 18:17:46

## **Contents**

1	Qua	ntum++													1
2	Nam	nespace	Index												5
	2.1	Names	space List				 	5							
3	Hier	archical	l Index												7
	3.1	Class I	Hierarchy				 	7							
4	Clas	s Index													9
	4.1	Class I	∟ist				 	9							
5	File	Index													11
	5.1	File Lis	st				 	11							
6	Nam	nespace	Docume	ntation	I										13
	6.1	qpp Na	amespace	Refere	nce .		 	13							
		6.1.1	Detailed	Descri	ption		 	23							
		6.1.2	Typedef	Docum	entatio	n .	 	23							
			6.1.2.1	bra .			 	23							
			6.1.2.2	cmat			 	23							
			6.1.2.3	cplx .			 	23							
			6.1.2.4	dmat			 	23							
			6.1.2.5	dyn_c	col_ved	ct .	 	23							
			6.1.2.6	dyn_ı	mat .		 	23							
			6.1.2.7	dyn_ı	row_ve	ct	 	24							
			6.1.2.8	idx .			 	24							
			6.1.2.9	ket .			 	24							
		6.1.3	Function	Docun	nentati	on	 	24							
			6.1.3.1	absm			 	24							
			6.1.3.2	absso	<b>q</b>		 	24							
			6.1.3.3	absso	<b>q</b>		 	24							
			6.1.3.4	adjoir	nt		 	25							
			6105	ontio	omm										25

iv CONTENTS

6.1.3.6	apply	25
6.1.3.7	apply	26
6.1.3.8	apply	26
6.1.3.9	apply	26
6.1.3.10	apply	27
6.1.3.11	applyCTRL	27
6.1.3.12	applyCTRL	27
6.1.3.13	choi	28
6.1.3.14	choi2kraus	28
6.1.3.15	comm	28
6.1.3.16	compperm	29
6.1.3.17	concurrence	29
6.1.3.18	conjugate	29
6.1.3.19	contfrac2x	29
6.1.3.20	contfrac2x	30
6.1.3.21	cosm	30
6.1.3.22	cwise	30
6.1.3.23	det	30
6.1.3.24	disp	31
6.1.3.25	disp	31
6.1.3.26	disp	31
6.1.3.27	disp	31
6.1.3.28	disp	32
6.1.3.29	eig	32
6.1.3.30	entanglement	32
6.1.3.31	entropy	33
6.1.3.32	entropy	33
6.1.3.33	evals	33
6.1.3.34	evects	33
6.1.3.35	expm	34
6.1.3.36	funm	35
6.1.3.37	gcd	35
6.1.3.38	gcd	35
6.1.3.39	gconcurrence	35
6.1.3.40	grams	36
6.1.3.41	grams	36
6.1.3.42	grams	36
6.1.3.43	heig	36
6.1.3.44	hevals	37
6.1.3.45	hevects	37

CONTENTS

6.1.3.46	inverse	37
6.1.3.47	invperm	37
6.1.3.48	kron	38
6.1.3.49	kron	38
6.1.3.50	kron	38
6.1.3.51	kron	38
6.1.3.52	kronpow	39
6.1.3.53	lcm	39
6.1.3.54	lcm	39
6.1.3.55	load	39
6.1.3.56	loadMATLABmatrix	40
6.1.3.57	loadMATLABmatrix	40
6.1.3.58	loadMATLABmatrix	40
6.1.3.59	logdet	41
6.1.3.60	logm	41
6.1.3.61	lognegativity	41
6.1.3.62	measure	41
6.1.3.63	measure	42
6.1.3.64	measure	42
6.1.3.65	measure	43
6.1.3.66	measure	43
6.1.3.67	measure	43
6.1.3.68	measure	44
6.1.3.69	measure	44
6.1.3.70	measure	44
6.1.3.71	mket	45
6.1.3.72	mket	45
6.1.3.73	mprj	45
6.1.3.74	mprj	45
6.1.3.75	multiidx2n	46
6.1.3.76	n2multiidx	46
6.1.3.77	negativity	46
6.1.3.78	norm	46
6.1.3.79	omega	47
6.1.3.80	operator""""_i	47
6.1.3.81	operator"""_i	47
6.1.3.82	powm	47
6.1.3.83	prj	47
6.1.3.84	prod	48
6.1.3.85	prod	48

vi CONTENTS

6.1.3.86 ptrace	48
6.1.3.87 ptrace	48
6.1.3.88 ptrace1	49
6.1.3.89 ptrace2	49
6.1.3.90 ptranspose	49
6.1.3.91 ptranspose	50
6.1.3.92 qmutualinfo	50
6.1.3.93 qmutualinfo	50
6.1.3.94 rand	51
6.1.3.95 rand	51
6.1.3.96 rand	51
6.1.3.97 rand	52
6.1.3.98 randH	52
6.1.3.99 randidx	52
6.1.3.100 randket	52
6.1.3.101 randkraus	53
6.1.3.102 randn	53
6.1.3.103 randn	53
6.1.3.104 randn	54
6.1.3.105 randn	54
6.1.3.106 randperm	54
6.1.3.107 randrho	54
6.1.3.108 randU	55
6.1.3.109 randV	55
6.1.3.110 renyi	55
6.1.3.111 renyi	55
6.1.3.112 reshape	56
6.1.3.113 rho2pure	56
6.1.3.114 save	56
6.1.3.115 saveMATLABmatrix	57
6.1.3.116 saveMATLABmatrix	57
6.1.3.117 saveMATLABmatrix	57
6.1.3.118 schatten	57
6.1.3.119 schmidtA	57
6.1.3.120 schmidtB	58
6.1.3.121 schmidtcoeff	58
6.1.3.122 schmidtprob	58
6.1.3.123 sinm	59
	59
6.1.3.125 sqrtm	59

CONTENTS vii

		6.1.3.126	S sum	. 59
		6.1.3.127	7 sum	. 60
		6.1.3.128	3 super	. 60
		6.1.3.129	9 svals	. 60
		6.1.3.130	) svd	. 60
		6.1.3.131	I svdU	. 61
		6.1.3.132	2 svdV	. 61
		6.1.3.133	3 syspermute	. 61
		6.1.3.134	4 syspermute	. 61
		6.1.3.135	5 trace	. 62
		6.1.3.136	5 transpose	. 62
		6.1.3.137	7 tsallis	. 62
		6.1.3.138	3 tsallis	. 62
		6.1.3.139	9 x2contfrac	. 63
	6.1.4	Variable I	Documentation	. 63
		6.1.4.1	chop	. 63
		6.1.4.2	codes	. 63
		6.1.4.3	ee	. 63
		6.1.4.4	eps	. 63
		6.1.4.5	gt	. 63
		6.1.4.6	infty	. 64
		6.1.4.7	init	. 64
		6.1.4.8	maxn	. 64
		6.1.4.9	pi	. 64
		6.1.4.10	rdevs	. 64
		6.1.4.11	st	. 64
6.2	qpp::ex	perimenta	al Namespace Reference	. 64
	6.2.1	Detailed	Description	. 64
6.3	qpp::int	ternal Nan	nespace Reference	. 64
	6.3.1	Detailed	Description	. 65
	6.3.2	Function	Documentation	. 65
		6.3.2.1	_check_col_vector	. 65
		6.3.2.2	_check_dims	. 65
		6.3.2.3	_check_dims_match_cvect	. 65
		6.3.2.4	_check_dims_match_mat	. 65
		6.3.2.5	_check_dims_match_rvect	. 65
		6.3.2.6	_check_eq_dims	. 65
		6.3.2.7	_check_nonzero_size	. 66
		6.3.2.8	_check_perm	. 66
		6.3.2.9	_check_row_vector	. 66

viii CONTENTS

			6.3.2.10 _check_square_mat	66
			6.3.2.11 _check_subsys_match_dims	66
			6.3.2.12 _check_vector	66
			6.3.2.13 _kron2	66
			6.3.2.14 _multiidx2n	66
			6.3.2.15 _n2multiidx	36
			6.3.2.16 variadic_vector_emplace	66
			6.3.2.17 variadic_vector_emplace	66
7	Clas	s Docu	mentation 6	67
	7.1	qpp::C	odes Class Reference	37
		7.1.1	Detailed Description	38
		7.1.2	Member Enumeration Documentation	38
			7.1.2.1 Type	38
		7.1.3	Constructor & Destructor Documentation	38
			7.1.3.1 Codes	38
		7.1.4	Member Function Documentation	38
			7.1.4.1 codeword	38
		7.1.5	Friends And Related Function Documentation	39
			7.1.5.1 internal::Singleton < const Codes >	39
	7.2	qpp::E	xception Class Reference	69
		7.2.1	Detailed Description	71
		7.2.2	Member Enumeration Documentation	71
			7.2.2.1 Type	71
		7.2.3	Constructor & Destructor Documentation	72
			7.2.3.1 Exception	72
			7.2.3.2 Exception	72
		7.2.4	Member Function Documentation	72
			7.2.4.1 _construct_exception_msg	72
			7.2.4.2 what	72
		7.2.5	Member Data Documentation	73
			7.2.5.1 _custom	73
			7.2.5.2 _msg	73
			7.2.5.3 _type	73
			7.2.5.4 _where	73
	7.3	qpp::G	ates Class Reference	73
		7.3.1	Detailed Description	75
		7.3.2	Constructor & Destructor Documentation	75
			7.3.2.1 Gates	75
		7.3.3	Member Function Documentation	75

CONTENTS

		7.3.3.1	CTRL	75
		7.3.3.2	expandout	75
		7.3.3.3	Fd	76
		7.3.3.4	ld	76
		7.3.3.5	Rn	76
		7.3.3.6	Xd	77
		7.3.3.7	Zd	77
	7.3.4	Friends A	nd Related Function Documentation	77
		7.3.4.1	$internal:: Singleton < const \ Gates > \ . \ . \ . \ . \ . \ . \ . \ . \ . \$	77
	7.3.5	Member D	Data Documentation	77
		7.3.5.1	CNOT	77
		7.3.5.2	CNOTba	77
		7.3.5.3	CZ	77
		7.3.5.4	FRED	77
		7.3.5.5	H	78
		7.3.5.6	ld2	78
		7.3.5.7	S	78
		7.3.5.8	SWAP	78
		7.3.5.9	T	78
			TOF	78
			X	78
			Y	78
			Z	78
7.4	qpp::In	it Class Re	ference	78
	7.4.1		Description	79
	7.4.2	Constructo	or & Destructor Documentation	80
			Init	80
			~Init	80
	7.4.3	Friends A	nd Related Function Documentation	80
		7.4.3.1	internal::Singleton < const Init >	80
7.5	qpp::in	ternal::IOM	anipEigen Class Reference	80
	7.5.1	Constructo	or & Destructor Documentation	80
		7.5.1.1	IOManipEigen	80
		7.5.1.2	IOManipEigen	80
	7.5.2	Friends A	nd Related Function Documentation	80
		7.5.2.1	operator <<	81
	7.5.3	Member D	Data Documentation	81
			_A	81
			_chop	81
7.6	qpp::in	ternal::IOM	anipPointer< PointerType > Class Template Reference	81

X CONTENTS

	7.6.1	Constructor & Destructor Documentation	32
		7.6.1.1 IOManipPointer	32
		7.6.1.2 IOManipPointer	32
	7.6.2	Member Function Documentation	32
		7.6.2.1 operator=	32
	7.6.3	Friends And Related Function Documentation	32
		7.6.3.1 operator<<	32
	7.6.4	Member Data Documentation	32
		7.6.4.1 _end	32
		7.6.4.2 _n	32
		7.6.4.3 _p	32
		7.6.4.4 _separator	32
		7.6.4.5 _start	32
7.7	qpp::in	ternal::IOManipRange< InputIterator > Class Template Reference	32
	7.7.1	Constructor & Destructor Documentation	33
		7.7.1.1 IOManipRange	33
	7.7.2	Friends And Related Function Documentation	33
		7.7.2.1 operator<<	34
	7.7.3	Member Data Documentation	34
		7.7.3.1 _end	34
		7.7.3.2 _first	34
		7.7.3.3 _last	34
		7.7.3.4 _separator	34
		7.7.3.5 _start	34
7.8	qpp::R	andomDevices Class Reference	34
	7.8.1	Detailed Description	35
	7.8.2	Constructor & Destructor Documentation	35
		7.8.2.1 RandomDevices	35
	7.8.3	Friends And Related Function Documentation	36
		7.8.3.1 internal::Singleton < RandomDevices >	36
	7.8.4	Member Data Documentation	36
		7.8.4.1 _rd	36
		7.8.4.2 _rng	36
7.9	qpp::in	ternal::Singleton< T > Class Template Reference	36
	7.9.1	Detailed Description	36
	7.9.2	Constructor & Destructor Documentation	37
		7.9.2.1 Singleton	37
		7.9.2.2 ~Singleton	37
		7.9.2.3 Singleton	37
	7.9.3	Member Function Documentation	37

CONTENTS xi

	7.9.3.1	get_instance	 	87
	7.9.3.2	operator=	 	87
7.10 qpp::S	tates Class	ss Reference	 	87
7.10.1	Detailed	Description	 	89
7.10.2	Construc	ctor & Destructor Documentation	 	89
	7.10.2.1	States	 	89
7.10.3	Friends A	And Related Function Documentation	 	89
	7.10.3.1	internal::Singleton < const States >	 	89
7.10.4	Member	Data Documentation	 	89
	7.10.4.1	b00	 	89
	7.10.4.2	b01	 	89
	7.10.4.3	b10	 	90
	7.10.4.4	b11	 	90
	7.10.4.5	GHZ	 	90
	7.10.4.6	pb00	 	90
	7.10.4.7	pb01	 	90
	7.10.4.8	pb10	 	90
	7.10.4.9	pb11	 	90
	7.10.4.10	0 pGHZ	 	90
	7.10.4.11	1 pW	 	90
	7.10.4.12	2 px0	 	90
	7.10.4.13	3 px1	 	90
	7.10.4.14	4 py0	 	90
	7.10.4.15	5 py1	 	91
	7.10.4.16	6 pz0	 	91
	7.10.4.17	7 pz1	 	91
	7.10.4.18	8 W	 	91
	7.10.4.19	9 x0	 	91
	7.10.4.20	0 x1	 	91
	7.10.4.21	1 y0	 	91
	7.10.4.22	2 y1	 	91
	7.10.4.23	3 z0	 	91
	7.10.4.24	4 z1	 	91
7.11 qpp::Ti	imer Class	s Reference	 	91
7.11.1	Detailed	Description	 	92
7.11.2	Construc	ctor & Destructor Documentation	 	92
	7.11.2.1	Timer	 	92
7.11.3	Member	Function Documentation	 	92
	7.11.3.1	seconds	 	92
	7.11.3.2	tic	 	92

xii CONTENTS

			93
		7.11.4 Friends And Related Function Documentation	93
		7.11.4.1 operator <<	93
		7.11.5 Member Data Documentation	93
		7.11.5.1 _end	93
		7.11.5.2 _start	93
8	File I	Documentation 9	95
	8.1	classes/codes.h File Reference	95
		8.1.1 Detailed Description	95
	8.2		95
		8.2.1 Detailed Description	96
	8.3	classes/gates.h File Reference	96
		8.3.1 Detailed Description	97
	8.4	classes/init.h File Reference	97
		8.4.1 Detailed Description	98
	8.5	classes/random_devices.h File Reference	98
		8.5.1 Detailed Description	98
	8.6	classes/states.h File Reference	99
		8.6.1 Detailed Description	99
	8.7	classes/timer.h File Reference	99
		8.7.1 Detailed Description	00
	8.8	constants.h File Reference	00
		8.8.1 Detailed Description	01
	8.9	entanglement.h File Reference	01
		8.9.1 Detailed Description	02
	8.10	entropies.h File Reference	02
		8.10.1 Detailed Description	03
	8.11	experimental/test.h File Reference	03
		8.11.1 Detailed Description	03
	8.12	functions.h File Reference	04
		8.12.1 Detailed Description	07
	8.13	input_output.h File Reference	07
		8.13.1 Detailed Description	80
	8.14	instruments.h File Reference	09
		8.14.1 Detailed Description	10
	8.15	internal/classes/iomanip.h File Reference	
		8.15.1 Detailed Description	
	8.16	internal/classes/singleton.h File Reference	
		8.16.1 Detailed Description	11

CONTENTS xiii

8.17	internal/util.h File Reference	112
	8.17.1 Detailed Description	113
8.18	MATLAB/matlab.h File Reference	113
	8.18.1 Detailed Description	114
8.19	number_theory.h File Reference	114
	8.19.1 Detailed Description	115
8.20	operations.h File Reference	115
	8.20.1 Detailed Description	116
8.21	qpp.h File Reference	117
	8.21.1 Detailed Description	118
8.22	random.h File Reference	118
	8.22.1 Detailed Description	119
8.23	types.h File Reference	119
	8.23.1 Detailed Description	120
Index		121

### **Chapter 1**

### Quantum++

Quantum++ is a C++11 general purpose quantum computing library, composed solely of template header files. It uses the Eigen 3 linear algebra library and, if available, the OpenMP multi-processing library. For additional Eigen 3 documentation see http://eigen.tuxfamily.org/dox/. For a simple Eigen 3 quick ASCII reference see http://eigen.tuxfamily.org/dox/AsciiQuickReference.txt.

Copyright (c) 2013 - 2014 Vlad Gheorghiu, vgheorgh AT gmail DOT com.

If you are interesting in contributing, please let me know. There is still work left to be done, and I can provide you with more details about what I have in mind. To contribute, you need to have a decent knowledge of C++ (preferably C++11), including templates and the standard library, a basic knowledge of quantum computing and linear algebra, and some working experience with Eigen 3.

The ultimate goal of this project is to build a universal quantum simulator, applicable to a vast majority of problems in quantum information/computation. The simulator should be fast but nevertheless user-friendly for anyone with a basic knowledge of C/C++.

Quantum++ is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Quantum++ is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with Quantum++. If not, see http-://www.gnu.org/licenses/.

#### **Building instructions**

#### Configuration:

- Compiler:  $g++ \ge 4.8$  (for good C++11 support)
- Eigen 3 library located in \$HOME/eigen
- Quantum++ library located in \$HOME/qpp
- MATLAB compiler include header files: /Applications/MATLAB\_R2014b.app/extern/include
- MATLAB compiler shared library files: /Applications/MATLAB\_R2014b.app/bin/maci64

### Building without a build system

- Example file: \$HOME/qpp/examples/example.cpp
- Output executable: \$HOME/qpp/examples/example

2 Quantum++

Must run the commands below from inside the directory \$HOME/qpp/examples

### Release version (without MATLAB support):

```
g++ -pedantic -std=c++11 -Wall -Wextra -Weffc++ -fopenmp \
    -03 -DNDEBUG -DEIGEN_NO_DEBUG \
    -isystem $HOME/eigen -I $HOME/qpp/include \
    example.cpp -o example
```

#### Debug version (without MATLAB support):

```
g++ -pedantic -std=c++11 -Wall -Wextra -Weffc++ -fopenmp \
    -g3 -DDEBUG \
    -isystem $HOME/eigen -I $HOME/qpp/include \
    example.cpp -o example
```

#### Release version (with MATLAB support):

```
g++ -pedantic -std=c++11 -Wall -Wextra -Weffc++ -fopenmp \
    -03 -DNDEBUG -DEIGEN_NO_DEBUG \
    -isystem $HOME/eigen -I $HOME/qpp/include \
    -I/Applications/MATLAB_R2014b.app/extern/include \
    -L/Applications/MATLAB_R2014b.app/bin/maci64 \
    -lmx -lmat example.cpp -o example
```

### Debug version (with MATLAB support):

```
g++ -pedantic -std=c++11 -Wall -Wextra -Weffc++ -fopenmp \
    -g3 -DDEBUG \
    -isystem $HOME/eigen -I $HOME/qpp/include \
    -I /Applications/MATLAB_R2014b.app/extern/include \
    -L /Applications/MATLAB_R2014b.app/bin/maci64 \
    -lmx -lmat example.cpp -o example
```

### Building using cmake

The current version of the repository has a CMakeLists.txt configuration file for building examples using cmake (cmake needs to be installed). To build an example using cmake, I recommend an out-of-source build, i.e., from the root of the project (where ./include is located), type

```
mkdir ./build
cd ./build
cmake ..
make
```

The above commands build the relase version (default) executable qpp, from the source file ./examples/example.cpp, without MATLAB support (default), inside the directory ./build. To build a different configuration, e.g. debug version with MATLAB support, type from the root of the project

```
cd ./build  \begin{tabular}{ll} $\sf cd ./build \\ $\sf rm -rf * \\ $\sf cmake -DCMAKE\_BUILD\_TYPE=Debug -DWITH\_MATLAB=ON .. \\ $\sf make \\ \end{tabular}
```

### Or, to disable OpenMP support (enabled by default), type

```
cd ./build
rm -rf *
cmake -DWITH_OPENMP=OFF ..
make
```

To change the name of the example file, the location of the Eigen 3 library or the location of MATLAB installation, edit the CMakeLists.txt file. See also CMakeLists.txt for additional options. Do not forget to remove everything from the ./build directory before a fresh build!

#### Additional remarks

- The C++ compiler must be C++11 compliant.
- If your compiler does not support OpenMP (as it is the case e.g with clang++), disable OpenMP in your build, as otherwise the linker may not find the gomp library.
- If you run the program on OS X with MATLAB support, make sure that the environment variable DYLD\_L← IBRARY\_PATH is set to point to the MATLAB compiler library location, see the run\_OSX\_MATLAB script. Otherwise, you will get a runtime error like dyld: Library not loaded: @rpath/libmat.← dylib.

 $\star$  I recommend running via a script, as otherwise setting the

- 'DYLD\_LIBRARY\_PATH' globally may interfere with Macports' 'cmake' installation (in case you use 'cmake' from 'macports'). If you use a script, then the environment variable is local to the script and does not interfere with the rest of the system.

  \* Example of running script, run from inside the directory where the executable 'qpp' is located:

  #!/bin/sh # Run Quantum++ under OS X with MATLAB support
- If you build a debug version with g++ under OS X and use gdb to step inside template functions you may want to add -fno-weak compiler flag. See http://stackoverflow.← com/questions/23330641/gnu-gdb-can-not-step-into-template-functions-os-x-mavericks for more details about this problem.

export DYLD\_LIBRARY\_PATH=\$DYLD\_LIBRARY\_PATH:"/Applications/MATLAB\_R2014b.app/bin/maci64"

Quantum++

# Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<del>qpp</del>	
Quantum++ main namespace	13
qpp::experimental	
Experimental/test functions/classes, do not use or modify	64
qpp::internal	
Internal utility functions, do not use/modify	64

6 Namespace Index

# Chapter 3

## **Hierarchical Index**

### 3.1 Class Hierarchy

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

staexception	
qpp::Exception	69
qpp::internal::IOManipEigen	80
qpp::internal::IOManipPointer< PointerType >	81
qpp::internal::IOManipRange < InputIterator >	82
$qpp::internal::Singleton < T > \dots \dots$	86
qpp::internal::Singleton< const Codes >	86
qpp::Codes	67
qpp::internal::Singleton< const Gates >	86
qpp::Gates	73
$\label{lem:qpp::internal::Singleton} $$\operatorname{qpp}::\operatorname{internal}::\operatorname{Singleton}<\operatorname{const}\operatorname{Init}>\dots$	86
qpp::Init	78
$\label{lem:qpp::internal::Singleton} $$\operatorname{qpp}::\operatorname{internal}::\operatorname{Singleton}<\operatorname{const} \operatorname{States}>\ldots\ldots\ldots\ldots\ldots\ldots\ldots$	86
qpp::States	87
$\label{lem:qpp::internal::Singleton} \mbox{\tt RandomDevices} > \dots $	86
qpp::RandomDevices	84
qpp::Timer	91

8 **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	67
qpp::Exception	
Generates custom exceptions, used when validating function parameters	69
qpp::Gates	
Const Singleton class that implements most commonly used gates	73
qpp::Init	
Const Singleton class that performs additional initializations/cleanups	78
qpp::internal::IOManipEigen	80
qpp::internal::IOManipPointer< PointerType >	81
$qpp::internal::IOManipRange < Input Iterator > \dots \dots \dots \dots \dots \dots \\$	82
qpp::RandomDevices	
Singeleton class that manages the source of randomness in the library	84
qpp::internal::Singleton< T >	
Singleton policy class, used internally to implement the singleton pattern via CRTP (Curiously	
recurring template pattern)	86
qpp::States	
Const Singleton class that implements most commonly used states	87
qpp::Timer	
Measures time	91

10 Class Index

## **Chapter 5**

## File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

constants.h
Constants
entanglement.h
Entanglement functions
entropies.h
Entropy functions
functions.h
Generic quantum computing functions
input_output.h
Input/output functions
instruments.h
Measurement functions
number_theory.h
Number theory functions
operations.h
Quantum operation functions
qpp.h
Quantum++ main header file, includes all other necessary headers
random.h
Randomness-related functions
types.h
Type aliases
classes/codes.h
Quantum error correcting codes
classes/exception.h
Exceptions
classes/gates.h
Quantum gates
classes/init.h
Initialization
classes/random_devices.h
Random devices
classes/states.h
Quantum states
classes/timer.h Timing
experimental/test.h
Experimental/test functions/classes

12 File Index

internal/util.h												
Internal utility functions		 		 			 					112
internal/classes/iomanip.h												
Input/output manipulators		 		 			 					110
internal/classes/singleton.h												
Singleton pattern via CRTP		 		 			 					111
MATLAB/matlab.h												
Input/output interfacing with MATLA	ιВ	 	 	 			 					113

### **Chapter 6**

## **Namespace Documentation**

### 6.1 qpp Namespace Reference

Quantum++ main namespace.

### **Namespaces**

· experimental

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

internal

Internal utility functions, do not use/modify.

### 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 dyn_mat = Eigen::Matrix< Scalar, Eigen::Dynamic, Eigen::Dynamic >
        Dynamic Eigen matrix over the field specified by Scalar.
```

```
• template<typename Scalar >
      using dyn_col_vect = Eigen::Matrix< Scalar, Eigen::Dynamic, 1 >
          Dynamic Eigen column vector over the field specified by Scalar.

    template<typename Scalar >

      using dyn_row_vect = Eigen::Matrix < Scalar, 1, Eigen::Dynamic >
          Dynamic Eigen row vector over the field specified by Scalar.
    using ket = dyn_col_vect< cplx >
          Complex (double precision) dynamic Eigen column vector.
    using bra = dyn_row_vect< cplx >
          Complex (double precision) dynamic Eigen row vector.
    using cmat = dyn mat < cplx >
          Complex (double precision) dynamic Eigen matrix.
    using dmat = dyn mat< double >
          Real (double precision) dynamic Eigen matrix.
    using idx = std::size t
          Non-negative integer index.
Functions

    constexpr cplx operator" i (unsigned long long int x)

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

    constexpr cplx operator""_i (long double x)

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

    cplx omega (idx D)

          D-th root of unity.

    template<typename Derived >

      dyn col vect< double > schmidtcoeff (const Eigen::MatrixBase< Derived > &A, const std::vector< idx >
      &dims)
          Schmidt coefficients of the bi-partite pure state A.

    template<typename Derived >

      cmat schmidtA (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)
          Schmidt basis on Alice's side.

    template<typename Derived >

      cmat schmidtB (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)
          Schmidt basis on Bob's side.

    template<typename Derived >

      std::vector< double > schmidtprob (const Eigen::MatrixBase< Derived > &A, const std::vector< idx >
      &dims)
          Schmidt probabilities of the bi-partite pure state A.

    template<typename Derived >

      double entanglement (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &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< idx > &dims)
          Negativity of the bi-partite mixed state A.

    template<typename Derived >

      double lognegativity (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &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 entropy (const Eigen::MatrixBase< Derived > &A)
      von-Neumann entropy of the density matrix A

    double entropy (const std::vector< double > &prob)

     Shannon entropy of the probability distribution prob.
template<typename Derived >
  double renyi (const Eigen::MatrixBase< Derived > &A, double alpha)
     Renyi- \alpha entropy of the density matrix A, for \alpha \geq 0.

    double renyi (const std::vector< double > &prob, double alpha)

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

    double tsallis (const std::vector< double > &prob, double q)

      Tsallis- q entropy of the probability distribution prob, for q \ge 0.
• template<typename Derived >
  double qmutualinfo (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsysA, const
  std::vector< idx > &subsysB, const std::vector< idx > &dims)
     Quantum mutual information between 2 subsystems of a composite system.

    template<typename Derived >

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

    template<typename Derived >

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

    template<typename Derived >

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

    template<typename Derived >

  dyn mat< typename Derived::Scalar > adjoint (const Eigen::MatrixBase< Derived > &A)
     Adjoint.
• template<typename Derived >
  dyn mat< typename Derived::Scalar > inverse (const Eigen::MatrixBase< Derived > &A)
     Inverse.

    template<typename Derived >

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

    template<typename Derived >

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

    template<typename Derived >

  Derived::Scalar sum (const Eigen::MatrixBase< Derived > &A)
     Element-wise sum of A.

    template<typename Derived >

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

Element-wise product of A.

```
• template<typename Derived >
  double norm (const Eigen::MatrixBase< Derived > &A)
     Frobenius norm.

    template<typename Derived >

  std::pair< dyn col vect< cplx >
  , cmat > eig (const Eigen::MatrixBase< Derived > &A)
     Full eigen decomposition.
• template<typename Derived >
  dyn_col_vect< cplx > evals (const Eigen::MatrixBase< Derived > &A)
     Eigenvalues.

    template<typename Derived >

  cmat evects (const Eigen::MatrixBase< Derived > &A)
     Eigenvectors.
• template<typename Derived >
  std::pair < dyn_col_vect
  < double >, cmat > heig (const Eigen::MatrixBase< Derived > &A)
     Full eigen decomposition of Hermitian expression.

    template<typename Derived >

  dyn_col_vect< double > hevals (const Eigen::MatrixBase< Derived > &A)
     Hermitian eigenvalues.
• template<typename Derived >
  cmat hevects (const Eigen::MatrixBase< Derived > &A)
     Hermitian eigenvectors.
 \bullet \ \ \mathsf{template} \mathord{<} \mathsf{typename} \ \mathsf{Derived} >
  std::tuple < cmat, dyn mat
  < double >, cmat > svd (const Eigen::MatrixBase< Derived > &A)
     Full singular value decomposition.

    template<typename Derived >

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

    template<typename Derived >

  cmat svdU (const Eigen::MatrixBase< Derived > &A)
     Left singular vectors.
• template<typename Derived >
  cmat svdV (const Eigen::MatrixBase< Derived > &A)
     Right singular vectors.

    template<typename Derived >

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

    template<typename Derived >

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

    template<typename Derived >

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

    template<typename Derived >

  cmat logm (const Eigen::MatrixBase< Derived > &A)
     Matrix logarithm.

    template<typename Derived >

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

Matrix sin. • template<typename Derived > cmat cosm (const Eigen::MatrixBase< Derived > &A) Matrix cos. template<typename Derived > cmat spectralpowm (const Eigen::MatrixBase< Derived > &A, const cplx z) Matrix power. template<typename Derived > dyn mat< typename Derived::Scalar > powm (const Eigen::MatrixBase< Derived > &A, idx n) Matrix power. template<typename Derived > double schatten (const Eigen::MatrixBase< Derived > &A, idx p) Schatten norm. • template<typename OutputScalar , typename Derived > dyn mat< OutputScalar > cwise (const Eigen::MatrixBase< Derived > &A, OutputScalar(\*f)(const typename Derived::Scalar &)) Functor. template<typename T > dyn\_mat< typename T::Scalar > kron (const T &head) Kronecker product. • template<typename T , typename... Args> dyn\_mat< typename T::Scalar > kron (const T &head, const Args &...tail) Kronecker product. • template<typename Derived > dyn\_mat< typename Derived::Scalar > kron (const std::vector< Derived > &As) Kronecker product. • template<typename Derived > dyn\_mat< typename Derived::Scalar > kron (const std::initializer\_list< Derived > &As) Kronecker product.  $\bullet \ \ \text{template}{<} \text{typename Derived} >$ dyn\_mat< typename Derived::Scalar > kronpow (const Eigen::MatrixBase< Derived > &A, idx n) Kronecker power. template<typename Derived > dyn mat< typename Derived::Scalar > reshape (const Eigen::MatrixBase< Derived > &A, idx rows, idx cols) Reshape. template<typename Derived1 , typename Derived2 > dyn mat< typename Derived1::Scalar > comm (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)Commutator. template<typename Derived1 , typename Derived2 > dyn\_mat< typename</pre> Derived1::Scalar > anticomm (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)Anti-commutator. template<typename Derived > dyn\_mat< typename Derived::Scalar > prj (const Eigen::MatrixBase< Derived > &V) Projector. template<typename Derived > dyn mat< typename Derived::Scalar > grams (const std::vector< Derived > &Vs)

dyn\_mat< typename Derived::Scalar > grams (const std::initializer\_list< Derived > &Vs)

Gram-Schmidt orthogonalization.

template<typename Derived >

Gram-Schmidt orthogonalization.

• template<typename Derived >

dyn\_mat< typename Derived::Scalar > grams (const Eigen::MatrixBase< Derived > &A)

Gram-Schmidt orthogonalization.

std::vector< idx > n2multiidx (idx n, const std::vector< idx > &dims)

Non-negative integer index to multi-index.

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

Multi-index to non-negative integer index.

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

Multi-partite qudit ket.

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

Multi-partite qudit ket.

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

Projector onto multi-partite qudit ket.

cmat mprj (const std::vector < idx > &mask, idx 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 >

InputIterator::value\_type sum (InputIterator first, InputIterator last)

Element-wise sum of a range.

• template<typename InputIterator >

InputIterator::value\_type prod (InputIterator first, InputIterator last)

Element-wise product of a range.

• template<typename Derived >

```
dyn col vect< 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.

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

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, idx 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 > dyn_mat< 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 < idx, std::vector</li>
    double >, std::vector < cmat > > measure (const Eigen::MatrixBase < Derived > &A, const std::vector
```

cmat > &Ks, const std::vector < idx > &subsys, const std::vector < idx > &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 < idx, std::vector
```

< double >, std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const std $\leftarrow$  ::initializer\_list< cmat > &Ks, const std::vector< idx > &subsys, const std::vector< idx > &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 < idx, std::vector
```

< double >, std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks, const std::vector< idx > &subsys, const idx 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 < idx, std::vector
```

< double >, std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const std $\leftarrow$  ::initializer\_list< cmat > &Ks, const std::vector< idx > &subsys, const idx 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 < idx, std::vector
```

< double >, std::vector< cmat >> measure (const Eigen::MatrixBase< Derived > &A, const cmat &U, const std::vector< idx > &subsys, const std::vector< idx > &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 < idx, std::vector
```

< double >, std::vector< cmat >> measure (const Eigen::MatrixBase< Derived > &A, const cmat &U, const std::vector< idx > &subsys, const idx 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 < idx, 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 < idx, 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 < idx, 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.

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

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< long long int > x2contfrac (double x, idx n, idx cut=1e5)

Simple continued fraction expansion.

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

Real representation of a simple continued fraction.

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

Real representation of a simple continued fraction.

• idx gcd (idx m, idx n)

Greatest common divisor of two non-negative integers.

idx gcd (const std::vector < idx > &ns)

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

• idx lcm (idx m, idx n)

Least common multiple of two positive integers.

idx lcm (const std::vector < idx > &ns)

Least common multiple of a list of positive integers.

std::vector< idx > invperm (const std::vector< idx > &perm)

Inverse permutation.

• std::vector< idx > compperm (const std::vector< idx > &perm, const std::vector< idx > &sigma)

Compose permutations.

template<typename Derived1 , typename Derived2 >

dyn\_mat< typename

Derived1::Scalar > applyCTRL (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< idx > &ctrl, const std::vector< idx > &subsys, const std::vector< idx > &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 >

dyn\_mat< typename</pre>

Derived1::Scalar > applyCTRL (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< idx > &ctrl, const std::vector< idx > &subsys, idx 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 >

dyn\_mat< typename</pre>

Derived1::Scalar > apply (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< idx > &subsys, const std::vector< idx > &dims)

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

template<typename Derived1 , typename Derived2 >

dyn mat< typename

 $\label{lem:decomposition} Derived 1 :: Scalar > apply \; (const \; Eigen::MatrixBase < \; Derived 1 > \&state, \; const \; Eigen::MatrixBase < \; Derived 2 > \&A, \; const \; std::vector < idx > \&subsys, idx \; d=2)$ 

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

• template<typename Derived >

```
cmat apply (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 apply (const Eigen::MatrixBase< Derived > &rho, const std::vector< cmat > &Ks, const std::vector<
idx > &subsys, const std::vector< idx > &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 apply (const Eigen::MatrixBase< Derived > &rho, const std::vector< cmat > &Ks, const std::vector< idx > &subsys, idx 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 >

```
dyn_mat< typename Derived::Scalar > ptrace1 (const Eigen::MatrixBase< Derived > &A, const std ::vector < idx > &dims)
```

Partial trace.

• template<typename Derived >

```
dyn_mat< typename Derived::Scalar > ptrace2 (const Eigen::MatrixBase< Derived > &A, const std↔ ::vector< idx > &dims)
```

Partial trace.

• template<typename Derived >

```
dyn_mat< typename Derived::Scalar > ptrace (const Eigen::MatrixBase< Derived > &A, const std::vector<
idx > &subsys, const std::vector< idx > &dims)
```

Partial trace.

• template<typename Derived >

```
dyn_mat< typename Derived::Scalar > ptrace (const Eigen::MatrixBase< Derived > &A, const std::vector
idx > &subsys, idx d=2)
```

Partial trace.

• template<typename Derived >

```
dyn_mat< typename Derived::Scalar > ptranspose (const Eigen::MatrixBase< Derived > &A, const std
::vector< idx > &subsys, const std::vector< idx > &dims)
```

Partial transpose.

template<typename Derived >

```
dyn_mat< typename Derived::Scalar > ptranspose (const Eigen::MatrixBase< Derived > &A, const std ← ::vector < idx > &subsys, idx d=2)
```

Partial transpose.

• template<typename Derived >

System permutation.

template<typename Derived >

```
\frac{dyn\_mat}{<} typename \ Derived::Scalar > syspermute \ (const \ Eigen::MatrixBase < Derived > \&A, \ const \ std \leftarrow ::vector < idx > \&perm, \ idx \ d=2)
```

System permutation.

template<typename Derived >

Derived rand (idx rows, idx cols, double a=0, double b=1)

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

• template<>

dmat rand (idx rows, idx cols, double a, double b)

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

template<>

cmat rand (idx rows, idx 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)

idx randidx (idx a=std::numeric\_limits< idx >::min(), idx b=std::numeric\_limits< idx >::max())

Generates a random index (idx) uniformly distributed in the interval [a, b].

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

Derived randn (idx rows, idx cols, double mean=0, double sigma=1)

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

template<>

dmat randn (idx rows, idx 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 (idx rows, idx 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 (idx D)

Generates a random unitary matrix.

· cmat randV (idx Din, idx Dout)

Generates a random isometry matrix.

std::vector< cmat > randkraus (idx N, idx D)

Generates a set of random Kraus operators.

· cmat randH (idx D)

Generates a random Hermitian matrix.

ket randket (idx D)

Generates a random normalized ket (pure state vector)

cmat randrho (idx D)

Generates a random density matrix.

std::vector< idx > randperm (idx n)

Generates a random uniformly distributed permutation.

### **Variables**

• constexpr double chop = 1e-10

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

• constexpr double eps = 1e-12

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

constexpr idx maxn = 64

Maximum number of allowed qu(d)its (subsystems)

constexpr double pi = 3.141592653589793238462643383279502884

π

• constexpr double ee = 2.718281828459045235360287471352662497

Base of natural logarithm, e.

• constexpr double infty = std::numeric\_limits<double>::infinity()

Used to denote infinity in double precision.

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 Detailed Description

Quantum++ main namespace.

## 6.1.2 Typedef Documentation

6.1.2.1 using qpp::bra = typedef dyn\_row\_vect<cplx>

Complex (double precision) dynamic Eigen row vector.

6.1.2.2 using qpp::cmat = typedef dyn\_mat<cplx>

Complex (double precision) dynamic Eigen matrix.

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

Complex number in double precision.

6.1.2.4 using qpp::dmat = typedef dyn\_mat < double >

Real (double precision) dynamic Eigen matrix.

6.1.2.5 template<typename Scalar > using qpp::dyn\_col\_vect = 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 = dyn_col_vect<float>(2);
```

6.1.2.6 template<typename Scalar > using qpp::dyn\_mat = 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 = dyn_mat<float>(2,3);
```

6.1.2.7 template < typename Scalar > using qpp::dyn\_row\_vect = 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 = dyn_row_vect<float>(3);
```

6.1.2.8 using qpp::idx = typedef std::size\_t

Non-negative integer index.

6.1.2.9 using qpp::ket = typedef dyn col vect<cplx>

Complex (double precision) dynamic Eigen column vector.

## 6.1.3 Function Documentation

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

Matrix absolut value.

Parameters

Α	Eigen expression

## Returns

Matrix absolut value of A

6.1.3.2 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

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

Computes the absolut values squared of a column vector.

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

## Returns

Real vector consisting of the absolut values squared

6.1.3.4 template<typename Derived > dyn\_mat<typename Derived::Scalar> qpp::adjoint ( const Eigen::MatrixBase< Derived > & A )

## Adjoint.

#### **Parameters**

	T = .
A	Eigen expression
	0 1

#### Returns

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

6.1.3.5 template<typename Derived1 , typename Derived2 > dyn\_mat<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

6.1.3.6 template < typename Derived1 , typename Derived2 > dyn\_mat < typename Derived1::Scalar > qpp::apply ( const Eigen::MatrixBase < Derived2 > & A, const std::vector < idx > & subsys, const std::vector < idx > & 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

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

dims	Dimensions of the multi-partite system

Gate A applied to the part subsys of state

6.1.3.7 template < typename Derived1 , typename Derived2 > dyn\_mat < typename Derived1::Scalar > qpp::apply ( const Eigen::MatrixBase < Derived1 > & state, const Eigen::MatrixBase < Derived2 > & A, const std::vector < idx > & subsys, idx 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

6.1.3.8 template<typename Derived > cmat qpp::apply ( 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

6.1.3.9 template<typename Derived > cmat qpp::apply ( const Eigen::MatrixBase< Derived > & rho, const std::vector< cmat > & Ks, const std::vector< idx > & subsys, const std::vector< idx > & 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

Output density matrix after the action of the channel

6.1.3.10 template < typename Derived > cmat qpp::apply ( const Eigen::MatrixBase < Derived > & rho, const std::vector < cmat > & Ks, const std::vector < idx > & subsys, idx 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

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

6.1.3.12 template < typename Derived1 , typename Derived2 > dyn\_mat < typename Derived1::Scalar > qpp::applyCTRL ( const Eigen::MatrixBase < Derived1 > & state, const Eigen::MatrixBase < Derived2 > & A, const std::vector < idx > & ctrl, const std::vector < idx > & subsys, idx 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

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

6.1.3.13 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

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

Extracts orthogonal Kraus operators from Choi matrix.

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

Note

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

## **Parameters**

Α	Choi matrix
---	-------------

## Returns

Set of Kraus operators

6.1.3.15 template < typename Derived1 , typename Derived2 >  $dyn_mat$  < typename Derived1::Scalar >  $dyn_mat$  < typename Derived1 >  $dyn_mat$  < typename Derived2 >  $dyn_mat$  < typename Derived1 >  $dyn_mat$  < typename Derived2 >  $dyn_mat$  < typename Derived1 >  $dyn_mat$  < typename Derived2 >  $dyn_mat$  < typename Derived1 >  $dyn_mat$  < typename Deri

## Commutator.

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

Α	Eigen expression
В	Eigen expression

## Returns

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

6.1.3.16 std::vector < idx > qpp::compperm ( const std::vector < idx > & perm, const std::vector < idx > & sigma )

Compose permutations.

#### **Parameters**

perm	Permutation
sigma	Permutation

## Returns

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

6.1.3.17 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

6.1.3.18 template<typename Derived > dyn\_mat<typename Derived::Scalar> qpp::conjugate ( const Eigen::MatrixBase< Derived > & A )

Complex conjugate.

## **Parameters**

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

## Returns

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

6.1.3.19 double qpp::contfrac2x ( const std::vector < int > & cf, idx n )

Real representation of a simple continued fraction.

	cf	Integer vector containing the simple continued fraction expansion
ĺ	n	Number of terms considered in the continued fraction expansion. If $n$ 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

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

Real representation of a simple continued fraction.

## **Parameters**

cf	Integer vector containing the simple continued fraction expansion

#### Returns

Real representation of the simple continued fraction

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

## Matrix cos.

#### **Parameters**

Α	Eigen expression

## Returns

Matrix cosine of A

 $6.1.3.22 \quad template < typename \ Output Scalar \ , \ typename \ Derived > dyn\_mat < Output Scalar > qpp::cwise \ ( \ const \ Eigen::MatrixBase < Derived > \& \textit{A, Output Scalar}(*) (const \ typename \ Derived::Scalar \&) \textit{f })$ 

# Functor.

### **Parameters**

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

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

Determinant.

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.

6.1.3.24 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

6.1.3.25 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 chop

## Returns

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

6.1.3.26 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.

## **Parameters**

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

# Returns

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

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

X	Container
separator	Separator
start	Left marking
end	Right marking

## Returns

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

6.1.3.28 template<typename PointerType > internal::IOManipPointer<PointerType> qpp::disp ( const PointerType \* p, idx 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

6.1.3.29 template<typename Derived > std::pair<dyn\_col\_vect<cplx>, cmat> qpp::eig ( const Eigen::MatrixBase< Derived > & A )

Full eigen decomposition.

## **Parameters**

Α	Eigen expression

# Returns

Pair of: 1. Eigenvalues of *A*, as a complex dynamic column vector, and 2. Eigenvectors of *A*, as columns of a complex dynamic matrix

6.1.3.30 template < typename Derived > double qpp::entanglement ( const Eigen::MatrixBase < Derived > & A, const std::vector < idx > & 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::entropy()

Α	Eigen expression
dims	Dimensions of the bi-partite system

## Returns

Entanglement, with the logarithm in base 2

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

von-Neumann entropy of the density matrix A

**Parameters** 

Α
---

### Returns

von-Neumann entropy, with the logarithm in base 2

6.1.3.32 double qpp::entropy ( const std::vector < double > & prob )

Shannon entropy of the probability distribution prob.

## **Parameters**

prob	Real probability vector
------	-------------------------

## Returns

Shannon entropy, with the logarithm in base 2

6.1.3.33 template<typename Derived > dyn\_col\_vect<cplx> qpp::evals ( const Eigen::MatrixBase< Derived > & A )

Eigenvalues.

**Parameters** 

Α	Eigen expression

## Returns

Eigenvalues of A, as a complex dynamic column vector

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

Eigenvectors.

**Parameters** 

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

## Returns

Eigenvectors of A, as columns of a complex dynamic matrix

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

Matrix exponential.

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

### Returns

Matrix exponential of A

6.1.3.36 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)

6.1.3.37 idx qpp::gcd ( idx m, idx n )

Greatest common divisor of two non-negative integers.

**Parameters** 

m	Non-negative integer
n	Non-negative integer

## Returns

Greatest common divisor of *m* and *n* 

6.1.3.38 idx qpp::gcd ( const std::vector < idx > & 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

6.1.3.39 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()

A Eigen expression

Returns

G-concurrence

6.1.3.40 template < typename Derived > dyn\_mat < typename Derived::Scalar > qpp::grams ( const std::vector < Derived > & Vs )

Gram-Schmidt orthogonalization.

**Parameters** 

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

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

6.1.3.42 template<typename Derived > dyn\_mat<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

6.1.3.43 template<typename Derived > std::pair<dyn\_col\_vect<double>, cmat> qpp::heig ( const Eigen::MatrixBase< Derived > & A )

Full eigen decomposition of Hermitian expression.

A Eigen expression

Returns

Pair of: 1. Eigenvalues of A, as a real dynamic column vector, and 2. Eigenvectors of A, as columns of a complex dynamic matrix

6.1.3.44 template<typename Derived > dyn\_col\_vect<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

6.1.3.45 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

6.1.3.46 template < typename Derived >  $dyn_mat$  < 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

6.1.3.47 std::vector<idx> qpp::invperm ( const std::vector< idx > & perm )

Inverse permutation.

perm	Permutation

Inverse of the permutation perm

6.1.3.48 template<typename T > dyn\_mat<typename T::Scalar> qpp::kron ( const T & head )

Kronecker product.

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

**Parameters** 

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

## Returns

Its argument head

6.1.3.49 template<typename T , typename... Args> dyn\_mat<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

6.1.3.50 template<typename Derived > dyn\_mat<typename Derived::Scalar> qpp::kron ( const std::vector< Derived > & As )

Kronecker product.

## Parameters

As	std::vector of Eigen expressions
	otamosta a Ligari suprassiona

## 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

6.1.3.51 template<typename Derived > dyn\_mat<typename Derived::Scalar> qpp::kron ( const std::initializer\_list< Derived > & As )

Kronecker product.

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

6.1.3.52 template < typename Derived > dyn\_mat < typename Derived::Scalar > qpp::kronpow ( const Eigen::MatrixBase < Derived > & A, idx 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

6.1.3.53 idx qpp::lcm ( idx m, idx n )

Least common multiple of two positive integers.

#### **Parameters**

m	Positive integer
n	Positive integer

## Returns

Least common multiple of m and n

6.1.3.54 idx qpp::lcm ( const std::vector < idx > & 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

 $6.1.3.55 \quad template < typename \ Derived > \ dyn\_mat < typename \ Derived:: Scalar > \ qpp:: load \ ( \ const \ std:: string \ \& \ \textit{fname} \ )$ 

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**

A	Eigen expression
fname	Output file name

6.1.3.56 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)

6.1.3.57 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)

6.1.3.58 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");
```

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)

6.1.3.59 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
	9 1

#### Returns

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

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

Matrix logarithm.

**Parameters** 

Α	Eigen expression

## Returns

Matrix logarithm of A

6.1.3.61 template < typename Derived > double qpp::lognegativity ( const Eigen::MatrixBase < Derived > & A, const std::vector < idx > & 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

6.1.3.62 template < typename Derived > std::tuple < idx, std::vector < double >, std::vector < cmat > > qpp::measure ( const Eigen::MatrixBase < Derived > & A, const std::vector < cmat > & Ks, const std::vector < idx > & subsys, const std::vector < idx > & 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
Ks	Set of Kraus operators
subsys	Subsystem indexes that are measured
dims	Dimensions of the multi-partite system

#### Returns

Tuple consisiting of 1. Result of the measurement, 2. Vector of outcome probabilities and 3. Vector of post-measurement normalized states

6.1.3.63 template < typename Derived > std::tuple < idx, std::vector < double >, std::vector < cmat > > qpp::measure ( const Eigen::MatrixBase < Derived > & A, const std::initializer\_list < cmat > & Ks, const std::vector < idx > & subsys, const std::vector < idx > & 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.

#### **Parameters**

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 1. Result of the measurement,

- 1. Vector of outcome probabilities and 3. Vector of post-measurement normalized states
- 6.1.3.64 template < typename Derived > std::tuple < idx, std::vector < double >, std::vector < cmat > > qpp::measure ( const Eigen::MatrixBase < Derived > & A, const std::vector < cmat > & Ks, const std::vector < idx > & subsys, const idx 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 1. Result of the measurement,

1. Vector of outcome probabilities and 3. Vector of post-measurement normalized states

6.1.3.65 template < typename Derived > std::tuple < idx, std::vector < double >, std::vector < cmat > > qpp::measure ( const Eigen::MatrixBase < Derived > & A, const std::initializer\_list < cmat > & Ks, const std::vector < idx > & subsys, const idx 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 1. Result of the measurement,

- 1. Vector of outcome probabilities and 3. Vector of post-measurement normalized states
- 6.1.3.66 template < typename Derived > std::tuple < idx, std::vector < double >, std::vector < cmat > > qpp::measure ( const Eigen::MatrixBase < Derived > & A, const cmat & U, const std::vector < idx > & subsys, const std::vector < idx > & 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 1. Result of the measurement,

- 1. Vector of outcome probabilities and 3. Vector of post-measurement normalized states
- 6.1.3.67 template < typename Derived > std::tuple < idx, std::vector < double >, std::vector < cmat > > qpp::measure ( const Eigen::MatrixBase < Derived > & A, const cmat & U, const std::vector < idx > & subsys, const idx 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*.

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 1. Result of the measurement,

- 1. Vector of outcome probabilities and 3. Vector of post-measurement normalized states
- 6.1.3.68 template < typename Derived > std::tuple < idx, std::vector < double >, std::vector < cmat > > qpp::measure ( const Eigen::MatrixBase < Derived > & A, const std::vector < cmat > & Ks)

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

### **Parameters**

Α	Eigen expression
Ks	Set of Kraus operators

#### Returns

Tuple consisiting of 1. Result of the measurement,

- 1. Vector of outcome probabilities and 3. Vector of post-measurement normalized states
- 6.1.3.69 template < typename Derived > std::tuple < idx, 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 1. Result of the measurement,

- 1. Vector of outcome probabilities and 3. Vector of post-measurement normalized states
- 6.1.3.70 template < typename Derived > std::tuple < idx, 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 1. Result of the measurement,

1. Vector of outcome probabilities and 3. Vector of post-measurement normalized states

6.1.3.71 ket qpp::mket ( const std::vector < idx > & mask, const std::vector < idx > & 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

6.1.3.72 ket qpp::mket ( const std::vector < idx > & mask, idx 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*.

## **Parameters**

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

#### Returns

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

6.1.3.73 cmat qpp::mprj ( const std::vector < idx > & mask, const std::vector < idx > & 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

6.1.3.74 cmat qpp::mprj ( const std::vector < idx > & mask, idx 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*.

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

## Returns

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

6.1.3.75 idx qpp::multiidx2n ( const std::vector< idx > & midx, const std::vector< idx > & dims )

Multi-index to non-negative integer index.

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

## **Parameters**

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

## Returns

Non-negative integer index

6.1.3.76 std::vector<idx> qpp::n2multiidx ( idx n, const std::vector< idx> & dims )

Non-negative integer index to multi-index.

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

#### **Parameters**

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

## Returns

Multi-index of the same size as dims

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

Negativity of the bi-partite mixed state A.

## **Parameters**

Α	Eigen expression
dims	Dimensions of the bi-partite system

## Returns

Negativity

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

Frobenius norm.

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

Returns

Frobenius norm of A, as a real number

```
6.1.3.79 cplx qpp::omega (idx D) [inline]
```

D-th root of unity.

**Parameters** 

```
D Non-negative integer
```

Returns

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

```
6.1.3.80 constexpr cplx 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>
```

6.1.3.81 constexpr cplx 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>
```

6.1.3.82 template < typename Derived >  $dyn_mat$  < typename Derived::Scalar >  $dyn_mat$  < typename Derived >  $dyn_mat$  < ty

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

6.1.3.83 template < typename Derived > dyn\_mat < typename Derived::Scalar > qpp::prj ( const Eigen::MatrixBase < Derived > & V )

Projector.

Normalized projector onto state vector

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

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

Element-wise product of A.

## **Parameters**

Δ.	Final companies
A	Eigen expression

#### Returns

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

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

Element-wise product of a range.

#### **Parameters**

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

#### Returns

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

6.1.3.86 template < typename Derived > dyn\_mat < typename Derived::Scalar > qpp::ptrace ( const Eigen::MatrixBase < Derived > & A, const std::vector < idx > & subsys, const std::vector < idx > & 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

6.1.3.87 template<typename Derived > dyn\_mat<typename Derived::Scalar> qpp::ptrace ( const Eigen::MatrixBase< Derived > & A, const std::vector< idx > & subsys, idx d = 2)

## Partial trace.

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

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

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 subsystem A in a bi-partite system  $A \otimes B$ , as a dynamic matrix over the same scalar field as A

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

6.1.3.90 template<typename Derived > dyn\_mat<typename Derived::Scalar> qpp::ptranspose ( const Eigen::MatrixBase < Derived > & A, const std::vector< idx > & subsys, const std::vector< idx > & dims )

Partial transpose.

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

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

subsys	Subsystem indexes
dims	Dimensions of the multi-partite system

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

6.1.3.91 template<typename Derived > dyn\_mat<typename Derived::Scalar> qpp::ptranspose ( const Eigen::MatrixBase< Derived > & A, const std::vector< idx > & subsys, idx 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

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

Quantum mutual information between 2 subsystems of a composite system.

#### **Parameters**

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

6.1.3.93 template<typename Derived > double qpp::qmutualinfo ( const Eigen::MatrixBase< Derived > & A, const std::vector< idx > & subsysB, idx d = 2 )

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

d	Subsystem dimensions

Mutual information between the 2 subsystems

```
6.1.3.94 template < typename Derived > Derived qpp::rand ( idx rows, idx 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

```
6.1.3.95 template <> dmat qpp::rand ( idx rows, idx 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

```
6.1.3.96 template<> cmat qpp::rand ( idx rows, idx 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
Ī	а	Beginning of the interval, belongs to it
Ī	b	End of the interval, does not belong to it

Random complex matrix

6.1.3.97 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)

6.1.3.98 cmat qpp::randH ( idx D )

Generates a random Hermitian matrix.

#### **Parameters**

D	Dimension of the Hilbert space
	· ·

## Returns

Random Hermitian matrix

Generates a random index (idx) uniformly distributed in the interval [a, b].

## **Parameters**

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

## Returns

Random index (idx) uniformly distributed in the interval [a, b]

6.1.3.100 ket qpp::randket ( idx D )

Generates a random normalized ket (pure state vector)

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

## Returns

Random normalized ket

```
6.1.3.101 std::vector<cmat> qpp::randkraus ( idx N, idx 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

```
6.1.3.102 template < typename Derived > Derived qpp::randn ( idx rows, idx 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

```
6.1.3.103 template<> dmat qpp::randn ( idx rows, idx 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

6.1.3.104 template <> cmat qpp::randn ( idx rows, idx 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

```
6.1.3.105 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)

```
6.1.3.106 std::vector<idx> qpp::randperm ( idx 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

6.1.3.107 cmat qpp::randrho ( idx D )

Generates a random density matrix.

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

#### Returns

Random density matrix

6.1.3.108 cmat qpp::randU ( idx D )

Generates a random unitary matrix.

#### **Parameters**

D	Dimension of the Hilbert space

#### Returns

Random unitary

6.1.3.109 cmat qpp::randV ( idx Din, idx 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

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

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

Note

When  $\alpha \to 1$  the Renyi entropy converges to the von-Neumann entropy, with the logarithm in base 2

### **Parameters**

Α	Eigen expression
alpha	Non-negative real number, use qpp::infty for $\alpha = \infty$

## Returns

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

6.1.3.111 double qpp::renyi ( const std::vector < double > & prob, double alpha )

Renyi-  $\alpha$  entropy of the probability distribution *prob*, for  $\alpha \geq 0$ .

Note

When lpha 
ightarrow 1 the Renyi entropy converges to the Shannon entropy, with the logarithm in base 2

prob	Real probability vector
alpha	Non-negative real number, use qpp::infty for $\alpha = \infty$

#### Returns

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

6.1.3.112 template<typename Derived > dyn\_mat<typename Derived::Scalar> qpp::reshape ( const Eigen::MatrixBase< Derived > & A, idx rows, idx cols )

## Reshape.

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

#### **Parameters**

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

## Returns

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

6.1.3.113 template<typename Derived > dyn\_col\_vect<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

6.1.3.114 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()

Α	Eigen expression
fname	Output file name

6.1.3.115 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)

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

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

#### **Parameters**

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

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

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

## **Parameters**

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

6.1.3.118 template < typename Derived > double qpp::schatten ( const Eigen::MatrixBase < Derived > & A, idx p )

Schatten norm.

### **Parameters**

Α	Eigen expression
р	Integer, greater or equal to 1

#### Returns

Schatten-p norm of A, as a real number

6.1.3.119 template < typename Derived > cmat qpp::schmidtA ( const Eigen::MatrixBase < Derived > & A, const std::vector < idx > & dims )

Schmidt basis on Alice's side.

Α	Eigen expression
dims	Dimensions of the bi-partite system

## Returns

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

6.1.3.120 template < typename Derived > cmat qpp::schmidtB ( const Eigen::MatrixBase < Derived > & A, const std::vector < idx > & dims )

Schmidt basis on Bob's side.

#### **Parameters**

Α	Eigen expression
dims	Dimensions of the bi-partite system

#### Returns

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

6.1.3.121 template<typename Derived > dyn\_col\_vect<double> qpp::schmidtcoeff ( const Eigen::MatrixBase< Derived > & A, const std::vector< idx > & 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
Ì	dims	Dimensions of the bi-partite system

## Returns

Schmidt coefficients of A, as a real dynamic column vector

6.1.3.122 template < typename Derived > std::vector < double > qpp::schmidtprob ( const Eigen::MatrixBase < Derived > & A, const std::vector < idx > & dims )

Schmidt probabilities of the bi-partite pure state A.

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

### See also

qpp::schmidtcoeff()

Α	Eigen expression
dims	Dimensions of the bi-partite system

#### Returns

Real vector consisting of the Schmidt probabilites of A

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

Matrix sin.

**Parameters** 

Α	Eigen expression

# Returns

Matrix sine of A

6.1.3.124 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  $A^z$ 

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

Matrix square root.

# Parameters

Α	Eigen expression

#### Returns

Matrix square root of A

6.1.3.126 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

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

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

6.1.3.128 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

6.1.3.129 template<typename Derived > dyn\_col\_vect<double> qpp::svals ( const Eigen::MatrixBase< Derived > & A )

Singular values.

#### **Parameters**

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

# Returns

Singular values of  $\emph{A}$ , ordered in decreasing order, as a real dynamic column vector

6.1.3.130 template < typename Derived > std::tuple < cmat, dyn\_mat < double >, cmat > qpp::svd ( const Eigen::MatrixBase < Derived > & A )

Full singular value decomposition.

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

#### Returns

Tuple of: 1. Left sigular vectors of *A*, as columns of a complex dynamic matrix, 2. Singular values of *A*, ordered in decreasing order, as a real dynamic column vector, and 3. Right singular vectors of *A*, as columns of a complex dynamic matrix

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

Left singular vectors.

#### **Parameters**

Α	Eigen expression

#### Returns

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

6.1.3.132 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

6.1.3.133 template<typename Derived > dyn\_mat<typename Derived::Scalar> qpp::syspermute ( const Eigen::MatrixBase< Derived > & A, const std::vector< idx > & perm, const std::vector< idx > & 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

6.1.3.134 template<typename Derived > dyn\_mat<typename Derived::Scalar> qpp::syspermute ( const Eigen::MatrixBase< Derived > & A, const std::vector< idx > & perm, idx d = 2)

System permutation.

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

Α	Eigen expression
perm	Permutation
d	Subsystem dimensions

#### Returns

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

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

Trace.

#### **Parameters**

Α	Eigen expression

#### Returns

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

6.1.3.136 template<typename Derived > dyn\_mat<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

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

Tsallis- q entropy of the density matrix A, for  $q \ge 0$ .

Note

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

### **Parameters**

Α	Eigen expression
q	Non-negative real number

#### Returns

Tsallis- q entropy

6.1.3.138 double qpp::tsallis ( const std::vector< double > & prob, double q )

Tsallis- q entropy of the probability distribution prob, for  $q \ge 0$ .

Note

When  $q \to 1$  the Tsallis entropy converges to the Shannon entropy, with the logarithm in base e

prob	Real probability vector
q	Non-negative real number

#### Returns

Tsallis- q entropy

6.1.3.139 std::vector<long long int> qpp::x2contfrac ( double x, idx n, idx 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 cut

#### 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.

# 6.1.4 Variable Documentation

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

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

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

qpp::Codes const Singleton

Initializes the codes, see the class qpp::Codes

6.1.4.3 constexpr double qpp::ee = 2.718281828459045235360287471352662497

Base of natural logarithm, e.

6.1.4.4 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>
```

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

qpp::Gates const Singleton

Initializes the gates, see the class qpp::Gates

6.1.4.6 constexpr double qpp::infty = std::numeric\_limits < double > ::infinity()

Used to denote infinity in double precision.

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

qpp::Init const Singleton

Additional initializations/cleanups

6.1.4.8 constexpr idx qpp::maxn = 64

Maximum number of allowed qu(d)its (subsystems)

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

6.1.4.9 constexpr double qpp::pi = 3.141592653589793238462643383279502884

 $\pi$ 

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

qpp::RandomDevices Singleton

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

6.1.4.11 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

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

# 6.2.1 Detailed Description

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

# 6.3 qpp::internal Namespace Reference

Internal utility functions, do not use/modify.

# 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**

```
• idx _multiidx2n (const idx *midx, idx numdims, const idx *dims)

    template<typename Derived >

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

    template<typename Derived >

  bool <u>_check_vector</u> (const Eigen::MatrixBase< Derived > &A)
\bullet \ \ {\it template}{<} {\it typename Derived} >
  bool check row vector (const Eigen::MatrixBase< Derived > &A)

    template<typename Derived >

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

    template<typename T >
```

- bool \_check\_nonzero\_size (const T &x)
- bool <u>\_check\_dims</u> (const std::vector< idx > &dims)

void \_n2multiidx (idx n, idx numdims, const idx \*dims, idx \*result)

- $\bullet \ \ \mathsf{template} \mathord{<} \mathsf{typename} \ \mathsf{Derived} >$ bool \_check\_dims\_match\_mat (const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &A)
- template<typename Derived > bool <u>\_check\_dims\_match\_cvect</u> (const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &V)
- template<typename Derived > bool \_check\_dims\_match\_rvect (const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &V)
- bool check eq dims (const std::vector < idx > &dims, idx dim)
- bool \_check\_subsys\_match\_dims (const std::vector < idx > &subsys, const std::vector < idx > &dims)
- bool <u>\_check\_perm</u> (const std::vector < idx > &perm)
- template<typename Derived1 , typename Derived2 >dyn mat< typename Derived1::Scalar > kron2 (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)
- template<typename T > void variadic\_vector\_emplace (std::vector< T > &)
- template<typename T , typename First , typename... Args> void variadic\_vector\_emplace (std::vector< T > &v, First &&first, Args &&...args)

#### **Detailed Description** 6.3.1

Internal utility functions, do not use/modify.

#### 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 < idx > & dims )
- 6.3.2.3 template<typename Derived > bool qpp::internal::\_check\_dims\_match\_cvect ( const std::vector< idx > & dims, const Eigen::MatrixBase< Derived > & V)
- 6.3.2.4 template < typename Derived > bool qpp::internal:: check dims match mat ( const std::vector < idx > & dims, const Eigen::MatrixBase < Derived > & A )
- 6.3.2.5 template < typename Derived > bool qpp::internal:: check dims\_match\_rvect ( const std::vector < idx > & dims, const Eigen::MatrixBase< Derived > & V )
- 6.3.2.6 bool qpp::internal::\_check\_eq\_dims ( const std::vector < idx > & dims, idx dim )

6.3.2.7 template<typename T > bool qpp::internal::\_check\_nonzero\_size ( const T & x )
6.3.2.8 bool qpp::internal::\_check\_perm ( const std::vector < idx > & perm )
6.3.2.9 template<typename Derived > bool qpp::internal::\_check\_row\_vector ( const Eigen::MatrixBase < Derived > & A )
6.3.2.10 template<typename Derived > bool qpp::internal::\_check\_square\_mat ( const Eigen::MatrixBase < Derived > & A )
6.3.2.11 bool qpp::internal::\_check\_subsys\_match\_dims ( const std::vector < idx > & subsys, const std::vector < idx > & dims )
6.3.2.12 template<typename Derived > bool qpp::internal::\_check\_vector ( const Eigen::MatrixBase < Derived > & A )
6.3.2.13 template<typename Derived1, typename Derived2 > dyn\_mat<typename Derived1::Scalar > qpp::internal::\_kron2 ( const Eigen::MatrixBase < Derived1 > & A, const Eigen::MatrixBase < Derived2 > & B )
6.3.2.14 idx qpp::internal::\_multiidx2n ( const idx \* midx, idx numdims, const idx \* dims ) [inline]
6.3.2.15 void qpp::internal::\_n2multiidx ( idx n, idx numdims, const idx \* dims, idx \* result ) [inline]
6.3.2.16 template<typename T > void qpp::internal::variadic\_vector\_emplace ( std::vector < T > & )

6.3.2.17 template < typename T , typename First , typename... Args > void qpp::internal::variadic\_vector\_emplace (

std::vector < T > & v, First && first, Args &&... args )

Generated on Tue Dec 9 2014 18:17:46 for Quantum++ by Doxygen

# **Chapter 7**

# **Class Documentation**

# 7.1 qpp::Codes Class Reference

const Singleton class that defines quantum error correcting codes

#include <classes/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, idx 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, idx i ) const [inline]

Returns the codeword of the specified code.

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:

· classes/codes.h

# 7.2 qpp::Exception Class Reference

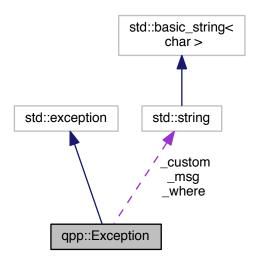
Generates custom exceptions, used when validating function parameters.

#include <classes/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,

 $\label{type::SUBSYS_MISMATCH_DIMS, Type::NOT_QUBIT\_GATE, Type::NOT\_QUBIT\_SUBSYS, Type::NOT\_BIPARTITE, \\$ 

Type::NO\_CODEWORD, Type::PERM\_INVALID, Type::PERM\_MISMATCH\_DIMS, Type::OUT\_OF\_RANG←

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**

void \_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())

DIMS\_INVALID std::vector<idx> representing the dimensions has zero size or contains zeros

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

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

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

**PERM\_MISMATCH\_DIMS** Size of the std::vector<idx> representing the permutation is different from the size of the std::vector<idx> representing the dimensions

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 void qpp::Exception::\_construct\_exception\_msg( )** [inline], [private]

Constructs the exception's description from its type.

See also

qpp::Exception::Type

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

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:

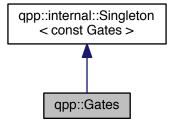
· classes/exception.h

# 7.3 qpp::Gates Class Reference

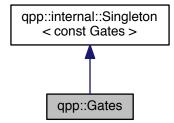
const Singleton class that implements most commonly used gates

#include <classes/gates.h>

Inheritance diagram for qpp::Gates:



Collaboration diagram for qpp::Gates:



#### **Public Member Functions**

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

• cmat FRED {cmat::Identity(8, 8)}

Toffoli gate.

Fredkin gate.

 cmat Rn (double theta, std::vector< double > n) const Rotation of theta about the 3-dimensional real unit vector n. • cmat Zd (idx D) const Generalized Z gate for qudits. • cmat Fd (idx D) const Fourier transform gate for qudits. • cmat Xd (idx D) const Generalized X gate for qudits. template<typename Derived = Eigen::MatrixXcd> Derived Id (idx D) const Identity gate. • template<typename Derived > dyn\_mat< typename Derived::Scalar > CTRL (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &ctrl, const std::vector< idx > &subsys, idx n, idx d=2) const Generates the multi-partite multiple-controlled-A gate in matrix form.  $\bullet \ \ \mathsf{template} \mathord{<} \mathsf{typename} \ \mathsf{Derived} >$ dyn\_mat< typename Derived::Scalar > expandout (const Eigen::MatrixBase< Derived > &A, idx pos, const std::vector< idx > &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 CNOT {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.

#### **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 > dyn\_mat<typename Derived::Scalar> qpp::Gates::CTRL ( const Eigen::MatrixBase< Derived > & A, const std::vector< idx > & ctrl, const std::vector< idx > & subsys, idx n, idx 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 > dyn\_mat<typename Derived::Scalar> qpp::Gates::expandout ( const Eigen::MatrixBase< Derived > & A, idx pos, const std::vector< idx > & 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(idx 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::ld(idx 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(idx 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(idx 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::CNOT {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.

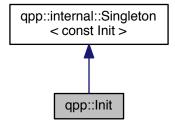
```
7.3.5.5 cmat qpp::Gates::H {cmat::Zero(2, 2)}
Hadamard gate.
7.3.5.6 cmat qpp::Gates::ld2 {cmat::ldentity(2, 2)}
Identity gate.
7.3.5.7 cmat qpp::Gates::S {cmat::Zero(2, 2)}
S gate.
7.3.5.8 cmat qpp::Gates::SWAP {cmat::Identity(4, 4)}
SWAP gate.
7.3.5.9 cmat qpp::Gates::T {cmat::Zero(2, 2)}
T gate.
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:
    · classes/gates.h
```

# 7.4 qpp::Init Class Reference

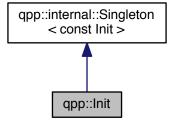
const Singleton class that performs additional initializations/cleanups

#include <classes/init.h>

Inheritance diagram for qpp::Init:



Collaboration diagram for qpp::Init:



# **Public Member Functions**

• Init ()

Additional initializations.

# **Private Member Functions**

Cleanups.

• ∼Init ()

# 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:

· classes/init.h

# 7.5 qpp::internal::IOManipEigen Class Reference

```
#include <internal/classes/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]

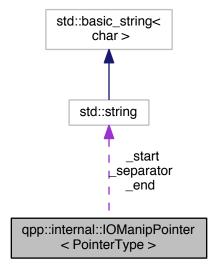
The documentation for this class was generated from the following file:

• internal/classes/iomanip.h

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

#include <internal/classes/iomanip.h>

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



#### **Public Member Functions**

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

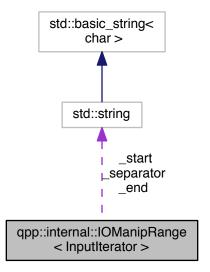
The documentation for this class was generated from the following file:

· internal/classes/iomanip.h

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

#include <internal/classes/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:

• internal/classes/iomanip.h

# 7.8 qpp::RandomDevices Class Reference

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

#include <classes/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:

• 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 <internal/classes/singleton.h>
```

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

• static T & get\_instance ()

#### **Protected Member Functions**

- Singleton ()
- virtual ∼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:

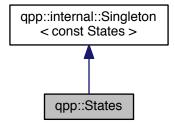
· internal/classes/singleton.h

# 7.10 qpp::States Class Reference

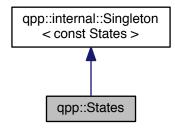
const Singleton class that implements most commonly used states

```
#include <classes/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:

    classes/states.h
```

# 7.11 qpp::Timer Class Reference

#### Measures time.

#include <classes/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 <a href="mailto:qpp::Timer::toc()">qpp::Timer::toc()</a>.

# 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:

· classes/timer.h

# **Chapter 8**

# **File Documentation**

# 8.1 classes/codes.h File Reference

Quantum error correcting codes.

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

Quantum++ main namespace.

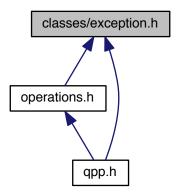
# 8.1.1 Detailed Description

Quantum error correcting codes.

# 8.2 classes/exception.h File Reference

Exceptions.

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

Quantum++ main namespace.

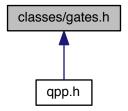
# 8.2.1 Detailed Description

Exceptions.

# 8.3 classes/gates.h File Reference

Quantum gates.

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

Quantum++ main namespace.

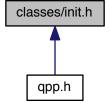
# 8.3.1 Detailed Description

Quantum gates.

# 8.4 classes/init.h File Reference

Initialization.

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

Quantum++ main namespace.

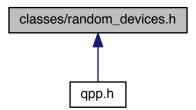
# 8.4.1 Detailed Description

Initialization.

# 8.5 classes/random\_devices.h File Reference

Random devices.

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



## **Classes**

• class qpp::RandomDevices

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

# **Namespaces**

• qpp

Quantum++ main namespace.

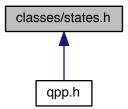
# 8.5.1 Detailed Description

Random devices.

# 8.6 classes/states.h File Reference

Quantum states.

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

Quantum++ main namespace.

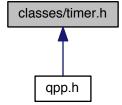
# 8.6.1 Detailed Description

Quantum states.

# 8.7 classes/timer.h File Reference

Timing.

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



#### **Classes**

class qpp::Timer

Measures time.

## **Namespaces**

• qpp

Quantum++ main namespace.

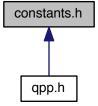
## 8.7.1 Detailed Description

Timing.

# 8.8 constants.h File Reference

#### Constants.

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



## **Namespaces**

• qpp

Quantum++ main namespace.

#### **Functions**

- constexpr cplx qpp::operator""\_i (unsigned long long int x)
  - User-defined literal for complex  $i = \sqrt{-1}$  (integer overload)
- constexpr cplx qpp::operator""\_i (long double x)

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

• cplx qpp::omega (idx D)

D-th root of unity.

#### **Variables**

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

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

• constexpr double qpp::eps = 1e-12

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

• constexpr idx qpp::maxn = 64

Maximum number of allowed qu(d)its (subsystems)

• constexpr double qpp::pi = 3.141592653589793238462643383279502884

π

constexpr double qpp::ee = 2.718281828459045235360287471352662497

Base of natural logarithm, e.

constexpr double <a href="mailto:qpp::infty">qpp::infty</a> = std::numeric\_limits<double>::infinity()

Used to denote infinity in double precision.

#### 8.8.1 Detailed Description

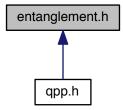
Constants.

# 8.9 entanglement.h File Reference

#### Entanglement functions.

```
#include <types.h>
#include <vector>
```

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



# Namespaces

qpp

Quantum++ main namespace.

### **Functions**

template<typename Derived >
 dyn\_col\_vect< double > qpp::schmidtcoeff (const Eigen::MatrixBase< Derived > &A, const std::vector< idx
 > &dims)

Schmidt coefficients of the bi-partite pure state A.

template < typename Derived >
 cmat qpp::schmidtA (const Eigen::MatrixBase < Derived > &A, const std::vector < idx > &dims)
 Schmidt basis on Alice's side.

template<typename Derived >
 cmat qpp::schmidtB (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)
 Schmidt basis on Bob's side.

template<typename Derived >
 std::vector< double > qpp::schmidtprob (const Eigen::MatrixBase< Derived > &A, const std::vector< idx >
 &dims)

Schmidt probabilities of the bi-partite pure state A.

template<typename Derived >
 double qpp::entanglement (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)
 Entanglement of the bi-partite pure state A.

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

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

template<typename Derived >
 double qpp::negativity (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)
 Negativity of the bi-partite mixed state A.

template<typename Derived >
 double qpp::lognegativity (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)
 Logarithmic negativity of the bi-partite mixed state A.

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

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

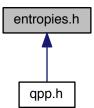
#### 8.9.1 Detailed Description

Entanglement functions.

# 8.10 entropies.h File Reference

Entropy functions.

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



#### **Namespaces**

• qpp

Quantum++ main namespace.

#### **Functions**

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

von-Neumann entropy of the density matrix A

double qpp::entropy (const std::vector< double > &prob)

Shannon entropy of the probability distribution prob.

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

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

Renyi-  $\alpha$  entropy of the density matrix A, for  $\alpha > 0$ .

double qpp::renyi (const std::vector< double > &prob, double alpha)

Renyi-  $\alpha$  entropy of the probability distribution prob, for  $\alpha \geq 0$ .

template<typename Derived >

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

Tsallis- q entropy of the density matrix A, for  $q \ge 0$ .

double qpp::tsallis (const std::vector< double > &prob, double q)

Tsallis- q entropy of the probability distribution prob, for  $q \ge 0$ .

• template<typename Derived >

double qpp::qmutualinfo (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsysA, const std::vector< idx > &subsysB, const std::vector< idx > &dims)

Quantum mutual information between 2 subsystems of a composite system.

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

double qpp::qmutualinfo (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsysA, const std::vector< idx > &subsysB, idx d=2)

Quantum mutual information between 2 subsystems of a composite system.

#### 8.10.1 Detailed Description

Entropy functions.

# 8.11 experimental/test.h File Reference

Experimental/test functions/classes.

# **Namespaces**

qpp

Quantum++ main namespace.

qpp::experimental

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

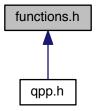
#### 8.11.1 Detailed Description

Experimental/test functions/classes.

#### 8.12 functions.h File Reference

Generic quantum computing functions.

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



#### **Namespaces**

qpp

Quantum++ main namespace.

• template<typename Derived >

Element-wise sum of A.
• template<typename Derived >

#### **Functions**

```
    template<typename Derived >

  dyn_mat< typename Derived::Scalar > qpp::transpose (const Eigen::MatrixBase< Derived > &A)
• template<typename Derived >
  dyn mat< typename Derived::Scalar > qpp::conjugate (const Eigen::MatrixBase< Derived > &A)
      Complex conjugate.
• template<typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::adjoint (const Eigen::MatrixBase< Derived > &A)
      Adjoint.

    template<typename Derived >

  dyn_mat< typename Derived::Scalar > qpp::inverse (const Eigen::MatrixBase< Derived > &A)
      Inverse.
\bullet \ \ \text{template}{<} \text{typename Derived} >
  Derived::Scalar qpp::trace (const Eigen::MatrixBase < Derived > &A)

    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.
```

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

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 >

  std::pair< dyn_col_vect< cplx >
  , cmat > qpp::eig (const Eigen::MatrixBase< Derived > &A)
     Full eigen decomposition.

    template<typename Derived >

  dyn col vect< cplx > qpp::evals (const Eigen::MatrixBase< Derived > &A)
     Eigenvalues.

    template<typename Derived >

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

    template<typename Derived >

  std::pair< dyn col vect
  < double >, cmat > qpp::heig (const Eigen::MatrixBase< Derived > &A)
     Full eigen decomposition of Hermitian expression.

    template<typename Derived >

  dyn_col_vect< 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 >

  std::tuple < cmat, dyn mat
  < double >, cmat > qpp::svd (const Eigen::MatrixBase< Derived > &A)
     Full singular value decomposition.

    template<typename Derived >

  dyn_col_vect< 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 >

  dyn mat< typename Derived::Scalar > qpp::powm (const Eigen::MatrixBase< Derived > &A, idx n)
     Matrix power.

    template<typename Derived >

  double <a href="mailto:qpp::schatten">qpp::schatten</a> (const Eigen::MatrixBase</a> Derived > &A, idx p)
     Schatten norm.
• template<typename OutputScalar , typename Derived >
  dyn mat< OutputScalar > qpp::cwise (const Eigen::MatrixBase< Derived > &A, OutputScalar(*f)(const
  typename Derived::Scalar &))
     Functor.

    template<typename T >

  dyn_mat< typename T::Scalar > qpp::kron (const T &head)
     Kronecker product.
• template<typename T , typename... Args>
  dyn_mat< typename T::Scalar > qpp::kron (const T &head, const Args &...tail)
     Kronecker product.

    template<typename Derived >

  dyn mat< typename Derived::Scalar > qpp::kron (const std::vector< Derived > &As)
     Kronecker product.
\bullet \ \ \text{template}{<} \text{typename Derived} >
  dyn_mat< typename Derived::Scalar > qpp::kron (const std::initializer_list< Derived > &As)
     Kronecker product.

    template<typename Derived >

  dyn_mat< typename Derived::Scalar > qpp::kronpow (const Eigen::MatrixBase< Derived > &A, idx n)
     Kronecker power.

    template<typename Derived >

  dyn mat< typename Derived::Scalar > gpp::reshape (const Eigen::MatrixBase< Derived > &A, idx rows,
  idx cols)
     Reshape.

    template<typename Derived1 , typename Derived2 >

  dyn mat< typename
  Derived1::Scalar > qpp::comm (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< De-
  rived2 > &B)
      Commutator.
• template<typename Derived1 , typename Derived2 >
  dyn_mat< typename
  Derived1::Scalar > qpp::anticomm (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase<
  Derived2 > &B)
     Anti-commutator.

    template<typename Derived >

  dyn_mat< typename Derived::Scalar > qpp::prj (const Eigen::MatrixBase< Derived > &V)
     Projector.

    template<typename Derived >

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

    template<typename Derived >

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

Gram-Schmidt orthogonalization.

template<typename Derived >
 dyn\_mat< typename Derived::Scalar > qpp::grams (const Eigen::MatrixBase< Derived > &A)

Gram-Schmidt orthogonalization.

std::vector< idx > qpp::n2multiidx (idx n, const std::vector< idx > &dims)

Non-negative integer index to multi-index.

• idx qpp::multiidx2n (const std::vector< idx > &midx, const std::vector< idx > &dims)

Multi-index to non-negative integer index.

ket qpp::mket (const std::vector< idx > &mask, const std::vector< idx > &dims)

Multi-partite qudit ket.

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

Multi-partite qudit ket.

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

Projector onto multi-partite qudit ket.

cmat qpp::mprj (const std::vector< idx > &mask, idx 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.

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

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

Computes the absolut values squared of a column vector.

• template<typename InputIterator >

InputIterator::value\_type qpp::sum (InputIterator first, InputIterator last)

Element-wise sum of a range.

• template<typename InputIterator >

InputIterator::value\_type qpp::prod (InputIterator first, InputIterator last)

Element-wise product of a range.

• template<typename Derived >

```
dyn_col_vect< 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.

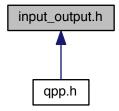
#### 8.12.1 Detailed Description

Generic quantum computing functions.

# 8.13 input\_output.h File Reference

Input/output functions.

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



#### **Namespaces**

• qpp

Quantum++ main namespace.

#### **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.

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

internal::IOManipPointer

< PointerType > qpp::disp (const PointerType \*p, idx 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 >

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

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

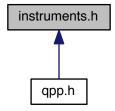
#### 8.13.1 Detailed Description

Input/output functions.

#### 8.14 instruments.h File Reference

Measurement functions.

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



#### **Namespaces**

• qpp

Quantum++ main namespace.

#### **Functions**

template<typename Derived >
 std::tuple< idx, std::vector
 < double >, std::vector< cmat > > qpp::measure (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks, const std::vector< idx > &subsys, const std::vector< idx > &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 < idx, std::vector
 < double >, std::vector < cmat > > qpp::measure (const Eigen::MatrixBase < Derived > &A, const std
 ::initializer\_list < cmat > &Ks, const std::vector < idx > &subsys, const std::vector < idx > &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 < idx, std::vector
 < double >, std::vector < cmat > > qpp::measure (const Eigen::MatrixBase < Derived > &A, const std
 ::vector < cmat > &Ks, const std::vector < idx > &subsys, const idx 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 < idx, std::vector
 < double >, std::vector < cmat > > qpp::measure (const Eigen::MatrixBase < Derived > &A, const std
 ::initializer\_list < cmat > &Ks, const std::vector < idx > &subsys, const idx 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< idx, std::vector
 < double >, std::vector< cmat > > qpp::measure (const Eigen::MatrixBase< Derived > &A, const cmat &U, const std::vector< idx > &subsys, const std::vector< idx > &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< idx, std::vector
 < double >, std::vector< cmat > > qpp::measure (const Eigen::MatrixBase< Derived > &A, const cmat &U, const std::vector< idx > &subsys, const idx 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< idx, 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 < idx, 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< idx, 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.

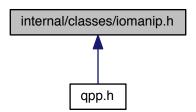
#### 8.14.1 Detailed Description

Measurement functions.

# 8.15 internal/classes/iomanip.h File Reference

Input/output manipulators.

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

Quantum++ main namespace.

• qpp::internal

Internal utility functions, do not use/modify.

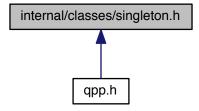
# 8.15.1 Detailed Description

Input/output manipulators.

# 8.16 internal/classes/singleton.h File Reference

Singleton pattern via CRTP.

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

Quantum++ main namespace.

· qpp::internal

Internal utility functions, do not use/modify.

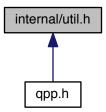
## 8.16.1 Detailed Description

Singleton pattern via CRTP.

#### 8.17 internal/util.h File Reference

Internal utility functions.

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



### **Namespaces**

• qpp

Quantum++ main namespace.

qpp::internal

Internal utility functions, do not use/modify.

#### **Functions**

- void qpp::internal::\_n2multiidx (idx n, idx numdims, const idx \*dims, idx \*result)
- idx qpp::internal::\_multiidx2n (const idx \*midx, idx numdims, const idx \*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< idx > &dims)
- template<typename Derived >
   bool qpp::internal::\_check\_dims\_match\_mat (const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >
   bool qpp::internal::\_check\_dims\_match\_cvect (const std::vector< idx > &dims, const Eigen::MatrixBase
   Derived > &V)
- template<typename Derived >
   bool qpp::internal::\_check\_dims\_match\_rvect (const std::vector< idx > &dims, const Eigen::MatrixBase
   Derived > &V)
- bool qpp::internal::\_check\_eq\_dims (const std::vector< idx > &dims, idx dim)
- bool qpp::internal::\_check\_subsys\_match\_dims (const std::vector < idx > &subsys, const std::vector < idx > &dims)

- bool qpp::internal::\_check\_perm (const std::vector< idx > &perm)
- template<typename Derived1 , typename Derived2 >

dyn mat< typename

 $\label{lem:decomposition} Derived1::Scalar > qpp::internal::\_kron2 \ (const \ Eigen::MatrixBase < Derived1 > \&A, \ const \ Eigen::MatrixBase < Derived1 >$ 

• 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)

#### 8.17.1 Detailed Description

Internal utility functions.

### 8.18 MATLAB/matlab.h File Reference

Input/output interfacing with MATLAB.

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

#### **Namespaces**

• qpp

Quantum++ main namespace.

#### **Functions**

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

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)

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

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)

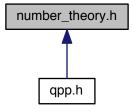
## 8.18.1 Detailed Description

Input/output interfacing with MATLAB.

# 8.19 number\_theory.h File Reference

Number theory functions.

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



## **Namespaces**

• qpp

Quantum++ main namespace.

## **Functions**

- std::vector< long long int > qpp::x2contfrac (double x, idx n, idx cut=1e5)

Simple continued fraction expansion.

double <a href="mailto:qpp::contfrac2x">qpp::contfrac2x</a> (const std::vector< int > &cf, idx n)

Real representation of a simple continued fraction.

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

Real representation of a simple continued fraction.

• idx qpp::gcd (idx m, idx n)

Greatest common divisor of two non-negative integers.

idx qpp::gcd (const std::vector< idx > &ns)

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

• idx qpp::lcm (idx m, idx n)

Least common multiple of two positive integers.

idx qpp::lcm (const std::vector< idx > &ns)

Least common multiple of a list of positive integers.

std::vector< idx > qpp::invperm (const std::vector< idx > &perm)

Inverse permutation.

• std::vector < idx > qpp::compperm (const std::vector < idx > &perm, const std::vector < idx > &sigma)

Compose permutations.

#### 8.19.1 Detailed Description

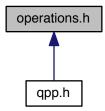
Number theory functions.

# 8.20 operations.h File Reference

Quantum operation functions.

#include <classes/exception.h>

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



#### **Namespaces**

dbb

Quantum++ main namespace.

#### **Functions**

template < typename Derived1 , typename Derived2 >

dyn\_mat< typename

Derived1::Scalar > qpp::applyCTRL (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< idx > &ctrl, const std::vector< idx > &subsys, const std::vector< idx > &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 >

dyn\_mat< typename

Derived1::Scalar > qpp::applyCTRL (const Eigen::MatrixBase< Derived1 > &state, const Eigen::Matrix← Base< Derived2 > &A, const std::vector< idx > &ctrl, const std::vector< idx > &subsys, idx 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 >

dyn\_mat< typename

Derived1::Scalar > qpp::apply (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< idx > &subsys, const std::vector< idx > &dims)

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

- template<typename Derived1 , typename Derived2 >

dyn\_mat< typename

Derived1::Scalar > qpp::apply (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< idx > &subsys, idx 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::apply (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::apply (const Eigen::MatrixBase< Derived > &rho, const std::vector< cmat > &Ks, const std ::vector< idx > &subsys, const std::vector< idx > &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 qpp::apply (const Eigen::MatrixBase< Derived > &rho, const std::vector< cmat > &Ks, const std::vector< idx > &subsys, idx 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 >

dyn\_mat< typename Derived::Scalar > qpp::ptrace1 (const Eigen::MatrixBase< Derived > &A, const std ← ::vector < idx > &dims)

Partial trace.

• template<typename Derived >

dyn\_mat< typename Derived::Scalar > qpp::ptrace2 (const Eigen::MatrixBase< Derived > &A, const std← ::vector< idx > &dims)

Partial trace.

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

dyn\_mat< typename Derived::Scalar > qpp::ptrace (const Eigen::MatrixBase< Derived > &A, const std $\leftarrow$  ::vector< idx > &subsys, const std::vector< idx > &dims)

Partial trace.

template<typename Derived >

Partial trace.

• template<typename Derived >

 $\label{lem:dyn_mat} $$ dyn_mat< typename Derived::Scalar > qpp::ptranspose (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsys, const std::vector< idx > &dims)$ 

Partial transpose.

• template<typename Derived >

dyn\_mat< typename Derived::Scalar > qpp::ptranspose (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsys, idx d=2)

Partial transpose.

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

dyn\_mat< typename Derived::Scalar > qpp::syspermute (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &perm, const std::vector< idx > &dims)

System permutation.

template<typename Derived >

 $dyn_mat < typename Derived::Scalar > qpp::syspermute (const Eigen::MatrixBase < Derived > &A, const std::vector < idx > &perm, idx d=2)$ 

System permutation.

#### 8.20.1 Detailed Description

Quantum operation functions.

# 8.21 qpp.h File Reference

Quantum++ main header file, includes all other necessary headers.

```
#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 "types.h"
#include "constants.h"
#include "classes/exception.h"
#include "internal/util.h"
#include "internal/classes/iomanip.h"
#include "input_output.h"
#include "internal/classes/singleton.h"
#include "classes/init.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

Quantum++ main namespace.

#### **Variables**

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

```
qpp::Gates const Singleton
```

qpp::Codes const Singleton

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

qpp::States const Singleton

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

qpp::RandomDevices Singleton

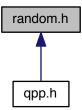
#### 8.21.1 Detailed Description

Quantum++ main header file, includes all other necessary headers.

## 8.22 random.h File Reference

Randomness-related functions.

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



# **Namespaces**

qpp

Quantum++ main namespace.

#### **Functions**

template<typename Derived >
 Derived qpp::rand (idx rows, idx cols, double a=0, double b=1)

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

template<>

dmat qpp::rand (idx rows, idx 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 (idx rows, idx cols, double a, double b)

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

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

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

• idx qpp::randidx (idx a=std::numeric\_limits< idx >::min(), idx b=std::numeric\_limits< idx >::max())

Generates a random index (idx) uniformly distributed in the interval [a, b].

template<typename Derived >

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

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

template<>

dmat qpp::randn (idx rows, idx 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 (idx rows, idx 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 qpp::randn (double mean=0, double sigma=1)

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

cmat qpp::randU (idx D)

Generates a random unitary matrix.

cmat qpp::randV (idx Din, idx Dout)

Generates a random isometry matrix.

std::vector< cmat > qpp::randkraus (idx N, idx D)

Generates a set of random Kraus operators.

cmat qpp::randH (idx D)

Generates a random Hermitian matrix.

ket qpp::randket (idx D)

Generates a random normalized ket (pure state vector)

cmat qpp::randrho (idx D)

Generates a random density matrix.

std::vector< idx > qpp::randperm (idx n)

Generates a random uniformly distributed permutation.

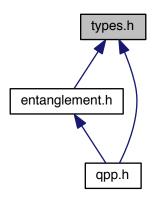
#### 8.22.1 Detailed Description

Randomness-related functions.

# 8.23 types.h File Reference

Type aliases.

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



#### **Namespaces**

• qpp

Quantum++ main namespace.

#### **Typedefs**

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

Complex number in double precision.

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

```
using <a href="mailto:qpp::dyn_mat">qpp::dyn_mat</a> = Eigen::Matrix< Scalar, Eigen::Dynamic, Eigen::Dynamic >
```

Dynamic Eigen matrix over the field specified by Scalar.

• template<typename Scalar >

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

Dynamic Eigen column vector over the field specified by Scalar.

• template<typename Scalar >

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

Dynamic Eigen row vector over the field specified by Scalar.

using qpp::ket = dyn\_col\_vect< cplx >

Complex (double precision) dynamic Eigen column vector.

using qpp::bra = dyn\_row\_vect< cplx >

Complex (double precision) dynamic Eigen row vector.

using qpp::cmat = dyn\_mat< cplx >

Complex (double precision) dynamic Eigen matrix.

using qpp::dmat = dyn\_mat < double >

Real (double precision) dynamic Eigen matrix.

• using qpp::idx = std::size\_t

Non-negative integer index.

#### 8.23.1 Detailed Description

Type aliases.

# Index

absm	qpp::Exception, 71
qpp, 24	DIMS_MISMATCH_VECTOR
abssq	qpp::Exception, 71
qpp, 24	DIMS_NOT_EQUAL
adjoint	qpp::Exception, 71
qpp, 25	det
anticomm	qpp, 30
qpp, 25	disp
apply	qpp, 31, 32
qpp, 25–27	dmat
	qpp, <mark>23</mark>
bra	
qpp, 23	ee
	qpp, 63
CUSTOM_EXCEPTION	eig
qpp::Exception, 72	qpp, 32
choi	entanglement
qpp, 28	qpp, 32
choi2kraus	entropy
qpp, 28	qpp, <mark>33</mark>
chop	eps
qpp, 63	qpp, <mark>63</mark>
cmat	evals
qpp, 23	qpp, <mark>33</mark>
codes	evects
qpp, 63	qpp, <mark>33</mark>
comm	expm
qpp, 28	qpp, <mark>33</mark>
compperm	=»./= o=.
qpp, 29	FIVE_QUBIT
concurrence	qpp::Codes, 68
qpp, 29	funm
conjugate	qpp, 35
qpp, 29	and
contfrac2x	gcd
qpp, 29, 30	qpp, 35
cosm	gconcurrence
qpp, 30	qpp, 35
cplx	grams
qpp, 23	qpp, 36
cwise	gt
qpp, 30	qpp, 63
<b>"</b> 1 ?	heig
DIMS_INVALID	qpp, 36
qpp::Exception, 71	hevals
DIMS_MISMATCH_CVECTOR	qpp, 37
qpp::Exception, 71	hevects
DIMS MISMATCH MATRIX	qpp, 37
qpp::Exception, 71	<b>4ρρ, 3</b> 7
DIMS MISMATCH RVECTOR	idx
•	-

122 INDEX

qpp, 24	NO CODEWORD
infty	qpp::Exception, 72
qpp, 63	NOT BIPARTITE
init	qpp::Exception, 71
	NOT_QUBIT_GATE
qpp, 64	
inverse	qpp::Exception, 71
qpp, 37	NOT_QUBIT_SUBSYS
invperm	qpp::Exception, 71
qpp, 37	negativity
	qpp, 46
ket	norm
qpp, 24	qpp, 46
kron	
qpp, 38	OUT_OF_RANGE
kronpow	qpp::Exception, 72
qpp, 39	omega
	qpp, 47
lcm	
qpp, 39	PERM_INVALID
load	qpp::Exception, 72
qpp, 39	PERM MISMATCH DIMS
logdet	qpp::Exception, 72
qpp, 41	pi
logm	qpp, 64
qpp, 41	powm
	•
lognegativity	qpp, 47
qpp, 41	prj
MATRIX MICHATOLI OLIDOVO	qpp, 47
MATRIX_MISMATCH_SUBSYS	prod
qpp::Exception, 71	qpp, 48
MATRIX_NOT_CVECTOR	ptrace
qpp::Exception, 71	qpp, 48
MATRIX_NOT_RVECTOR	ptrace1
qpp::Exception, 71	qpp, 49
MATRIX_NOT_SQUARE	ptrace2
qpp::Exception, 71	qpp, 49
MATRIX_NOT_SQUARE_OR_CVECTOR	ptranspose
qpp::Exception, 71	qpp, 49, 50
MATRIX_NOT_SQUARE_OR_RVECTOR	11 7 7
qpp::Exception, 71	qmutualinfo
MATRIX NOT SQUARE OR VECTOR	qpp, <u>50</u>
qpp::Exception, 71	qpp, 13
MATRIX NOT VECTOR	absm, 24
app::Exception, 71	abssq, 24
	adjoint, 25
maxn	anticomm, 25
qpp, 64	
measure	apply, 25–27
qpp, 41–44	bra, 23
mket	choi, 28
qpp, 44, 45	choi2kraus, 28
mprj	chop, 63
qpp, 45	cmat, 23
multiidx2n	codes, 63
qpp, 46	comm, 28
	compperm, 29
n2multiidx	concurrence, 29
qpp, 46	
qpp, 40	conjugate, 29
NINE_QUBIT_SHOR	
	conjugate, 29

INDEX 123

cplx, 23	randrho, 54
cwise, 30	rdevs, 64
det, 30	renyi, 55
disp, 31, 32	reshape, 56
dmat, 23	rho2pure, 56
ee, 63	save, 56
eig, 32	schatten, 57
entanglement, 32	schmidtcoeff, 58 schmidtprob, 58
entropy, 33	sinm, 59
eps, 63	spectralpowm, 59
evals, 33	sqrtm, 59
evects, 33	st, 64
expm, 33	sum, 59, 60
funm, 35	super, 60
gcd, 35 gconcurrence, 35	svals, 60
grams, 36	svd, 60
-	syspermute, 61
gt, 63 heig, 36	trace, 62
hevals, 37	transpose, 62
hevects, 37	tsallis, 62
idx, 24	x2contfrac, 63
infty, 63	qpp::Codes
init, 64	FIVE_QUBIT, 68
inverse, 37	NINE_QUBIT_SHOR, 68
invperm, 37	SEVEN_QUBIT_STEANE, 68
ket, 24	qpp::Exception
kron, 38	CUSTOM_EXCEPTION, 72
kronpow, 39	DIMS_INVALID, 71
lcm, 39	DIMS_MISMATCH_CVECTOR, 71
load, 39	DIMS_MISMATCH_MATRIX, 71
logdet, 41	DIMS_MISMATCH_RVECTOR, 71
logm, 41	DIMS_MISMATCH_VECTOR, 71
lognegativity, 41	DIMS_NOT_EQUAL, 71
maxn, 64	MATRIX_MISMATCH_SUBSYS, 71
measure, 41-44	MATRIX_NOT_CVECTOR, 71
mket, 44, 45	MATRIX_NOT_RVECTOR, 71
mprj, 45	MATRIX_NOT_SQUARE, 71  MATRIX NOT SQUARE OR CVECTOR, 71
multiidx2n, 46	MATRIX NOT SQUARE OR RVECTOR, 71
n2multiidx, 46	MATRIX NOT SQUARE OR VECTOR, 71
negativity, 46	MATRIX_NOT_VECTOR, 71
norm, 46	NO CODEWORD, 72
omega, 47	NOT BIPARTITE, 71
pi, 64	NOT_QUBIT_GATE, 71
powm, 47	NOT_QUBIT_SUBSYS, 71
prj, 47	OUT OF RANGE, 72
prod, 48	PERM INVALID, 72
ptrace, 48	PERM MISMATCH DIMS, 72
ptrace1, 49	SUBSYS_MISMATCH_DIMS, 71
ptrace2, 49	TYPE_MISMATCH, 72
ptranspose, 49, 50	UNDEFINED_TYPE, 72
qmutualinfo, 50	UNKNOWN_EXCEPTION, 71
rand, 51, 52	ZERO_SIZE, 71
randidx, 52	
randket, 52	rand
randkraus, 53	qpp, 51, 52
randn, 53, 54	randidx
randperm, 54	qpp, 52

124 INDEX

randket qpp, 52	UNDEFINED_TYPE qpp::Exception, 72
randkraus qpp, 53	UNKNOWN_EXCEPTION  qpp::Exception, 71
randn qpp, 53, 54	x2contfrac qpp, 63
randperm qpp, 54 randrho	ZERO_SIZE
qpp, 54	qpp::Exception, 71
qpp, 64 renyi	
qpp, 55 reshape	
qpp, 56 rho2pure	
qpp, 56	
SEVEN_QUBIT_STEANE qpp::Codes, 68	
SUBSYS_MISMATCH_DIMS  qpp::Exception, 71	
save	
qpp, 56 schatten	
qpp, 57 schmidtcoeff	
qpp, 58 schmidtprob	
qpp, 58 sinm	
qpp, 59 spectralpowm	
qpp, 59 sqrtm	
qpp, 59 st	
qpp, 64 sum	
qpp, 59, 60 super	
qpp, 60 svals	
qpp, 60 svd	
qpp, 60 syspermute	
qpp, 61	
TYPE_MISMATCH app::Exception, 72	
trace qpp, 62	
transpose qpp, 62	
tsallis qpp, 62	