Quantum++ v0.1

Generated by Doxygen 1.8.7

Sun Dec 21 2014 06:04:35

Contents

1	Qua	ntum++														1	
2	Nam	nespace	Index													5	;
	2.1	Names	space List					 			 	 	 	 	 	. 5	,
3	Hier	archica	l Index													7	,
	3.1	Class I	Hierarchy					 			 	 	 	 	 	. 7	,
4	Clas	ss Index														9)
	4.1	Class I	List					 			 	 	 	 	 	. 9)
5	File	Index														11	
	5.1	File Lis	st					 			 	 	 	 	 	. 11	
6	Nam	nespace	Docume	ntatio	on											13	}
	6.1	qpp Na	amespace	Refe	rence			 			 	 	 	 	 	. 13	;
		6.1.1	Detailed	Desc	cription	n .		 			 	 	 	 	 	. 23	;
		6.1.2	Typedef	Docu	menta	ation		 			 	 	 	 	 	. 23	;
			6.1.2.1	bra				 		 	 	 	 	 	 	. 23	;
			6.1.2.2	cma	at			 		 	 	 	 	 	 	. 23	;
			6.1.2.3	cpl	x			 		 	 	 	 	 	 	. 24	Ļ
			6.1.2.4	dma	at			 			 	 	 	 	 	. 24	Ļ
			6.1.2.5	dyn	_col_	vect		 			 	 	 	 	 	. 24	Ļ
			6.1.2.6	dyn	_mat			 			 	 	 	 	 	. 24	Ļ
			6.1.2.7	dyn	_row_	_vect	t .	 			 	 	 	 	 	. 24	Ļ
			6.1.2.8	idx				 			 	 	 	 	 	. 24	ļ
			6.1.2.9	ket				 			 	 	 	 	 	. 24	Ļ
		6.1.3	Function	n Doci	ument	atior	١.	 			 	 	 	 	 	. 24	Ļ
			6.1.3.1	abs	sm			 			 	 	 	 	 	. 24	Ļ
			6.1.3.2	abs	sq			 			 	 	 	 	 	. 25	,
			6.1.3.3	abs	sq			 			 	 	 	 	 	. 25	,
			6.1.3.4		oint .												,
			6135	anti	icomm	า										25	

iv CONTENTS

6.1.3.6	apply	26
6.1.3.7	apply	26
6.1.3.8	apply	26
6.1.3.9	apply	27
6.1.3.10	apply	27
6.1.3.11	applyCTRL	27
6.1.3.12	applyCTRL	28
6.1.3.13	choi2kraus	28
6.1.3.14	choi2super	29
6.1.3.15	comm	29
6.1.3.16	compperm	29
6.1.3.17	concurrence	30
6.1.3.18	conjugate	31
6.1.3.19	contfrac2x	31
6.1.3.20	contfrac2x	31
6.1.3.21	cosm	31
6.1.3.22	cwise	32
6.1.3.23	det	32
6.1.3.24	dirsum	32
6.1.3.25	dirsum	33
6.1.3.26	dirsum	33
6.1.3.27	dirsum	33
6.1.3.28	dirsumpow	34
6.1.3.29	disp	34
6.1.3.30	disp	34
6.1.3.31	disp	34
6.1.3.32	disp	35
6.1.3.33	disp	35
6.1.3.34	eig	35
6.1.3.35	entanglement	36
6.1.3.36	entropy	36
6.1.3.37	entropy	36
6.1.3.38	evals	37
6.1.3.39	evects	37
6.1.3.40	expm	37
6.1.3.41	funm	37
6.1.3.42	gcd	38
6.1.3.43	gcd	38
6.1.3.44	gconcurrence	38
6.1.3.45	grams	38

CONTENTS

6.1.3.46	grams	39
6.1.3.47	grams	39
6.1.3.48	heig	39
6.1.3.49	hevals	40
6.1.3.50	hevects	40
6.1.3.51	inverse	40
6.1.3.52	invperm	40
6.1.3.53	kraus2choi	41
6.1.3.54	kraus2super	41
6.1.3.55	kron	41
6.1.3.56	kron	42
6.1.3.57	kron	42
6.1.3.58	kron	42
6.1.3.59	kronpow	43
6.1.3.60	lcm	43
6.1.3.61	lcm	43
6.1.3.62	load	43
6.1.3.63	loadMATLABmatrix	44
6.1.3.64	loadMATLABmatrix	44
6.1.3.65	loadMATLABmatrix	45
6.1.3.66	logdet	45
6.1.3.67	logm	45
6.1.3.68	lognegativity	45
6.1.3.69	measure	46
6.1.3.70	measure	46
6.1.3.71	measure	47
6.1.3.72	measure	47
6.1.3.73	measure	47
6.1.3.74	measure	48
6.1.3.75	measure	48
6.1.3.76	measure	48
6.1.3.77	measure	49
6.1.3.78	mket	49
6.1.3.79	mket	49
6.1.3.80	mprj	50
6.1.3.81	mprj	50
6.1.3.82	multiidx2n	50
6.1.3.83	n2multiidx	50
6.1.3.84	negativity	51
6.1.3.85	norm	51

vi CONTENTS

6.1.3.86	omega	51
6.1.3.87	operator"""_i	51
6.1.3.88	operator"""_i	52
6.1.3.89	powm	52
6.1.3.90	prj	52
6.1.3.91	prod	52
6.1.3.92	prod	52
6.1.3.93	ptrace	53
6.1.3.94	ptrace	53
6.1.3.95	ptrace1	53
6.1.3.96	ptrace2	54
6.1.3.97	ptranspose	54
6.1.3.98	ptranspose	54
6.1.3.99	qmutualinfo	55
6.1.3.100) qmutualinfo	55
6.1.3.101	I rand	55
6.1.3.102	2 rand	55
6.1.3.103	3 rand	56
6.1.3.104	1 rand	56
6.1.3.105	5 randH	56
6.1.3.106	S randidx	57
6.1.3.107	7 randket	57
6.1.3.108	3 randkraus	57
6.1.3.109	Prandn	57
6.1.3.110	randn	58
6.1.3.111	randn	58
6.1.3.112	2 randn	58
6.1.3.113	3 randperm	59
6.1.3.114	frandrho	59
6.1.3.115	5 randU	59
6.1.3.116	S randV	59
6.1.3.117	⁷ renyi	60
6.1.3.118	3 renyi	60
6.1.3.119	Preshape	60
6.1.3.120	rho2pure	61
6.1.3.121	save	61
6.1.3.122	2 saveMATLABmatrix	61
6.1.3.123	B saveMATLABmatrix	62
6.1.3.124	saveMATLABmatrix	62
6.1.3.125	schatten	62

CONTENTS vii

	6.1.3.126 schmidtA	62
	6.1.3.127 schmidtB	63
	6.1.3.128 schmidtcoeffs	63
	6.1.3.129 schmidtprobs	63
	6.1.3.130 sinm	64
	6.1.3.131 spectralpowm	64
	6.1.3.132 sqrtm	64
	6.1.3.133 sum	64
	6.1.3.134 sum	65
	6.1.3.135 super2choi	65
	6.1.3.136 svals	65
	6.1.3.137 svd	65
	6.1.3.138 svdU	66
	6.1.3.139 svdV	66
	6.1.3.140 syspermute	66
	6.1.3.141 syspermute	66
	6.1.3.142 trace	67
	6.1.3.143 transpose	67
	6.1.3.144 tsallis	67
	6.1.3.145 tsallis	67
	6.1.3.146 x2contfrac	68
6.1.4	Variable Documentation	68
	6.1.4.1 chop	68
	6.1.4.2 codes	68
	6.1.4.3 ee	68
	6.1.4.4 eps	68
	6.1.4.5 gt	68
	6.1.4.6 infty	69
	6.1.4.7 init	69
	6.1.4.8 maxn	69
	6.1.4.9 pi	69
	6.1.4.10 rdevs	69
	6.1.4.11 st	69
qpp::e	perimental Namespace Reference	69
6.2.1	Detailed Description	69
qpp::in	ternal Namespace Reference	69
6.3.1	Detailed Description	70
6.3.2	Function Documentation	70
	6.3.2.1 _check_col_vector	70
	6.3.2.2 _check_dims	70

6.2

6.3

viii CONTENTS

			6.3.2.3 _check_dims_match_cvect	0
			6.3.2.4 _check_dims_match_mat	'0
			6.3.2.5 _check_dims_match_rvect	'0
			6.3.2.6 _check_eq_dims	'1
			6.3.2.7 _check_nonzero_size	'1
			6.3.2.8 _check_perm	'1
			6.3.2.9 _check_row_vector	'1
			6.3.2.10 _check_square_mat	'1
			6.3.2.11 _check_subsys_match_dims	'1
			6.3.2.12 _check_vector	'1
			6.3.2.13 _dirsum2	'1
			6.3.2.14 _kron2	'1
			6.3.2.15 _multiidx2n	'1
			6.3.2.16 _n2multiidx	'1
			6.3.2.17 variadic_vector_emplace	'1
			6.3.2.18 variadic_vector_emplace	'1
7	Clas	e Docu	mentation 7	72
•	7.1		odes Class Reference	
	7.1	7.1.1	Detailed Description	
		7.1.2	Member Enumeration Documentation	
		7.1.2	7.1.2.1 Type	
		7.1.3	Constructor & Destructor Documentation	
		7.1.0	7.1.3.1 Codes	
		7.1.4	Member Function Documentation	
			7.1.4.1 codeword	
		7.1.5	Friends And Related Function Documentation	
		7.1.0	7.1.5.1 internal::Singleton < const Codes >	
	7.2	app::F	xception Class Reference	
		7.2.1	Detailed Description	
		7.2.2	Member Enumeration Documentation	
			7.2.2.1 Type	
		7.2.3	Constructor & Destructor Documentation	
			7.2.3.1 Exception	
			7.2.3.2 Exception	
		7.2.4	Member Function Documentation	
		•	7.2.4.1 _construct_exception_msg	
			7.2.4.2 what	
		7.2.5	Member Data Documentation	
		-	7.2.5.1 _custom	
			_	_

CONTENTS

		7.2.5.2 _msg
		7.2.5.3 _type
		7.2.5.4 _where
7.3	qpp::G	ates Class Reference
	7.3.1	Detailed Description
	7.3.2	Constructor & Destructor Documentation
		7.3.2.1 Gates
	7.3.3	Member Function Documentation
		7.3.3.1 CTRL
		7.3.3.2 expandout
		7.3.3.3 Fd
		7.3.3.4 ld
		7.3.3.5 Rn
		7.3.3.6 Xd
		7.3.3.7 Zd
	7.3.4	Friends And Related Function Documentation
		7.3.4.1 internal::Singleton < const Gates >
	7.3.5	Member Data Documentation
		7.3.5.1 CNOT
		7.3.5.2 CNOTba
		7.3.5.3 CZ
		7.3.5.4 FRED
		7.3.5.5 H
		7.3.5.6 ld2
		7.3.5.7 S
		7.3.5.8 SWAP
		7.3.5.9 T
		7.3.5.10 TOF
		7.3.5.11 X
		7.3.5.12 Y
		7.3.5.13 Z
7.4	qpp::In	it Class Reference
	7.4.1	Detailed Description
	7.4.2	Constructor & Destructor Documentation
		7.4.2.1 Init
		7.4.2.2 ~Init
	7.4.3	Friends And Related Function Documentation
		7.4.3.1 internal::Singleton < const Init >
7.5	qpp::in	ternal::IOManipEigen Class Reference
	7.5.1	Constructor & Destructor Documentation

CONTENTS

		7.5.1.1 IOManipEigen	86
		7.5.1.2 IOManipEigen	87
	7.5.2	Friends And Related Function Documentation	87
		7.5.2.1 operator <<	87
	7.5.3	Member Data Documentation	87
		7.5.3.1 _A	87
		7.5.3.2 _chop	87
7.6	qpp::in	ternal::IOManipPointer< PointerType > Class Template Reference	87
	7.6.1	Constructor & Destructor Documentation	88
		7.6.1.1 IOManipPointer	88
		7.6.1.2 IOManipPointer	88
	7.6.2	Member Function Documentation	88
		7.6.2.1 operator=	88
	7.6.3	Friends And Related Function Documentation	88
		7.6.3.1 operator <<	88
	7.6.4	Member Data Documentation	88
		7.6.4.1 _end	88
		7.6.4.2 _n	88
		7.6.4.3 _p	88
		7.6.4.4 _separator	88
		7.6.4.5 _start	88
7.7	qpp::in	ternal::IOManipRange< InputIterator > Class Template Reference	89
	7.7.1	Constructor & Destructor Documentation	89
		7.7.1.1 IOManipRange	89
	7.7.2	Friends And Related Function Documentation	90
		7.7.2.1 operator <<	90
	7.7.3	Member Data Documentation	90
		7.7.3.1 _end	90
		7.7.3.2 _first	90
		7.7.3.3 _last	90
		7.7.3.4 _separator	90
		7.7.3.5 _start	90
7.8	qpp::R	andomDevices Class Reference	90
	7.8.1	Detailed Description	91
	7.8.2	Constructor & Destructor Documentation	91
		7.8.2.1 RandomDevices	91
	7.8.3	Friends And Related Function Documentation	92
		7.8.3.1 internal::Singleton < RandomDevices >	92
	7.8.4	Member Data Documentation	92
		7.8.4.1 _rd	92

CONTENTS xi

		7.8.4.2 _rng	 . !	92
7.9	qpp::int	ernal::Singleton< T > Class Template Reference	 . !	92
	7.9.1	Detailed Description	 . !	92
	7.9.2	Constructor & Destructor Documentation	 . !	93
		7.9.2.1 Singleton	 . !	93
		7.9.2.2 ~Singleton	 . !	93
		7.9.2.3 Singleton	 . !	93
	7.9.3	Member Function Documentation	 . !	93
		7.9.3.1 get_instance	 . !	93
		7.9.3.2 operator=	 . !	93
7.10	qpp::St	ates Class Reference	 . !	93
	7.10.1	Detailed Description	 . !	95
	7.10.2	Constructor & Destructor Documentation	 . 9	95
		7.10.2.1 States	 . !	95
	7.10.3	Friends And Related Function Documentation	 . !	95
		7.10.3.1 internal::Singleton < const States >	 . !	95
	7.10.4	Member Data Documentation	 . !	95
		7.10.4.1 b00	 . !	95
		7.10.4.2 b01	 . !	95
		7.10.4.3 b10	 . !	96
		7.10.4.4 b11	 . !	96
		7.10.4.5 GHZ	 . !	96
		7.10.4.6 pb00	 . !	96
		7.10.4.7 pb01	 . !	96
		7.10.4.8 pb10	 . !	96
		7.10.4.9 pb11	 . !	96
		7.10.4.10 pGHZ		96
		7.10.4.11 pW		96
		7.10.4.12 px0		96
		7.10.4.13 px1		96
		7.10.4.14 py0		96
		7.10.4.15 py1		97
		7.10.4.16 pz0		97
		7.10.4.17 pz1		97
		7.10.4.18 W		97
		7.10.4.19 x0		97
		7.10.4.20 x1		97
		7.10.4.21 y0		97
		7.10.4.22 y1		97
		7.10.4.23 z0	 . !	97

xii CONTENTS

		7.10.4.24 z1	7
	7.11	qpp::Timer Class Reference	7
		7.11.1 Detailed Description	8
		7.11.2 Constructor & Destructor Documentation	8
		7.11.2.1 Timer	8
		7.11.3 Member Function Documentation	8
		7.11.3.1 seconds	8
		7.11.3.2 tic	8
		7.11.3.3 toc	9
		7.11.4 Friends And Related Function Documentation	9
		7.11.4.1 operator<<	9
		7.11.5 Member Data Documentation	9
		7.11.5.1 _end	9
		7.11.5.2 _start	9
•	Tile I	Decumentation 10	
8		Documentation 10	
	8.1	classes/codes.h File Reference	
	0.0	8.1.1 Detailed Description	
	8.2	classes/exception.h File Reference	
	0.0		
	8.3	classes/gates.h File Reference	
	8.4	8.3.1 Detailed Description 10. classes/init.h File Reference 10.	
	0.4		
	0.5	8.4.1 Detailed Description	
	8.5	-	
	0.6		
	8.6	classes/states.h File Reference	
	8.7	classes/timer.h File Reference	
	0.7	8.7.1 Detailed Description	
	8.8	constants.h File Reference	
	0.0	8.8.1 Detailed Description	
	8.9	entanglement.h File Reference	
	0.5	8.9.1 Detailed Description	
	8 1N	entropies.h File Reference	
	0.10	8.10.1 Detailed Description	
	g 11	experimental/test.h File Reference	
	0.11	8.11.1 Detailed Description	
	8 12	functions.h File Reference	
	0.12	8.12.1 Detailed Description	
		- on En - Dotallou Doubliption	J

CONTENTS xiii

8.13	input_output.h File Reference	113
	8.13.1 Detailed Description	114
8.14	instruments.h File Reference	114
	8.14.1 Detailed Description	116
8.15	internal/classes/iomanip.h File Reference	116
	8.15.1 Detailed Description	117
8.16	internal/classes/singleton.h File Reference	117
	8.16.1 Detailed Description	117
8.17	internal/util.h File Reference	117
	8.17.1 Detailed Description	119
8.18	MATLAB/matlab.h File Reference	119
	8.18.1 Detailed Description	120
8.19	number_theory.h File Reference	120
	8.19.1 Detailed Description	121
8.20	operations.h File Reference	121
	8.20.1 Detailed Description	123
8.21	qpp.h File Reference	123
	8.21.1 Detailed Description	124
8.22	random.h File Reference	124
	8.22.1 Detailed Description	125
8.23	types.h File Reference	126
	8.23.1 Detailed Description	126
Index		127
HIUCX		141

Chapter 1

Quantum++

Development version, use it at your own risk!

For the latest stable version, switch to the master branch.

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 $http \leftarrow ://www.gnu.org/licenses/$.

Building instructions

Configuration:

- Compiler: g++ version 4.8 or later (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

2 Quantum++

Building without a build system

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

```
* I recommend running via a script, as otherwise setting the 
'DYLD_LIBRARY_PATH' globally may interfere with 
[macports] (https://www.macports.org/)' [cmake] (http://www.cmake.org/) 
installation (in case you use [cmake] (http://www.cmake.org/) from 
[macports] (https://www.macports.org/)). 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

export DYLD_LIBRARY_PATH=$DYLD_LIBRARY_PATH:"/Applications/MATLAB_R2014b.app/bin/maci64"
./qpp
```

• 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.

Quantum++

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

qpp	
Quantum++ main namespace	. 13
qpp::experimental	
Experimental/test functions/classes, do not use or modify	. 69
qpp::internal	
Internal utility functions, do not use/modify	. 69

6 Namespace Index

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

staexception	
qpp::Exception	75
qpp::internal::IOManipEigen	86
qpp::internal::IOManipPointer< PointerType >	87
qpp::internal::IOManipRange < InputIterator >	89
qpp::internal::Singleton< T >	92
qpp::internal::Singleton < const Codes >	92
qpp::Codes	73
qpp::internal::Singleton< const Gates >	92
qpp::Gates	79
$\label{lem:qpp::internal::Singleton} $$\operatorname{qpp}::\operatorname{internal}::\operatorname{Singleton}<\operatorname{const}\operatorname{Init}>\dots$	92
qpp::Init	85
qpp::internal::Singleton < const States >	92
qpp::States	93
qpp::internal::Singleton< RandomDevices >	92
qpp::RandomDevices	90
qpp::Timer	97

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	73
qpp::Exception	
Generates custom exceptions, used when validating function parameters	75
qpp::Gates	
Const Singleton class that implements most commonly used gates	79
qpp::Init	
Const Singleton class that performs additional initializations/cleanups	85
qpp::internal::IOManipEigen	86
qpp::internal::IOManipPointer< PointerType >	87
qpp::internal::IOManipRange < InputIterator >	89
qpp::RandomDevices	
Singeleton class that manages the source of randomness in the library	90
qpp::internal::Singleton< T >	
Singleton policy class, used internally to implement the singleton pattern via CRTP (Curiously	
recurring template pattern)	92
qpp::States	
Const Singleton class that implements most commonly used states	93
qpp::Timer	
Measures time	97

10 Class Index

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

constants.h	
Constants	 106
entanglement.h	
Entanglement functions	 107
entropies.h	
Entropy functions	 108
functions.h	
Generic quantum computing functions	 109
input_output.h	
Input/output functions	 113
instruments.h	
Measurement functions	 114
number_theory.h	
Number theory functions	 120
operations.h	
Quantum operation functions	 121
qpp.h	
Quantum++ main header file, includes all other necessary headers	 123
random.h	
Randomness-related functions	 124
types.h	
Type aliases	 126
classes/codes.h	
Quantum error correcting codes	 101
classes/exception.h	
Exceptions	 101
classes/gates.h	
Quantum gates	 102
classes/init.h	
Initialization	 103
classes/random_devices.h	
Random devices	 104
classes/states.h	
Quantum states	 104
classes/timer.h	
Timing	 105
experimental/test.h	
Experimental/test functions/classes	 109

12 File Index

internal/util.h								
Internal utility functions	 	 	 	 				117
internal/classes/iomanip.h								
Input/output manipulators	 	 	 	 				116
internal/classes/singleton.h								
Singleton pattern via CRTP	 	 	 	 				117
MATLAB/matlab.h								
Input/output interfacing with MATLAB	 	 	 	 		 		119

Chapter 6

Namespace Documentation

6.1 qpp Namespace Reference

Quantum++ main namespace.

Namespaces

· experimental

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

interna

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 idx = std::size_t
```

Non-negative integer index.

using cplx = std::complex < double >

Complex number in double precision.

• using ket = Eigen::VectorXcd

Complex (double precision) dynamic Eigen column vector. using bra = Eigen::RowVectorXcd Complex (double precision) dynamic Eigen row vector. • using cmat = Eigen::MatrixXcd Complex (double precision) dynamic Eigen matrix. using dmat = Eigen::MatrixXd Real (double precision) dynamic Eigen matrix. 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. **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 > schmidtcoeffs (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims) Schmidt coefficients of the bi-partite pure state A. $\bullet \ \ \text{template}{<} \text{typename Derived} >$ cmat schmidtA (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims) Schmidt basis on Alice side. • template<typename Derived > cmat schmidtB (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims) Schmidt basis on Bob side. template<typename Derived > std::vector< double > schmidtprobs (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 >

```
Logarithmic negativity of the bi-partite mixed state A.
```

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

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

Negativity of the bi-partite mixed state A.

template<typename Derived >

```
• 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)
     Adioint.
• 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_col_vect
  < 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, double p)
     Schatten matrix norm.
• template<typename OutputScalar , typename Derived >
  dyn mat< OutputScalar > cwise (const Eigen::MatrixBase< Derived > &A, OutputScalar(*f)(const type-
  name 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.
 \bullet \ \ \mathsf{template} \mathord{<} \mathsf{typename} \ \mathsf{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.

    template<typename Derived >

  dyn_mat< typename Derived::Scalar > kronpow (const Eigen::MatrixBase< Derived > &A, idx n)
     Kronecker power.
• template<typename T >
  dyn_mat< typename T::Scalar > dirsum (const T &head)
     Direct sum.
template<typename T, typename... Args>
  dyn_mat< typename T::Scalar > dirsum (const T &head, const Args &...tail)
     Direct sum.

    template<typename Derived >

  dyn_mat< typename Derived::Scalar > dirsum (const std::vector< Derived > &As)
     Direct sum.

    template<typename Derived >

  dyn_mat< typename Derived::Scalar > dirsum (const std::initializer_list< Derived > &As)
     Direct sum.

    template<typename Derived >
```

dyn_mat< typename Derived::Scalar > dirsumpow (const Eigen::MatrixBase< Derived > &A, idx n)

dyn_mat< typename Derived::Scalar > reshape (const Eigen::MatrixBase< Derived > &A, idx rows, idx

Direct sum power.
• template<typename Derived >

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
  Derived1::Scalar > anticomm (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< De-
  rived2 > &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)
     Gram-Schmidt orthogonalization.

    template<typename Derived >

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

    template<typename Derived >

  internal::IOManipEigen disp (const Eigen::MatrixBase< Derived > &A, double chop=qpp::chop)
     Eigen expression ostream manipulator.
```

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

Complex number ostream manipulator.

• template<typename InputIterator >

internal::IOManipRange

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

Range ostream manipulator.

• template<typename Container >

internal::IOManipRange

< typename

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

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

• template<typename PointerType >

internal::IOManipPointer

< PointerType > disp (const PointerType *p, 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

< 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.

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

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)

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

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 \leftarrow ::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.

• template<typename Derived >

Derived loadMATLABmatrix (const std::string &mat_file, const std::string &var_name)

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

template<>

dmat loadMATLABmatrix (const std::string &mat_file, const std::string &var_name)

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

template<>

cmat loadMATLABmatrix (const std::string &mat file, const std::string &var name)

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

template<typename Derived >

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

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

• template<>

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

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

template<>

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

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

std::vector< 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

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

Derived1::Scalar > 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 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 kraus2super (const std::vector< cmat > &Ks)

Superoperator matrix.

cmat kraus2choi (const std::vector< cmat > &Ks)

Choi matrix.

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

Orthogonal Kraus operators from Choi matrix.

cmat choi2super (const cmat &A)

Converts Choi matrix to superoperator matrix.

cmat super2choi (const cmat &A)

Converts superoperator matrix to Choi matrix.

template<typename Derived >

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

Partial trace.

template<typename Derived >

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

Partial trace.

template<typename Derived >

 $\label{localized-dyn-mat} $$ \down_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 >

Partial transpose.

• template<typename Derived >

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

Subsystem permutation.

• template<typename Derived >

Subsystem 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].

• template<typename 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 Eigen::RowVectorXcd

Complex (double precision) dynamic Eigen row vector.

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

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 Eigen::MatrixXd

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 Eigen::VectorXcd

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.

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 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.

Parameters

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

Α	Eigen expression

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.

See also

qpp::comm()

Anti-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

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 < 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.

Note

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

Parameters

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

Returns

Gate A applied to the part subsys of state

6.1.3.7 template < typename Derived1 , typename Derived2 > $dyn_mat < typename Derived1::Scalar > dpp::apply (const Eigen::MatrixBase < Derived2 > & A, const std::vector < idx > & subsys, idx d = 2)$

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*.

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.

Parameters

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

Returns

Output density matrix after the action of the channel

6.1.3.10 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.

See also

qpp::Gates::CTRL()

Note

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

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.

See also

qpp::Gates::CTRL()

Note

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

Parameters

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

Returns

CTRL-A gate applied to the part subsys of state

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

Orthogonal Kraus operators from Choi matrix.

See also

qpp::kraus2choi()

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

Note

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

Α	Choi matrix
---	-------------

Returns

Set of orthogonal Kraus operators

6.1.3.14 cmat qpp::choi2super (const cmat & A)

Converts Choi matrix to superoperator matrix.

See also

qpp::super2choi()

Parameters

Α	Choi matrix
---	-------------

Returns

Superoperator matrix

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

Commutator.

See also

qpp::anticomm()

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

Parameters

Α	Eigen expression
В	Eigen expression

Returns

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

6.1.3.16 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 $perm \circ sigma = perm(sigma)$

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.

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

Α	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.

See also

qpp::x2contfrac()

Parameters

cf	Integer vector containing the simple continued fraction expansion
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.

See also

qpp::x2contfrac()

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.

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

Returns

Matrix cosine of A

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

Functor.

Parameters

	Α	Eigen expression
Ì	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.

Parameters

Α	Eigen expression

Returns

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

6.1.3.24 template < typename T > dyn_mat < typename T::Scalar > qpp::dirsum (const T & head)

Direct sum.

See also

qpp::dirsumpow()

Used to stop the recursion for the variadic template version of qpp::dirsum()

Parameters

head	Eigen expression

Returns

Its argument head

6.1.3.25 template<typename T , typename... Args> dyn_mat<typename T::Scalar> qpp::dirsum (const T & head, const Args &... tail)

Direct sum.

See also

qpp::dirsumpow()

Parameters

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

Returns

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

6.1.3.26 template<typename Derived > dyn_mat<typename Derived::Scalar> qpp::dirsum (const std::vector< Derived > & As)

Direct sum.

See also

qpp::dirsumpow()

Parameters

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

Returns

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

6.1.3.27 template < typename Derived > dyn_mat < typename Derived::Scalar > qpp::dirsum (const std::initializer_list < Derived > & As)

Direct sum.

See also

qpp::dirsumpow()

Parameters

As	std::initializer list of Eigen expressions, such as {A1, A2,, Ak}

Returns

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

6.1.3.28 template<typename Derived > dyn_mat<typename Derived::Scalar> qpp::dirsumpow (const Eigen::MatrixBase< Derived > & A, idx n)

Direct sum power.

See also

qpp::dirsum()

Parameters

Α	Eigen expression
n	Non-negative integer

Returns

Direct sum of A with itself n times $A^{\oplus n}$, as a dynamic matrix over the same scalar field as A

6.1.3.29 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

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

Complex number ostream manipulator.

Parameters

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

Returns

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

6.1.3.31 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::IOManipRange

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

Parameters

С	Container
separator	Separator
start	Left marking
end	Right marking

Returns

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

6.1.3.33 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

р	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

Full eigen decomposition.

See also

qpp::heig()

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

6.1.3.35 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()

Parameters

Α	Eigen expression
dims	Dimensions of the bi-partite system

Returns

Entanglement, with the logarithm in base 2

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

von-Neumann entropy of the density matrix A

Parameters

Α	Eigen expression

Returns

von-Neumann entropy, with the logarithm in base 2

6.1.3.37 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

6.1.3.38 template<typename Derived > dyn_col_vect<cplx> qpp::evals (const Eigen::MatrixBase< Derived > & A)

Eigenvalues.

See also

qpp::hevals()

Parameters

A Eigen expression

Returns

Eigenvalues of A, as a complex dynamic column vector

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

Eigenvectors.

See also

qpp::hevects()

Parameters

A Eigen expression

Returns

Eigenvectors of A, as columns of a complex dynamic matrix

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

Matrix exponential.

Parameters

A Eigen expression

Returns

Matrix exponential of A

6.1.3.41 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)

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

Greatest common divisor of two non-negative integers.

See also

qpp::lcm()

Parameters

m	Non-negative integer
n	Non-negative integer

Returns

Greatest common divisor of m and n

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

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

See also

qpp::lcm()

Parameters

ns	List of non-negative integers

Returns

Greatest common divisor of all numbers in ns

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

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

Note

Both local dimensions must be equal

Uses qpp::logdet() to avoid overflows

See also

qpp::logdet()

Parameters

Α	Eigen expression

Returns

G-concurrence

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

Gram-Schmidt orthogonalization.

Vs	std::vector of Eigen exp	oressions as column vectors

Returns

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

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

Gram-Schmidt orthogonalization.

Parameters

Vs	std::initializer_list of Eigen expressions as column vectors

Returns

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

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

Gram-Schmidt orthogonalization.

Parameters

Α	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

6.1.3.48 template<typename Derived > std::pair<dyn_col_vect<double>, cmat> qpp::heig (const Eigen::MatrixBase< Derived > & A)

Full eigen decomposition of Hermitian expression.

See also

qpp::eig()

Parameters

Α	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

 $6.1.3.49 \quad template < typename \ Derived > dyn_col_vect < double > qpp::hevals \ (\ const \ Eigen::MatrixBase < Derived > \& \ A \)$

Hermitian eigenvalues.

See also

qpp::evals()

Parameters

Α	Eigen expression

Returns

Eigenvalues of Hermitian A, as a real dynamic column vector

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

Hermitian eigenvectors.

See also

qpp::evects()

Parameters

Α	Eigen expression

Returns

Eigenvectors of Hermitian A, as columns of a complex matrix

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

Inverse.

Parameters

Α	Eigen expression

Returns

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

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

Inverse permutation.

Parameters

perm	Permutation

Returns

Inverse of the permutation perm

6.1.3.53 cmat qpp::kraus2choi (const std::vector < cmat > & Ks)

Choi matrix.

See also

qpp::choi2kraus()

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

6.1.3.54 cmat qpp::kraus2super (const std::vector < cmat > & Ks)

Superoperator matrix.

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

6.1.3.55 template < typename T > dyn_mat < typename T::Scalar > qpp::kron (const T & head)

Kronecker product.

See also

qpp::kronpow()

Used to stop the recursion for the variadic template version of qpp::kron()

Parameters

head	Eigen expression

Returns

Its argument head

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

Kronecker product.

See also

qpp::kronpow()

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

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

Kronecker product.

See also

qpp::kronpow()

Parameters

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

Returns

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

Kronecker product.

See also

qpp::kronpow()

Parameters

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

Returns

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

6.1.3.59 template<typename Derived > dyn_mat<typename Derived::Scalar> qpp::kronpow (const Eigen::MatrixBase< Derived > & A, idx n)

Kronecker power.

See also

qpp::kron()

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

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

Least common multiple of two positive integers.

See also

qpp::gcd()

Parameters

m	Positive integer
n	Positive integer

Returns

Least common multiple of m and n

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

Least common multiple of a list of positive integers.

See also

qpp::gcd()

Parameters

ns	List of positive integers

Returns

Least common multiple of all numbers in ns

6.1.3.62 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.

See also

```
qpp::save()
```

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");
```

Parameters

Α	Eigen expression
fname	Output file name

6.1.3.63 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.

See also

```
qpp::saveMATLABmatrix()
```

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)

```
6.1.3.64 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)

See also

```
qpp::saveMATLABmatrix()
```

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)

6.1.3.65 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)

See also

qpp::saveMATLABmatrix()

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

Example:

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

Parameters

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

Returns

Eigen complex dynamic matrix (qpp::cmat)

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

Logarithm of the determinant.

Useful when the determinant overflows/underflows

Parameters

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

Returns

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

 $6.1.3.67 \quad template < typename \ Derived > cmat \ qpp::logm \ (\ const \ Eigen::MatrixBase < Derived > \& \ \textit{A} \)$

Matrix logarithm.

Parameters

Α	Eigen expression

Returns

Matrix logarithm of A

6.1.3.68 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.

Α	Eigen expression
dims	Dimensions of the bi-partite system

Returns

Logarithmic negativity, with the logarithm in base 2

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::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.

Parameters

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

Returns

Tuple consisiting of 1. Result of the measurement, 2. 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 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

6.1.3.71 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
- 6.1.3.72 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
- 6.1.3.73 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*.

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
- 6.1.3.74 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*.

Parameters

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

Returns

Tuple consisiting of 1. Result of the measurement,

- 1. Vector of outcome probabilities and 3. Vector of post-measurement normalized states
- 6.1.3.75 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
- 6.1.3.76 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.

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

6.1.3.77 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

6.1.3.78 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

6.1.3.79 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

6.1.3.80 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

6.1.3.81 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.

Parameters

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

Returns

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

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

Multi-index to non-negative integer index.

See also

qpp::n2multiidx()

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

Parameters

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

Returns

Non-negative integer index

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

Non-negative integer index to multi-index.

See also

qpp::multiidx2n()

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

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

Returns

Multi-index of the same size as dims

6.1.3.84 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

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

Frobenius norm.

Parameters

Α	Eigen expression

Returns

Frobenius norm of A

6.1.3.86 cplx qpp::omega (idx D) [inline]

D-th root of unity.

Parameters

D	Non-negative integer

Returns

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

6.1.3.87 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>
```

6.1.3.88 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>
```

6.1.3.89 template < typename Derived > $dyn_mat < typename Derived::Scalar > qpp::powm (const Eigen::MatrixBase < Derived > & A, idx n)$

Matrix power.

See also

qpp::spectralpowm()

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

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

Projector.

Normalized projector onto state vector

Parameters

V Eigen expression

Returns

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

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

Element-wise product of A.

Parameters

Α	Eigen expression

Returns

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

6.1.3.92 template < typename inputiterator > inputiterator::value_type qpp::prod (inputiterator first, inputiterator last)

Element-wise product of a range.

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

Returns

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

Partial trace.

See also

```
qpp::ptrace1(), qpp::ptrace2()
```

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

Parameters

	Α	Eigen expression
Ī	subsys	Subsystem indexes
Ī	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

6.1.3.94 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.

See also

```
qpp::ptrace1(), qpp::ptrace2()
```

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

Parameters

Α	Eigen expression
subsys	Subsystem indexes
d	Subsystem dimensions

Returns

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

Partial trace.

See also

qpp::ptrace2()

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.

See also

qpp::ptrace1()

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

6.1.3.97 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

Parameters

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

Returns

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

6.1.3.98 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

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

6.1.3.99 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

6.1.3.100 template < 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.

Parameters

A	Eigen expression
subsysA	Indexes of the first subsystem
subsysB	Indexes of the second subsystem
d	Subsystem dimensions

Returns

Mutual information between the 2 subsystems

6.1.3.101 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

6.1.3.102 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

```
6.1.3.103 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);
```

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 complex matrix

```
6.1.3.104 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)

```
6.1.3.105 cmat qpp::randH ( idx D )
```

Generates a random Hermitian matrix.

-		
ſ	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]

6.1.3.107 ket qpp::randket (idx D)

Generates a random normalized ket (pure state vector)

Parameters

Δ.	Dimonsion of the Hilbert space
U	Dimension of the Hilbert space

Returns

Random normalized ket

6.1.3.108 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

6.1.3.109 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

6.1.3.110 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

```
6.1.3.111 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

```
6.1.3.112 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)

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

Generates a random uniformly distributed permutation.

Uses Knuth 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*

6.1.3.114 cmat qpp::randrho (idx D)

Generates a random density matrix.

Parameters

D Dimension of the Hilbert space

Returns

Random density matrix

6.1.3.115 cmat qpp::randU (idx D)

Generates a random unitary matrix.

Parameters

D Dimension of the Hilbert space

Returns

Random unitary

6.1.3.116 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

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

Renyi- α 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 $lpha=\infty$

Returns

Renyi- α entropy, with the logarithm in base 2

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

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

Note

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

Parameters

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

Returns

Renyi- α entropy, with the logarithm in base 2

6.1.3.119 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

6.1.3.120 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

6.1.3.121 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::load()

Parameters

Α	Eigen expression
fname	Output file name

6.1.3.122 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.

See also

qpp::loadMATLABmatrix()

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)

6.1.3.123 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)

See also

qpp::loadMATLABmatrix()

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
ĺ	mode	Saving mode (append, overwrite etc.), see MATLAB matOpen() documentation for details

6.1.3.124 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)

See also

qpp::loadMATLABmatrix()

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 matOpen() documentation for details

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

Schatten matrix norm.

Parameters

Α	Eigen expression
р	Real number, greater or equal to 1, use qpp::infty for $p = \infty$

Returns

Schatten-p matrix norm of A

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

Schmidt basis on Alice side.

Parameters

Α	Eigen expression
dims	Dimensions of the bi-partite system

Returns

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

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

Schmidt basis on Bob 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 side.

6.1.3.128 template<typename Derived > dyn_col_vect<double> qpp::schmidtcoeffs (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::schmidtprobs()

Parameters

Α	Eigen expression
dims	Dimensions of the bi-partite system

Returns

Schmidt coefficients of A, as a real dynamic column vector

6.1.3.129 template < typename Derived > std::vector < double > qpp::schmidtprobs (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::schmidtcoeffs()

Α	Eigen expression
dims	Dimensions of the bi-partite system

Returns

Real vector consisting of the Schmidt probabilites of A

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

Matrix sin.

Parameters

Α	Eigen expression

Returns

Matrix sine of A

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

Matrix power.

See also

qpp::powm()

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

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

Matrix square root.

Parameters

Α	Eigen expression

Returns

Matrix square root of A

6.1.3.133 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

6.1.3.134 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

6.1.3.135 cmat qpp::super2choi (const cmat & A)

Converts superoperator matrix to Choi matrix.

See also

qpp::choi2super()

Parameters

Α	Superoperator matrix

Returns

Choi matrix

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

Singular values.

Parameters

Α	Eigen expression

Returns

Singular values of A, ordered in decreasing order, as a real dynamic column vector

6.1.3.137 template < typename Derived > std::tuple < cmat, dyn_col_vect < 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

6.1.3.138 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

6.1.3.139 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

6.1.3.140 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)

Subsystem 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

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

Subsystem 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

6.1.3.142 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

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

Transpose.

Parameters

Α	Eigen expression
	g

Returns

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

6.1.3.144 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

6.1.3.145 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

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

Simple continued fraction expansion.

See also

qpp::contfrac2x()

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, see the class qpp::Init

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

 π

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

- void _n2multiidx (idx n, idx numdims, const idx *dims, idx *result)
- idx _multiidx2n (const idx *midx, idx numdims, const idx *dims)
- template<typename Derived >

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

template<typename Derived >

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

• template<typename Derived >

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

• template<typename Derived >

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

• template<typename T >

bool check nonzero size (const T &x)

- bool check dims (const std::vector < idx > &dims)
- template<typename 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 <u>_check_dims_match_rvect</u> (const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &V)

- bool <u>_check_eq_dims</u> (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 Derived1 , typename Derived2 >

dyn mat< typename

 $\label{lem:decomposition} Derived1::Scalar > _dirsum2 \; (const \; Eigen::MatrixBase < \; Derived1 > \&A, \; const \; Eigen::MatrixBase < \; Derived2 > \&B)$

• template<typename T >

void variadic_vector_emplace (std::vector< T > &)

template<typename T, typename First, typename... Args>
 void variadic_vector_emplace (std::vector< T > &v, First &&first, Args &&...args)

6.3.1 Detailed Description

Internal 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)
- $\hbox{6.3.2.3 template$<$typename Derived>$bool qpp::internal::_check_dims_match_cvect(const std::vector< idx > \& \textit{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) 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::_dirsum2 (const Eigen::MatrixBase< Derived1 > & A, const Eigen::MatrixBase< Derived2 > & B) 6.3.2.14 template < typename Derived1 , typename Derived2 > dyn_mat < typename Derived1::Scalar > qpp::internal::_kron2 (const Eigen::MatrixBase< Derived1 > & A, const Eigen::MatrixBase< Derived2 > & B) 6.3.2.15 idx qpp::internal::_multiidx2n (const idx * midx, idx numdims, const idx * dims) [inline] 6.3.2.16 void qpp::internal::_n2multiidx (idx n, idx numdims, const idx * dims, idx * result) [inline] 6.3.2.17 template<typename T > void qpp::internal::variadic_vector_emplace (std::vector< T > &)

6.3.2.18 template < typename T , typename First , typename... Args > void qpp::internal::variadic_vector_emplace (

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

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 type.

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.

See also

qpp::Codes::Type

Parameters

type	Code type
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:

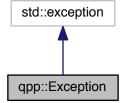
• classes/codes.h

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,

Type::SUBSYS_MISMATCH_DIMS, Type::PERM_INVALID, Type::PERM_MISMATCH_DIMS, Type::NOT ← QUBIT_GATE,

Type::NOT_QUBIT_SUBSYS, Type::NOT_BIPARTITE, Type::NO_CODEWORD, Type::OUT_OF_RANGE, Type::TYPE_MISMATCH, Type::UNDEFINED_TYPE, Type::CUSTOM_EXCEPTION }

Exception types, add more 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 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 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> of dimensions has zero size or contains zeros

DIMS_NOT_EQUAL Local/global dimensions are not equal

DIMS_MISMATCH_MATRIX Product of the elements of std::vector<idx> of dimensions is not equal to the number of rows of Eigen::Matrix (assumed to be a square matrix)

DIMS_MISMATCH_CVECTOR Product of the elements of std::vector<idx> of dimensions is not equal to the number of elements of Eigen::Matrix (assumed to be a column vector)

DIMS_MISMATCH_RVECTOR Product of the elements of std::vector<idx> of dimensions is not equal to the number of elements of Eigen::Matrix (assumed to be a row vector)

DIMS_MISMATCH_VECTOR Product of the elements of std::vector<idx> of dimensions is not equal to the number of elements of Eigen::Matrix (assumed to be a row/column vector)

SUBSYS_MISMATCH_DIMS std::vector<idx> of subsystem labels has duplicates, or has entries that are larger than the size of the std::vector<idx> of dimensions

PERM_INVALID std::vector<idx> does note represent a valid permutation

PERM_MISMATCH_DIMS Size of the std::vector<idx> representing the permutation is different from the size of the std::vector<idx> of dimensions

NOT_QUBIT_GATE Eigen::Matrix is not 2 x 2

NOT_QUBIT_SUBSYS Subsystems are not 2-dimensional

NOT_BIPARTITE std::vector<idx> of dimensions has size different from 2

NO_CODEWORD Codeword does not exist, thrown when calling qpp::Codes::codeword() with invalid index i

OUT_OF_RANGE Parameter out of range

TYPE_MISMATCH Scalar types do not match

UNDEFINED_TYPE Templated specialization 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 type, defined in 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 description

7.2.4 Member Function Documentation

7.2.4.1 void qpp::Exception::_construct_exception_msg() [inline], [private]

Constructs the exception 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 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:

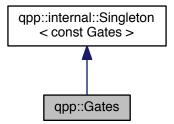
· classes/exception.h

7.3 qpp::Gates Class Reference

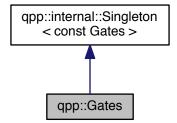
const Singleton class that implements most commonly used gates

#include <classes/gates.h>

Inheritance diagram for qpp::Gates:



Collaboration diagram for qpp::Gates:



Public Member Functions

Toffoli gate.

Fredkin gate.

• cmat FRED {cmat::ldentity(8, 8)}

```
    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.

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

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.

See also

qpp::applyCTRL()

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.

See also

qpp::kron()

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

Parameters

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

Returns

Fourier transform gate for qudits

7.3.3.4 template<typename Derived = Eigen::MatrixXcd> Derived qpp::Gates::Id (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.

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.

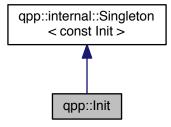
```
7.3.5.3 cmat qpp::Gates::CZ {cmat::Identity(4, 4)}
Controlled-Phase gate.
7.3.5.4 cmat qpp::Gates::FRED {cmat::Identity(8, 8)}
Fredkin gate.
7.3.5.5 cmat qpp::Gates::H {cmat::Zero(2, 2)}
Hadamard gate.
7.3.5.6 cmat qpp::Gates::ld2 {cmat::ldentity(2, 2)}
Identity gate.
7.3.5.7 cmat qpp::Gates::S {cmat::Zero(2, 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

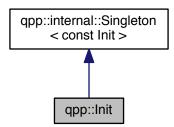
const Singleton class that performs additional initializations/cleanups

#include <classes/init.h>

Inheritance diagram for qpp::Init:



Collaboration diagram for qpp::Init:



Public Member Functions

• Init ()

Additional initializations.

Private Member Functions

• ∼Init ()

Cleanups.

Friends

class internal::Singleton< const Init >

Additional Inherited Members

7.4.1 Detailed Description

const Singleton class that performs additional initializations/cleanups

7.4.2 Constructor & Destructor Documentation

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

Additional initializations.

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

Cleanups.

7.4.3 Friends And Related Function Documentation

```
7.4.3.1 friend class internal::Singleton < const lnit > [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 <u>_chop</u>

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]

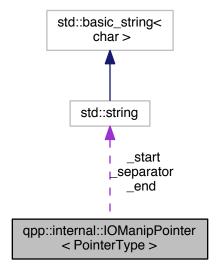
The documentation for this class was generated from the following file:

• 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]

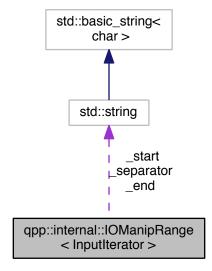
The documentation for this class was generated from the following file:

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 gpp::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.

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. This class also seeds the standard 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::Init, 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>:: \sim 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:

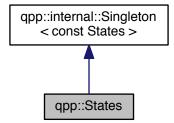
· internal/classes/singleton.h

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 $\mid -><-\mid$.

• 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

7.10.3.1 friend class internal::Singleton < const States > [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)

96 Class Documentation

```
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>

98 Class Documentation

Public Member Functions

• Timer ()

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

• void tic ()

Resets the chronometer.

• const Timer & toc ()

Stops the chronometer.

• double seconds () const

Time passed in seconds.

Protected Attributes

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

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

Friends

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

7.11.1 Detailed Description

Measures time.

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

7.11.2 Constructor & Destructor Documentation

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

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

7.11.3 Member Function Documentation

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

Time passed in seconds.

Returns

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

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

Resets the chronometer.

Resets the starting/ending point to the current time

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

Stops the chronometer.

Set the current time as the ending point

Returns

Current instance

7.11.4 Friends And Related Function Documentation

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

Overload for std::ostream operators.

Parameters

OS	Output stream
rhs	Timer instance

Returns

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

7.11.5 Member Data Documentation

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

7.11.5.2 std::chrono::steady_clock::time_point qpp::Timer::_start [protected]

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

· classes/timer.h

100 **Class Documentation**

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 qpp::chop = 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 qpp::infty = std::numeric_limits<double>::infinity()

Used to denote infinity in double precision.

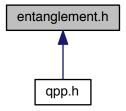
8.8.1 Detailed Description

Constants.

8.9 entanglement.h File Reference

Entanglement functions.

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::schmidtcoeffs (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 side.

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

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

Schmidt basis on Bob side.

• template<typename Derived >

std::vector< double > qpp::schmidtprobs (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.

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

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.

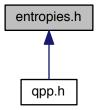
8.9.1 Detailed Description

Entanglement functions.

8.10 entropies.h File Reference

Entropy functions.

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



Namespaces

dbb

Quantum++ main namespace.

Functions

- template<typename Derived >
 double qpp::entropy (const Eigen::MatrixBase< Derived > &A)
- von-Neumann entropy of the density matrix Adouble qpp::entropy (const std::vector< double > &prob)

Shannon entropy of the probability distribution prob.

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

 Renyi- α entropy of the density matrix A, for α ≥ 0.
- double qpp::renyi (const std::vector< double > &prob, double alpha)

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

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

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

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.

template<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.

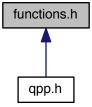
8.11.1 Detailed Description

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.

Functions

```
    template<typename Derived >

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

    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.
• template<typename Derived >
  Derived::Scalar <a href="mailto:qpp::trace">qpp::trace</a> (const Eigen::MatrixBase</a> Derived > &A)
      Trace.

    template<typename Derived >

  Derived::Scalar qpp::det (const Eigen::MatrixBase< Derived > &A)
      Determinant.

    template<typename Derived >

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

    template<typename Derived >

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

    template<typename Derived >

  double <a href="mailto:qpp::norm">qpp::norm</a> (const Eigen::MatrixBase</a> 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 col vect
  < 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, double p)
     Schatten matrix 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.

    template<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 T >
  dyn_mat< typename T::Scalar > qpp::dirsum (const T &head)
     Direct sum.
• template<typename T , typename... Args>
  dyn mat< typename T::Scalar > qpp::dirsum (const T &head, const Args &...tail)
     Direct sum.
• template<typename Derived >
  dyn mat< typename Derived::Scalar > qpp::dirsum (const std::vector< Derived > &As)
     Direct sum.

    template<typename Derived >

  dyn_mat< typename Derived::Scalar > qpp::dirsum (const std::initializer_list< Derived > &As)
     Direct sum.

    template<typename Derived >

  dyn_mat< typename Derived::Scalar > qpp::dirsumpow (const Eigen::MatrixBase< Derived > &A, idx n)
     Direct sum 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<
  Derived 2 > \&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.
```

dyn_mat< typename Derived::Scalar > qpp::grams (const Eigen::MatrixBase< Derived > &A)

template<typename Derived >

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.

template<typename Derived >

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

Computes the absolut values squared of a column vector.

template<typename InputIterator >

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

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.

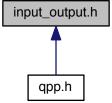
8.12.1 Detailed Description

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.

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.

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.

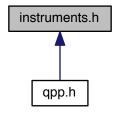
8.13.1 Detailed Description

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.

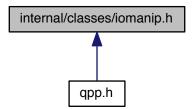
8.14.1 Detailed Description

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.

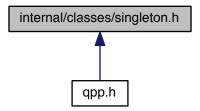
8.15.1 Detailed Description

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.

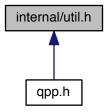
8.16.1 Detailed Description

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)
- ullet template<typename Derived >

bool pp::internal::_check_square_mat (const Eigen::MatrixBase< Derived > &A)

- $\bullet \ \ \text{template}{<} \text{typename Derived} >$
 - $bool\ qpp::internal::_check_vector\ (const\ Eigen::MatrixBase < Derived > \&A)$
- $\bullet \ \ \text{template}{<} \text{typename Derived}>$

bool qpp::internal::_check_row_vector (const Eigen::MatrixBase< Derived > &A)

- $\bullet \ \ {\it template}{<} {\it typename Derived} >$
 - 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

 $\label{lem:decomp} Derived1::Scalar > qpp::internal::_kron2 \ (const\ Eigen::MatrixBase < Derived1 > \&A,\ const\ Eigen::MatrixBase < Derived2 > \&B)$

```
    template<typename Derived1 , typename Derived2 > dyn_mat< typename
        Derived1::Scalar > qpp::internal::_dirsum2 (const Eigen::MatrixBase< Derived1 > &A, const Eigen::Matrix
        Base< Derived2 > &B)
    template<typename T > void qpp::internal::variadic_vector_emplace (std::vector< T > &)
    template<typename T , typename First , typename... Args> void qpp::internal::variadic_vector_emplace (std::vector< T > &v, First &&first, Args &&...args)
```

8.17.1 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

template<typename Derived >
 Derived qpp::loadMATLABmatrix (const std::string &mat_file, const std::string &var_name)
 Loads an Eigen dynamic matrix from a MATLAB .mat file, generic version.

template<>

dmat gpp::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 gpp::loadMATLABmatrix (const std::string &mat file, const std::string &var name)

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

• template<typename Derived >

void qpp::saveMATLABmatrix (const Eigen::MatrixBase< Derived > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)

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

• template<>

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

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

template<>

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

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

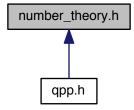
8.18.1 Detailed Description

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 qpp::contfrac2x (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.

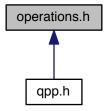
8.19.1 Detailed Description

Number theory functions.

8.20 operations.h File Reference

Quantum operation functions.

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



Namespaces

qpp

Quantum++ main namespace.

Functions

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, 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

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

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

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

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

• template<typename Derived >

cmat 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::kraus2super (const std::vector< cmat > &Ks)

Superoperator matrix.

cmat qpp::kraus2choi (const std::vector< cmat > &Ks)

Choi matrix.

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

Orthogonal Kraus operators from Choi matrix.

cmat qpp::choi2super (const cmat &A)

Converts Choi matrix to superoperator matrix.

cmat qpp::super2choi (const cmat &A)

Converts superoperator matrix to Choi matrix.

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

Partial trace.

template<typename Derived >

 $\label{lem:dyn_mat} $$ dyn_mat< typename Derived::Scalar > qpp::ptrace2 (const Eigen::MatrixBase< Derived > &A, const std $$::vector < idx > &dims)$$

Partial trace.

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.

template<typename Derived >

Partial trace.

• 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.

• template<typename Derived >

 $\label{lem:dyn_mat} $$ dyn_mat< typename Derived::Scalar > qpp::ptranspose (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsys, idx d=2) $$$

Partial transpose.

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)

Subsystem permutation.

• template<typename Derived >

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

Subsystem 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 Init & qpp::init = Init::get_instance()
```

```
qpp::Init const Singleton
```

• const Codes & qpp::codes = Codes::get_instance()

```
qpp::Codes const Singleton
```

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

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

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

```
qpp::States const Singleton
```

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

```
qpp::RandomDevices Singleton
```

8.21.1 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 qpp::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 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.

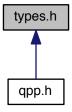
8.22.1 Detailed Description

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::idx = std::size_t
```

Non-negative integer index.

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

Complex number in double precision.

• using qpp::ket = Eigen::VectorXcd

Complex (double precision) dynamic Eigen column vector.

• using qpp::bra = Eigen::RowVectorXcd

Complex (double precision) dynamic Eigen row vector.

• using qpp::cmat = Eigen::MatrixXcd

Complex (double precision) dynamic Eigen matrix.

using qpp::dmat = Eigen::MatrixXd

Real (double precision) dynamic Eigen matrix.

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

```
using \ \ qpp:: dyn\_mat = 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.

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

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

Dynamic Eigen row vector over the field specified by Scalar.

8.23.1 Detailed Description

Type aliases.

Index

absm	qpp::Exception, 77
qpp, 24	DIMS_MISMATCH_VECTOR
abssq	qpp::Exception, 77
qpp, 25	DIMS_NOT_EQUAL
adjoint	qpp::Exception, 77
qpp, 25	det
anticomm	qpp, 32
	dirsum
qpp, 25	
apply	qpp, 32, 33
qpp, 26, 27	dirsumpow
	qpp, 33
bra	disp
qpp, 23	qpp, 34, 35
	dmat
CUSTOM_EXCEPTION	qpp, 24
qpp::Exception, 78	
choi2kraus	ee
qpp, 28	qpp, 68
choi2super	eig
qpp, 29	qpp, 35
chop	entanglement
qpp, 68	
cmat	qpp, 36
	entropy
qpp, 23	qpp, 36
codes	eps
qpp, 68	qpp, 68
comm	evals
qpp, 29	qpp, <mark>36</mark>
compperm	evects
qpp, 29	qpp, 37
concurrence	expm
qpp, 29	qpp, 37
conjugate	
qpp, 31	FIVE QUBIT
contfrac2x	qpp::Codes, 74
qpp, 31	funm
cosm	qpp, 37
qpp, 31	9PP; • 7
cplx	gcd
•	qpp, 37, 38
qpp, 23	
cwise	gconcurrence
qpp, 32	qpp, 38
DIMO INIVALID	grams
DIMS_INVALID	qpp, 38, 39
qpp::Exception, 77	gt
DIMS_MISMATCH_CVECTOR	qpp, 68
qpp::Exception, 77	
DIMS_MISMATCH_MATRIX	heig
qpp::Exception, 77	qpp, 39
DIMS MISMATCH RVECTOR	hevals

128 INDEX

qpp, 39	mprj
hevects	qpp, 49, 50
qpp, 40	multiidx2n
	qpp, 50
idx	0 1
qpp, 24	n2multiidx
infty	qpp, 50
qpp, 68	NINE_QUBIT_SHOR
init	qpp::Codes, 74 NO CODEWORD
qpp, 69 inverse	_
qpp, 40	qpp::Exception, 78 NOT BIPARTITE
invperm	qpp::Exception, 78
qpp, 40	NOT QUBIT GATE
4ρρ , το	qpp::Exception, 77
ket	NOT QUBIT SUBSYS
qpp, 24	qpp::Exception, 77
kraus2choi	negativity
qpp, 40	qpp, 51
kraus2super	norm
qpp, 41	qpp, 51
kron	
qpp, 41, 42	OUT_OF_RANGE
kronpow	qpp::Exception, 78
qpp, 42	omega
	qpp, 5 1
lcm	
qpp, 43	PERM_INVALID
load	qpp::Exception, 77
qpp, 43	PERM_MISMATCH_DIMS
logdet	qpp::Exception, 77
qpp, 45	pi
logm	qpp, 69
qpp, 45 lognegativity	powm qpp, 52
qpp, 45	prj
4ρρ , το	qpp, 52
MATRIX_MISMATCH_SUBSYS	prod
qpp::Exception, 77	qpp, 52
MATRIX_NOT_CVECTOR	ptrace
qpp::Exception, 77	qpp, <u>53</u>
MATRIX_NOT_RVECTOR	ptrace1
qpp::Exception, 77	qpp, 53
MATRIX_NOT_SQUARE	ptrace2
qpp::Exception, 77	qpp, 54
MATRIX_NOT_SQUARE_OR_CVECTOR	ptranspose
qpp::Exception, 77	qpp, 54
MATRIX_NOT_SQUARE_OR_RVECTOR	
qpp::Exception, 77	qmutualinfo
MATRIX_NOT_SQUARE_OR_VECTOR	qpp, 55
qpp::Exception, 77	qpp, 13
MATRIX_NOT_VECTOR	absm, 24
qpp::Exception, 77	abssq, 25
maxn	adjoint, 25
qpp, 69	anticomm, 25
measure	apply, 26, 27
qpp, 46–49 mket	bra, 23 choi2kraus, 28
qpp, 49	choi2super, 29
Abb	0110123apo1, 20

INDEX 129

chan 60	pri FO
chop, 68	prj, 52
cmat, 23	prod, 52
codes, 68	ptrace, 53
comm, 29	ptrace1, 53
compperm, 29	ptrace2, 54
concurrence, 29	ptranspose, 54
conjugate, 31	qmutualinfo, 55
contfrac2x, 31	rand, 55, 56
cosm, 31	randidx, 57
cplx, 23	randket, 57
cwise, 32	randkraus, 57
det, 32	randn, 57, 58
dirsum, 32, 33	randperm, 59
dirsumpow, 33	randrho, 59
disp, 34, 35	rdevs, 69
dmat, 24	renyi, 60
ee, 68	reshape, 60
eig, 35	rho2pure, 61
entanglement, 36	save, 61
entropy, 36	schatten, 62
	schmidtcoeffs, 63
eps, 68	
evals, 36	schmidtprobs, 63
evects, 37	sinm, 64
expm, 37	spectralpowm, 64
funm, 37	sqrtm, 64
gcd, 37, 38	st, 69
gconcurrence, 38	sum, 64, 65
grams, 38, 39	super2choi, 65
gt, 68	svals, 65
heig, 39	svd, 65
hevals, 39	syspermute, 66
hevects, 40	trace, 67
idx, 24	transpose, 67
infty, 68	tsallis, 67
init, 69	x2contfrac, 68
inverse, 40	qpp::Codes
invperm, 40	FIVE_QUBIT, 74
ket, 24	NINE_QUBIT_SHOR, 74
kraus2choi, 40	SEVEN_QUBIT_STEANE, 74
kraus2super, 41	qpp::Exception
kron, 41, 42	CUSTOM_EXCEPTION, 78
kronpow, 42	DIMS_INVALID, 77
lcm, 43	DIMS_MISMATCH_CVECTOR, 77
load, 43	DIMS MISMATCH MATRIX, 77
logdet, 45	DIMS MISMATCH RVECTOR, 77
logm, 45	DIMS_MISMATCH_VECTOR, 77
lognegativity, 45	DIMS_NOT_EQUAL, 77
maxn, 69	MATRIX_MISMATCH_SUBSYS, 77
measure, 46–49	MATRIX_NOT_CVECTOR, 77
mket, 49	MATRIX NOT RVECTOR, 77
mprj, 49, 50	MATRIX_NOT_SQUARE, 77
multiidx2n, 50	MATRIX_NOT_SQUARE_OR_CVECTOR, 77
n2multiidx, 50	MATRIX_NOT_SQUARE_OR_RVECTOR, 77
negativity, 51	MATRIX_NOT_SQUARE_OR_VECTOR, 77
norm, 51	MATRIX_NOT_VECTOR, 77
omega, 51	NO_CODEWORD, 78
pi, 69	NOT_BIPARTITE, 78
powm, 52	NOT_QUBIT_GATE, 77

130 INDEX

NOT_QUBIT_SUBSYS, 77 OUT_OF_RANGE, 78 PERM_INVALID, 77 PERM_MISMATCH_DIMS, 77 SUBSYS_MISMATCH_DIMS, 77 TYPE_MISMATCH, 78 UNDEFINED_TYPE, 78 UNKNOWN_EXCEPTION, 77 ZERO_SIZE, 77	svd qpp, 65 syspermute qpp, 66 TYPE_MISMATCH qpp::Exception, 78 trace qpp, 67 transpose
rand qpp, 55, 56 randidx	qpp, 67 tsallis qpp, 67
qpp, 57 randket qpp, 57 randkraus qpp, 57	UNDEFINED_TYPE qpp::Exception, 78 UNKNOWN_EXCEPTION qpp::Exception, 77
randn qpp, 57, 58 randperm	x2contfrac qpp, 68
qpp, 59 randrho qpp, 59 rdevs	ZERO_SIZE qpp::Exception, 77
qpp, 69 renyi qpp, 60 reshape	
qpp, 60 rho2pure qpp, 61	
SEVEN_QUBIT_STEANE qpp::Codes, 74 SUBSYS_MISMATCH_DIMS qpp::Exception, 77	
save qpp, 61 schatten qpp, 62 schmidtcoeffs	
qpp, 63 schmidtprobs qpp, 63 sinm	
qpp, 64 spectralpowm qpp, 64 sqrtm	
qpp, 64 st qpp, 69	
sum qpp, 64, 65 super2choi	
qpp, 65 svals	
qpp, 65	