

quantum++

0.1

Generated by Doxygen 1.8.7

Sun Nov 30 2014 17:55:26



# Contents

<b>1</b>	<b>Quantum++ - A C++11 quantum computing library</b>	<b>1</b>
<b>2</b>	<b>Namespace Index</b>	<b>3</b>
2.1	Namespace List . . . . .	3
<b>3</b>	<b>Hierarchical Index</b>	<b>5</b>
3.1	Class Hierarchy . . . . .	5
<b>4</b>	<b>Class Index</b>	<b>7</b>
4.1	Class List . . . . .	7
<b>5</b>	<b>File Index</b>	<b>9</b>
5.1	File List . . . . .	9
<b>6</b>	<b>Namespace Documentation</b>	<b>11</b>
6.1	qpp Namespace Reference . . . . .	11
6.1.1	Typedef Documentation . . . . .	21
6.1.1.1	bra . . . . .	21
6.1.1.2	cmat . . . . .	21
6.1.1.3	cplx . . . . .	21
6.1.1.4	dmat . . . . .	21
6.1.1.5	dyn_col_vect . . . . .	21
6.1.1.6	dyn_mat . . . . .	21
6.1.1.7	dyn_row_vect . . . . .	21
6.1.1.8	idx . . . . .	22
6.1.1.9	ket . . . . .	22
6.1.2	Function Documentation . . . . .	22
6.1.2.1	absm . . . . .	22
6.1.2.2	abssq . . . . .	22
6.1.2.3	abssq . . . . .	22
6.1.2.4	adjoint . . . . .	22
6.1.2.5	anticomm . . . . .	23
6.1.2.6	apply . . . . .	23

6.1.2.7	<code>apply</code>	23
6.1.2.8	<code>applyCTRL</code>	24
6.1.2.9	<code>applyCTRL</code>	24
6.1.2.10	<code>channel</code>	25
6.1.2.11	<code>channel</code>	26
6.1.2.12	<code>channel</code>	26
6.1.2.13	<code>choi</code>	26
6.1.2.14	<code>choi2kraus</code>	27
6.1.2.15	<code>comm</code>	27
6.1.2.16	<code>compperm</code>	27
6.1.2.17	<code>concurrence</code>	27
6.1.2.18	<code>conjugate</code>	28
6.1.2.19	<code>contfrac2x</code>	28
6.1.2.20	<code>contfrac2x</code>	28
6.1.2.21	<code>cosm</code>	28
6.1.2.22	<code>cwise</code>	29
6.1.2.23	<code>det</code>	29
6.1.2.24	<code>disp</code>	29
6.1.2.25	<code>disp</code>	29
6.1.2.26	<code>disp</code>	30
6.1.2.27	<code>disp</code>	30
6.1.2.28	<code>disp</code>	30
6.1.2.29	<code>eig</code>	31
6.1.2.30	<code>entanglement</code>	31
6.1.2.31	<code>evals</code>	31
6.1.2.32	<code>evecs</code>	31
6.1.2.33	<code>expm</code>	32
6.1.2.34	<code>funm</code>	32
6.1.2.35	<code>gcd</code>	32
6.1.2.36	<code>gcd</code>	32
6.1.2.37	<code>gconcurrence</code>	33
6.1.2.38	<code>grams</code>	33
6.1.2.39	<code>grams</code>	33
6.1.2.40	<code>grams</code>	33
6.1.2.41	<code>heig</code>	34
6.1.2.42	<code>hevals</code>	34
6.1.2.43	<code>hevecs</code>	34
6.1.2.44	<code>inverse</code>	34
6.1.2.45	<code>invperm</code>	35
6.1.2.46	<code>kron</code>	35

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

6.1.2.87	<code>ptrace2</code>	47
6.1.2.88	<code>ptranspose</code>	47
6.1.2.89	<code>ptranspose</code>	47
6.1.2.90	<code>qmutualinfo</code>	48
6.1.2.91	<code>qmutualinfo</code>	49
6.1.2.92	<code>rand</code>	49
6.1.2.93	<code>rand</code>	49
6.1.2.94	<code>rand</code>	50
6.1.2.95	<code>rand</code>	50
6.1.2.96	<code>randH</code>	50
6.1.2.97	<code>randidx</code>	50
6.1.2.98	<code>randket</code>	51
6.1.2.99	<code>randkraus</code>	51
6.1.2.100	<code>randn</code>	51
6.1.2.101	<code>randn</code>	51
6.1.2.102	<code>randn</code>	52
6.1.2.103	<code>randn</code>	52
6.1.2.104	<code>randperm</code>	52
6.1.2.105	<code>randrho</code>	53
6.1.2.106	<code>randU</code>	53
6.1.2.107	<code>randV</code>	53
6.1.2.108	<code>renyi</code>	53
6.1.2.109	<code>reshape</code>	54
6.1.2.110	<code>rho2pure</code>	54
6.1.2.111	<code>save</code>	54
6.1.2.112	<code>saveMATLABmatrix</code>	55
6.1.2.113	<code>saveMATLABmatrix</code>	55
6.1.2.114	<code>saveMATLABmatrix</code>	55
6.1.2.115	<code>schatten</code>	55
6.1.2.116	<code>schmidtA</code>	55
6.1.2.117	<code>schmidtB</code>	56
6.1.2.118	<code>schmidtcoeff</code>	56
6.1.2.119	<code>schmidtprob</code>	56
6.1.2.120	<code>shannon</code>	57
6.1.2.121	<code>sinm</code>	57
6.1.2.122	<code>spectralpwm</code>	57
6.1.2.123	<code>sqrtn</code>	57
6.1.2.124	<code>sum</code>	58
6.1.2.125	<code>sum</code>	58
6.1.2.126	<code>super</code>	58

6.1.2.127	svals	58
6.1.2.128	svd	59
6.1.2.129	svdU	59
6.1.2.130	svdV	59
6.1.2.131	syspermute	59
6.1.2.132	syspermute	60
6.1.2.133	trace	60
6.1.2.134	transpose	60
6.1.2.135	tsallis	60
6.1.2.136	x2contfrac	61
6.1.3	Variable Documentation	61
6.1.3.1	chop	61
6.1.3.2	codes	61
6.1.3.3	ee	61
6.1.3.4	eps	61
6.1.3.5	gt	61
6.1.3.6	infty	62
6.1.3.7	init	62
6.1.3.8	maxn	62
6.1.3.9	pi	62
6.1.3.10	rdevs	62
6.1.3.11	st	62
6.2	qpp::experimental Namespace Reference	62
6.2.1	Detailed Description	62
6.3	qpp::internal Namespace Reference	62
6.3.1	Detailed Description	63
6.3.2	Function Documentation	63
6.3.2.1	_check_col_vector	63
6.3.2.2	_check_dims	63
6.3.2.3	_check_dims_match_cvect	63
6.3.2.4	_check_dims_match_mat	63
6.3.2.5	_check_dims_match_rvect	63
6.3.2.6	_check_eq_dims	63
6.3.2.7	_check_nonzero_size	64
6.3.2.8	_check_perm	64
6.3.2.9	_check_row_vector	64
6.3.2.10	_check_square_mat	64
6.3.2.11	_check_subsys_match_dims	64
6.3.2.12	_check_vector	64
6.3.2.13	_kron2	64

6.3.2.14	<code>_multiidx2n</code>	64
6.3.2.15	<code>_n2multiidx</code>	64
6.3.2.16	<code>variadic_vector_emplace</code>	64
6.3.2.17	<code>variadic_vector_emplace</code>	64
<b>7</b>	<b>Class Documentation</b>	<b>65</b>
7.1	<code>qpp::Codes</code> Class Reference	65
7.1.1	Detailed Description	66
7.1.2	Member Enumeration Documentation	66
7.1.2.1	Type	66
7.1.3	Constructor & Destructor Documentation	66
7.1.3.1	<code>Codes</code>	66
7.1.4	Member Function Documentation	66
7.1.4.1	<code>codeword</code>	66
7.1.5	Friends And Related Function Documentation	67
7.1.5.1	<code>internal::Singleton&lt; const Codes &gt;</code>	67
7.2	<code>qpp::Exception</code> Class Reference	67
7.2.1	Detailed Description	69
7.2.2	Member Enumeration Documentation	69
7.2.2.1	Type	69
7.2.3	Constructor & Destructor Documentation	70
7.2.3.1	<code>Exception</code>	70
7.2.3.2	<code>Exception</code>	70
7.2.4	Member Function Documentation	70
7.2.4.1	<code>_construct_exception_msg</code>	70
7.2.4.2	<code>what</code>	70
7.2.5	Member Data Documentation	70
7.2.5.1	<code>_custom</code>	70
7.2.5.2	<code>_msg</code>	71
7.2.5.3	<code>_type</code>	71
7.2.5.4	<code>_where</code>	71
7.3	<code>qpp::Gates</code> Class Reference	71
7.3.1	Detailed Description	73
7.3.2	Constructor & Destructor Documentation	73
7.3.2.1	<code>Gates</code>	73
7.3.3	Member Function Documentation	73
7.3.3.1	<code>CTRL</code>	73
7.3.3.2	<code>expandout</code>	73
7.3.3.3	<code>Fd</code>	74
7.3.3.4	<code>Id</code>	74



7.3.3.5	Rn	74
7.3.3.6	Xd	75
7.3.3.7	Zd	75
7.3.4	Friends And Related Function Documentation	75
7.3.4.1	internal::Singleton< const Gates >	75
7.3.5	Member Data Documentation	75
7.3.5.1	CNOTab	75
7.3.5.2	CNOTba	75
7.3.5.3	CZ	75
7.3.5.4	FRED	75
7.3.5.5	H	76
7.3.5.6	Id2	76
7.3.5.7	S	76
7.3.5.8	SWAP	76
7.3.5.9	T	76
7.3.5.10	TOF	76
7.3.5.11	X	76
7.3.5.12	Y	76
7.3.5.13	Z	76
7.4	qpp::Init Class Reference	76
7.4.1	Detailed Description	77
7.4.2	Constructor & Destructor Documentation	78
7.4.2.1	Init	78
7.4.2.2	~Init	78
7.4.3	Friends And Related Function Documentation	78
7.4.3.1	internal::Singleton< const Init >	78
7.5	qpp::internal::IOManipEigen Class Reference	78
7.5.1	Constructor & Destructor Documentation	78
7.5.1.1	IOManipEigen	78
7.5.1.2	IOManipEigen	78
7.5.2	Friends And Related Function Documentation	78
7.5.2.1	operator<<	79
7.5.3	Member Data Documentation	79
7.5.3.1	_A	79
7.5.3.2	_chop	79
7.6	qpp::internal::IOManipPointer< PointerType > Class Template Reference	79
7.6.1	Constructor & Destructor Documentation	80
7.6.1.1	IOManipPointer	80
7.6.1.2	IOManipPointer	80
7.6.2	Member Function Documentation	80

7.6.2.1	<a href="#">operator=</a>	80
7.6.3	<a href="#">Friends And Related Function Documentation</a>	80
7.6.3.1	<a href="#">operator&lt;&lt;</a>	80
7.6.4	<a href="#">Member Data Documentation</a>	80
7.6.4.1	<a href="#">_end</a>	80
7.6.4.2	<a href="#">_n</a>	80
7.6.4.3	<a href="#">_p</a>	80
7.6.4.4	<a href="#">_separator</a>	80
7.6.4.5	<a href="#">_start</a>	80
7.7	<a href="#">qpp::internal::IOManipRange&lt; InputIterator &gt; Class Template Reference</a>	80
7.7.1	<a href="#">Constructor &amp; Destructor Documentation</a>	81
7.7.1.1	<a href="#">IOManipRange</a>	81
7.7.2	<a href="#">Friends And Related Function Documentation</a>	81
7.7.2.1	<a href="#">operator&lt;&lt;</a>	82
7.7.3	<a href="#">Member Data Documentation</a>	82
7.7.3.1	<a href="#">_end</a>	82
7.7.3.2	<a href="#">_first</a>	82
7.7.3.3	<a href="#">_last</a>	82
7.7.3.4	<a href="#">_separator</a>	82
7.7.3.5	<a href="#">_start</a>	82
7.8	<a href="#">qpp::RandomDevices Class Reference</a>	82
7.8.1	<a href="#">Detailed Description</a>	83
7.8.2	<a href="#">Constructor &amp; Destructor Documentation</a>	83
7.8.2.1	<a href="#">RandomDevices</a>	83
7.8.3	<a href="#">Friends And Related Function Documentation</a>	84
7.8.3.1	<a href="#">internal::Singleton&lt; RandomDevices &gt;</a>	84
7.8.4	<a href="#">Member Data Documentation</a>	84
7.8.4.1	<a href="#">_rd</a>	84
7.8.4.2	<a href="#">_rng</a>	84
7.9	<a href="#">qpp::internal::Singleton&lt; T &gt; Class Template Reference</a>	84
7.9.1	<a href="#">Detailed Description</a>	84
7.9.2	<a href="#">Constructor &amp; Destructor Documentation</a>	85
7.9.2.1	<a href="#">Singleton</a>	85
7.9.2.2	<a href="#">~Singleton</a>	85
7.9.2.3	<a href="#">Singleton</a>	85
7.9.3	<a href="#">Member Function Documentation</a>	85
7.9.3.1	<a href="#">get_instance</a>	85
7.9.3.2	<a href="#">operator=</a>	85
7.10	<a href="#">qpp::States Class Reference</a>	85
7.10.1	<a href="#">Detailed Description</a>	87

7.10.2	Constructor & Destructor Documentation . . . . .	87
7.10.2.1	States . . . . .	87
7.10.3	Friends And Related Function Documentation . . . . .	87
7.10.3.1	internal::Singleton< const States > . . . . .	87
7.10.4	Member Data Documentation . . . . .	87
7.10.4.1	b00 . . . . .	87
7.10.4.2	b01 . . . . .	87
7.10.4.3	b10 . . . . .	88
7.10.4.4	b11 . . . . .	88
7.10.4.5	GHZ . . . . .	88
7.10.4.6	pb00 . . . . .	88
7.10.4.7	pb01 . . . . .	88
7.10.4.8	pb10 . . . . .	88
7.10.4.9	pb11 . . . . .	88
7.10.4.10	pGHZ . . . . .	88
7.10.4.11	pW . . . . .	88
7.10.4.12	px0 . . . . .	88
7.10.4.13	px1 . . . . .	88
7.10.4.14	py0 . . . . .	88
7.10.4.15	py1 . . . . .	89
7.10.4.16	pz0 . . . . .	89
7.10.4.17	pz1 . . . . .	89
7.10.4.18	W . . . . .	89
7.10.4.19	x0 . . . . .	89
7.10.4.20	x1 . . . . .	89
7.10.4.21	y0 . . . . .	89
7.10.4.22	y1 . . . . .	89
7.10.4.23	z0 . . . . .	89
7.10.4.24	z1 . . . . .	89
7.11	qpp::Timer Class Reference . . . . .	89
7.11.1	Detailed Description . . . . .	90
7.11.2	Constructor & Destructor Documentation . . . . .	90
7.11.2.1	Timer . . . . .	90
7.11.3	Member Function Documentation . . . . .	90
7.11.3.1	seconds . . . . .	90
7.11.3.2	tic . . . . .	90
7.11.3.3	toc . . . . .	91
7.11.4	Friends And Related Function Documentation . . . . .	91
7.11.4.1	operator<< . . . . .	91
7.11.5	Member Data Documentation . . . . .	91

7.11.5.1	<code>_end</code>	91
7.11.5.2	<code>_start</code>	91
<b>8</b>	<b>File Documentation</b>	<b>93</b>
8.1	<code>classes/codes.h</code> File Reference	93
8.2	<code>classes/exception.h</code> File Reference	94
8.3	<code>classes/gates.h</code> File Reference	94
8.4	<code>classes/init.h</code> File Reference	95
8.5	<code>classes/random_devices.h</code> File Reference	95
8.6	<code>classes/states.h</code> File Reference	96
8.7	<code>classes/timer.h</code> File Reference	97
8.8	<code>constants.h</code> File Reference	97
8.9	<code>entanglement.h</code> File Reference	98
8.10	<code>entropies.h</code> File Reference	99
8.11	<code>experimental/test.h</code> File Reference	100
8.12	<code>functions.h</code> File Reference	100
8.13	<code>internal/functions.h</code> File Reference	100
8.14	<code>input_output.h</code> File Reference	105
8.15	<code>instruments.h</code> File Reference	106
8.16	<code>internal/classes/iomanip.h</code> File Reference	107
8.17	<code>internal/classes/singleton.h</code> File Reference	108
8.18	<code>MATLAB/matlab.h</code> File Reference	108
8.19	<code>number_theory.h</code> File Reference	109
8.20	<code>operations.h</code> File Reference	110
8.21	<code>qpp.h</code> File Reference	112
8.22	<code>random.h</code> File Reference	113
8.23	<code>types.h</code> File Reference	114
8.24	<code>/Users/vlad/Dropbox/programming/cpp/qpp_clion/mainpage.dox</code> File Reference	115
	<b>Index</b>	<b>116</b>

## Chapter 1

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

### Version

0.1

### Author

Vlad Gheorghiu ([vgheorgh@gmail.com](mailto:vgheorgh@gmail.com))

### Copyright

(c) 2013 - 2014 Vlad Gheorghiu ([vgheorgh@gmail.com](mailto:vgheorgh@gmail.com))

See the file `example.cpp` for some usage examples.



## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">qpp</a> . . . . .	11
<a href="#">qpp::experimental</a> . . . . .	62
<a href="#">qpp::internal</a> . . . . .	62





## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

std::exception	
qpp::Exception . . . . .	67
qpp::internal::IOManipEigen . . . . .	78
qpp::internal::IOManipPointer< PointerType > . . . . .	79
qpp::internal::IOManipRange< InputIterator > . . . . .	80
qpp::internal::Singleton< T > . . . . .	84
qpp::internal::Singleton< const Codes > . . . . .	84
qpp::Codes . . . . .	65
qpp::internal::Singleton< const Gates > . . . . .	84
qpp::Gates . . . . .	71
qpp::internal::Singleton< const Init > . . . . .	84
qpp::Init . . . . .	76
qpp::internal::Singleton< const States > . . . . .	84
qpp::States . . . . .	85
qpp::internal::Singleton< RandomDevices > . . . . .	84
qpp::RandomDevices . . . . .	82
qpp::Timer . . . . .	89



## Chapter 4

# Class Index

### 4.1 Class List

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

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



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">constants.h</a>	97
<a href="#">entanglement.h</a>	98
<a href="#">entropies.h</a>	99
<a href="#">functions.h</a>	100
<a href="#">input_output.h</a>	105
<a href="#">instruments.h</a>	106
<a href="#">number_theory.h</a>	109
<a href="#">operations.h</a>	110
<a href="#">qpp.h</a>	112
<a href="#">random.h</a>	113
<a href="#">types.h</a>	114
<a href="#">classes/codes.h</a>	93
<a href="#">classes/exception.h</a>	94
<a href="#">classes/gates.h</a>	94
<a href="#">classes/init.h</a>	95
<a href="#">classes/random_devices.h</a>	95
<a href="#">classes/states.h</a>	96
<a href="#">classes/timer.h</a>	97
<a href="#">experimental/test.h</a>	100
<a href="#">internal/functions.h</a>	100
<a href="#">internal/classes/iomanip.h</a>	107
<a href="#">internal/classes/singleton.h</a>	108
<a href="#">MATLAB/matlab.h</a>	108



## Chapter 6

# Namespace Documentation

### 6.1 qpp Namespace Reference

#### Namespaces

- [experimental](#)
- [internal](#)

#### Classes

- class [Codes](#)  
*const Singleton class that defines quantum error correcting codes*
- class [Exception](#)  
*Generates custom exceptions, used when validating function parameters.*
- class [Gates](#)  
*const Singleton class that implements most commonly used gates*
- class [Init](#)  
*const Singleton class that performs additional initializations/cleanups*
- class [RandomDevices](#)  
*Singleton class that manages the source of randomness in the library.*
- class [States](#)  
*const Singleton class that implements most commonly used states*
- class [Timer](#)  
*Measures time.*

#### Typedefs

- using [cplx](#) = std::complex< double >  
*Complex number in double precision.*
- template<typename Scalar >  
using [dyn\\_mat](#) = Eigen::Matrix< Scalar, Eigen::Dynamic, Eigen::Dynamic >  
*Dynamic Eigen matrix over the field specified by Scalar.*
- template<typename Scalar >  
using [dyn\\_col\\_vect](#) = Eigen::Matrix< Scalar, Eigen::Dynamic, 1 >  
*Dynamic Eigen column vector over the field specified by Scalar.*
- template<typename Scalar >  
using [dyn\\_row\\_vect](#) = Eigen::Matrix< Scalar, 1, Eigen::Dynamic >

- *Dynamic Eigen row vector over the field specified by Scalar.*  
using `ket = dyn_col_vect< cplx >`  
*Complex (double precision) dynamic Eigen column vector.*
- using `bra = dyn_row_vect< cplx >`  
*Complex (double precision) dynamic Eigen row vector.*
- using `cmat = dyn_mat< cplx >`  
*Complex (double precision) dynamic Eigen matrix.*
- using `dmat = dyn_mat< double >`  
*Real (double precision) dynamic Eigen matrix.*
- using `idx = std::size_t`  
*Non-negative integer index.*

## Functions

- `constexpr cplx operator""_i` (unsigned long long int x)  
*User-defined literal for complex  $i = \sqrt{-1}$  (integer overload)*
- `constexpr cplx operator""_i` (long double x)  
*User-defined literal for complex  $i = \sqrt{-1}$  (real overload)*
- `cplx omega` (idx D)  
*D-th root of unity.*
- `template<typename Derived >`  
`dyn_col_vect< cplx > schmidtcoeff` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)  
*Schmidt coefficients of the bi-partite pure state A.*
- `template<typename Derived >`  
`cmat schmidtA` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)  
*Schmidt basis on Alice's side.*
- `template<typename Derived >`  
`cmat schmidtB` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)  
*Schmidt basis on Bob's side.*
- `template<typename Derived >`  
`dyn_col_vect< double > schmidtprob` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)  
*Schmidt probabilities of the bi-partite pure state A.*
- `template<typename Derived >`  
`double entanglement` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)  
*Entanglement of the bi-partite pure state A.*
- `template<typename Derived >`  
`double gconcurrence` (const Eigen::MatrixBase< Derived > &A)  
*G-concurrence of the bi-partite pure state A.*
- `template<typename Derived >`  
`double negativity` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)  
*Negativity of the bi-partite mixed state A.*
- `template<typename Derived >`  
`double lognegativity` (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)  
*Logarithmic negativity of the bi-partite mixed state A.*
- `template<typename Derived >`  
`double concurrence` (const Eigen::MatrixBase< Derived > &A)  
*Wootters concurrence of the bi-partite qubit mixed state A.*
- `template<typename Derived >`  
`double shannon` (const Eigen::MatrixBase< Derived > &A)  
*Shannon/von-Neumann entropy of the probability distribution/density matrix A.*



- `template<typename Derived >`  
`double renyi (const Eigen::MatrixBase< Derived > &A, double alpha)`  
*Renyi-  $\alpha$  entropy of the probability distribution/density matrix A, for  $\alpha \geq 0$ .*
- `template<typename Derived >`  
`double tsallis (const Eigen::MatrixBase< Derived > &A, double alpha)`  
*Tsallis-  $\alpha$  entropy of the probability distribution/density matrix A, for  $\alpha \geq 0$ .*
- `template<typename Derived >`  
`double qmutualinfo (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsysA, const std::vector< idx > &subsysB, const std::vector< idx > &dims)`  
*Quantum mutual information between 2 subsystems of a composite system.*
- `template<typename Derived >`  
`double qmutualinfo (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsysA, const std::vector< idx > &subsysB, idx d=2)`  
*Quantum mutual information between 2 subsystems of a composite system.*
- `template<typename Derived >`  
`dyn\_mat< typename Derived::Scalar > transpose (const Eigen::MatrixBase< Derived > &A)`  
*Transpose.*
- `template<typename Derived >`  
`dyn\_mat< typename Derived::Scalar > conjugate (const Eigen::MatrixBase< Derived > &A)`  
*Complex conjugate.*
- `template<typename Derived >`  
`dyn\_mat< typename Derived::Scalar > adjoint (const Eigen::MatrixBase< Derived > &A)`  
*Adjoint.*
- `template<typename Derived >`  
`dyn\_mat< typename Derived::Scalar > inverse (const Eigen::MatrixBase< Derived > &A)`  
*Inverse.*
- `template<typename Derived >`  
`Derived::Scalar trace (const Eigen::MatrixBase< Derived > &A)`  
*Trace.*
- `template<typename Derived >`  
`Derived::Scalar det (const Eigen::MatrixBase< Derived > &A)`  
*Determinant.*
- `template<typename Derived >`  
`Derived::Scalar logdet (const Eigen::MatrixBase< Derived > &A)`  
*Logarithm of the determinant.*
- `template<typename Derived >`  
`Derived::Scalar sum (const Eigen::MatrixBase< Derived > &A)`  
*Element-wise sum of A.*
- `template<typename Derived >`  
`Derived::Scalar prod (const Eigen::MatrixBase< Derived > &A)`  
*Element-wise product of A.*
- `template<typename Derived >`  
`double norm (const Eigen::MatrixBase< Derived > &A)`  
*Frobenius norm.*
- `template<typename Derived >`  
`std::pair< dyn\_col\_vect< cplx > , cmat > eig (const Eigen::MatrixBase< Derived > &A)`  
*Full eigen decomposition.*
- `template<typename Derived >`  
`dyn\_col\_vect< cplx > evals (const Eigen::MatrixBase< Derived > &A)`  
*Eigenvalues.*
- `template<typename Derived >`  
`cmat evecs (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.*

- `template<typename Derived >`  
`std::tuple< cmat, dyn_mat`  
`< double >, cmat > svd` (const Eigen::MatrixBase< Derived > &A)

*Full singular value decomposition.*

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

*Singular values.*

- `template<typename Derived >`  
`cmat svdU` (const Eigen::MatrixBase< Derived > &A)

*Left singular vectors.*

- `template<typename Derived >`  
`cmat svdV` (const Eigen::MatrixBase< Derived > &A)

*Right singular vectors.*

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

*Functional calculus  $f(A)$* 

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

*Matrix square root.*

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

*Matrix absolut value.*

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

*Matrix exponential.*

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

*Matrix logarithm.*

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

*Matrix sin.*

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

*Matrix cos.*

- `template<typename Derived >`  
`cmat spectralpwm` (const Eigen::MatrixBase< Derived > &A, const `cplx` z)

*Matrix power.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > pwm` (const Eigen::MatrixBase< Derived > &A, `idx` n)

*Matrix power.*

- `template<typename Derived >`  
`double Schatten` (const Eigen::MatrixBase< Derived > &A, `idx` p)

*Schatten norm.*

- `template<typename OutputScalar, typename Derived >`  
`dyn_mat< OutputScalar > cwise` (const Eigen::MatrixBase< Derived > &A, OutputScalar(\*f)(const 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.*

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

*Kronecker product.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > kron` (const std::initializer\_list< Derived > &As)

*Kronecker product.*

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

*Kronecker power.*

- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > reshape` (const Eigen::MatrixBase< Derived > &A, `idx` rows, `idx`  
cols)

*Reshape.*

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

*Commutator.*

- `template<typename Derived1, typename Derived2 >`  
`dyn_mat< typename`  
`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::initializer\_list< cmat > &Ks, const std::vector< idx > &subsys, const std::vector< idx > &dims)
- Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*
- `template<typename Derived >`  
`std::tuple< idx, std::vector`  
`< double >, std::vector< cmat > > measure` (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks, const std::vector< idx > &subsys, const idx d=2)
- Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*
- `template<typename Derived >`  
`std::tuple< idx, std::vector`  
`< double >, std::vector< cmat > > measure` (const Eigen::MatrixBase< Derived > &A, const std::initializer\_list< cmat > &Ks, const std::vector< idx > &subsys, const idx d=2)
- Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*
- `template<typename Derived >`  
`std::tuple< idx, std::vector`  
`< double >, std::vector< cmat > > measure` (const Eigen::MatrixBase< Derived > &A, const cmat &U, const std::vector< idx > &subsys, const std::vector< idx > &dims)
- Measures the part subsys of the multi-partite state A in the orthonormal basis specified by the unitary matrix U.*
- `template<typename Derived >`  
`std::tuple< idx, std::vector`  
`< double >, std::vector< cmat > > measure` (const Eigen::MatrixBase< Derived > &A, const cmat &U, const std::vector< idx > &subsys, const idx d=2)
- Measures the part subsys of the multi-partite state A in the orthonormal basis specified by the unitary matrix U.*
- `template<typename Derived >`  
`std::tuple< idx, std::vector`  
`< double >, std::vector< cmat > > measure` (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks)
- Measures the state A using the set of Kraus operators Ks.*
- `template<typename Derived >`  
`std::tuple< idx, std::vector`  
`< double >, std::vector< cmat > > measure` (const Eigen::MatrixBase< Derived > &A, const std::initializer\_list< cmat > &Ks)
- Measures the state A using the set of Kraus operators Ks.*
- `template<typename Derived >`  
`std::tuple< idx, std::vector`  
`< double >, std::vector< cmat > > measure` (const Eigen::MatrixBase< Derived > &A, const cmat &U)
- Measures the state A in the orthonormal basis specified by the unitary matrix U.*
- `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 > x2confrac (double x, idx n, idx cut=1e5)`

*Simple continued fraction expansion.*
- `double confrac2x (const std::vector< int > &cf, idx n)`

*Real representation of a simple continued fraction.*
- `double confrac2x (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`  
`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 channel (const Eigen::MatrixBase< Derived > &rho, const std::vector< cmat > &Ks)`

*Applies the channel specified by the set of Kraus operators Ks to the density matrix rho.*
- `template<typename Derived >`  
`cmat channel (const Eigen::MatrixBase< Derived > &rho, const std::vector< cmat > &Ks, const std::vector< 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 channel` (const Eigen::MatrixBase< Derived > &rho, const std::vector< `cmat` > &Ks, const std::vector< `idx` > &subsys, `idx` d=2)  
*Applies the channel specified by the set of Kraus operators Ks to the part subsys of the multi-partite density matrix rho.*
- `cmat super` (const std::vector< `cmat` > &Ks)  
*Superoperator matrix representation.*
- `cmat choi` (const std::vector< `cmat` > &Ks)  
*Choi matrix representation.*
- `std::vector< cmat > choi2kraus` (const `cmat` &A)  
*Extracts orthogonal Kraus operators from Choi matrix.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > ptrace1` (const Eigen::MatrixBase< Derived > &A, const std::vector< `idx` > &dims)  
*Partial trace.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > ptrace2` (const Eigen::MatrixBase< Derived > &A, const std::vector< `idx` > &dims)  
*Partial trace.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > ptrace` (const Eigen::MatrixBase< Derived > &A, const std::vector< `idx` > &subsys, const std::vector< `idx` > &dims)  
*Partial trace.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > ptrace` (const Eigen::MatrixBase< Derived > &A, const std::vector< `idx` > &subsys, `idx` d=2)  
*Partial trace.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > ptranspose` (const Eigen::MatrixBase< Derived > &A, const std::vector< `idx` > &subsys, const std::vector< `idx` > &dims)  
*Partial transpose.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > ptranspose` (const Eigen::MatrixBase< Derived > &A, const std::vector< `idx` > &subsys, `idx` d=2)  
*Partial transpose.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > syspermute` (const Eigen::MatrixBase< Derived > &A, const std::vector< `idx` > &perm, const std::vector< `idx` > &dims)  
*System permutation.*
- `template<typename Derived >`  
`dyn_mat< typename Derived::Scalar > syspermute` (const Eigen::MatrixBase< Derived > &A, const std::vector< `idx` > &perm, `idx` d=2)  
*System permutation.*
- `template<typename Derived >`  
`Derived rand` (`idx` rows, `idx` cols, double a=0, double b=1)  
*Generates a random matrix with entries uniformly distributed in the interval [a, b)*
- `template<>`  
`dmat rand` (`idx` rows, `idx` cols, double a, double b)  
*Generates a random real matrix with entries uniformly distributed in the interval [a, b), specialization for double matrices (`qpp::dmat`)*
- `template<>`  
`cmat rand` (`idx` rows, `idx` cols, double a, double b)  
*Generates a random complex matrix with entries (both real and imaginary) uniformly distributed in the interval [a, b), specialization for complex matrices (`qpp::cmat`)*



- double `rand` (double a=0, double b=1)  
*Generates a random real number uniformly distributed in the interval [a, b]*
- `idx randidx` (`idx` a=std::numeric\_limits< `idx` >::min(), `idx` b=std::numeric\_limits< `idx` >::max())  
*Generates a random index (idx) uniformly distributed in the interval [a, b].*
- 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 qubits.*
- constexpr double `pi` = 3.141592653589793238462643383279502884  
 $\pi$
- constexpr double `ee` = 2.718281828459045235360287471352662497  
*Base of natural logarithm, e.*
- constexpr `idx` `infy` = -1  
*Used to denote infinity.*
- const `Init` & `init` = `Init::get_instance()`  
*`qpp::Init` const Singleton*
- const `Codes` & `codes` = `Codes::get_instance()`  
*`qpp::Codes` const Singleton*
- const `Gates` & `gt` = `Gates::get_instance()`



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

### 6.1.1 Typedef Documentation

#### 6.1.1.1 `using qpp::bra = typedef dyn_row_vect<cplx>`

Complex (double precision) dynamic Eigen row vector.

#### 6.1.1.2 `using qpp::cmat = typedef dyn_mat<cplx>`

Complex (double precision) dynamic Eigen matrix.

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

Complex number in double precision.

#### 6.1.1.4 `using qpp::dmat = typedef dyn_mat<double>`

Real (double precision) dynamic Eigen matrix.

#### 6.1.1.5 `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);
```

#### 6.1.1.6 `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);
```

#### 6.1.1.7 `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);
```

#### 6.1.1.8 using qpp::idx = typedef std::size\_t

Non-negative integer index.

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

Complex (double precision) dynamic Eigen column vector.

### 6.1.2 Function Documentation

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

Matrix absolut value.

Parameters

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

Returns

Matrix absolut value of *A*

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

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

Parameters

<i>first</i>	Iterator to the first element of the range
<i>last</i>	Iterator to the last element of the range

Returns

Real vector consisting of the range's absolut values squared

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

Computes the absolut values squared of a column vector.

Parameters

<i>V</i>	Eigen expression
----------	------------------

Returns

Real vector consisting of the absolut values squared

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

Adjoint.

## Parameters

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

## Returns

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

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

Anti-commutator.

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

## Parameters

$A$	Eigen expression
$B$	Eigen expression

## Returns

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

6.1.2.6 `template<typename Derived1 , typename Derived2 > 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

<i>state</i>	Eigen expression
$A$	Eigen expression
<i>subsys</i>	Subsystem indexes where the gate $A$ is applied
<i>dims</i>	Dimensions of the multi-partite system

## Returns

Gate  $A$  applied to the part *subsys* of *state*

6.1.2.7 `template<typename Derived1 , typename Derived2 > dyn_mat<typename Derived1::Scalar> qpp::apply ( const Eigen::MatrixBase< Derived1 > & state, const Eigen::MatrixBase< Derived2 > & A, const std::vector< idx > & subsys, idx d = 2 )`

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

## Note

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

## Parameters

<i>state</i>	Eigen expression
<i>A</i>	Eigen expression
<i>subsys</i>	Subsystem indexes where the gate <i>A</i> is applied
<i>d</i>	Subsystem dimensions

## Returns

Gate *A* applied to the part *subsys* of *state*

```
6.1.2.8 template<typename Derived1 , typename Derived2 > dyn_mat<typename Derived1::Scalar> qpp::applyCTRL (
    const Eigen::MatrixBase< Derived1 > & state, const Eigen::MatrixBase< Derived2 > & A, const std::vector< idx >
    & ctrl, const std::vector< idx > & subsys, const std::vector< idx > & dims )
```

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

## Note

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

## Parameters

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

## Returns

CTRL-A gate applied to the part *subsys* of *state*

```
6.1.2.9 template<typename Derived1 , typename Derived2 > dyn_mat<typename Derived1::Scalar> qpp::applyCTRL (
    const Eigen::MatrixBase< Derived1 > & state, const Eigen::MatrixBase< Derived2 > & A, const std::vector< idx >
    & ctrl, const std::vector< idx > & subsys, idx d = 2 )
```

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

## Note

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

## Parameters

<i>state</i>	Eigen expression
<i>A</i>	Eigen expression
<i>ctrl</i>	Control subsystem indexes
<i>subsys</i>	Subsystem indexes where the gate <i>A</i> is applied
<i>d</i>	Subsystem dimensions

## Returns

CTRL-A gate applied to the part *subsys* of *state*

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

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

## Parameters

<i>rho</i>	Eigen expression
<i>Ks</i>	Set of Kraus operators

## Returns

Output density matrix after the action of the channel

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

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

## Returns

Output density matrix after the action of the channel

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

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

## Returns

Output density matrix after the action of the channel

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

Choi matrix representation.

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

## Note

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

## Parameters

<i>Ks</i>	Set of Kraus operators
-----------	------------------------

## Returns

Choi matrix representation

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

Extracts orthogonal Kraus operators from Choi matrix.

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

## Note

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

## Parameters

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

## Returns

Set of Kraus operators

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

Commutator.

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

## Parameters

$A$	Eigen expression
$B$	Eigen expression

## Returns

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

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

Compose permutations.

## Parameters

<i>perm</i>	Permutation
<i>sigma</i>	Permutation

## Returns

Composition of the permutations  $perm \circ sigma = perm(sigma)$

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

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

## Parameters

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

## Returns

Wootters concurrence

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

Complex conjugate.

## Parameters

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

## Returns

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

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

Real representation of a simple continued fraction.

## Parameters

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

## Returns

Real representation of the simple continued fraction

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

Real representation of a simple continued fraction.

## Parameters

<i>cf</i>	Integer vector containing the simple continued fraction expansion
-----------	---

## Returns

Real representation of the simple continued fraction

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

Matrix cos.



## Parameters

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

## Returns

Matrix cosine of  $A$

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

Functor.

## Parameters

$A$	Eigen expression
$f$	Pointer-to-function from scalars of $A$ to <i>OutputScalar</i>

## Returns

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

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

Determinant.

## Parameters

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

## Returns

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

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

Eigen expression ostream manipulator.

## Parameters

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

## Returns

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

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

Complex number ostream manipulator.

## Parameters

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

## Returns

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

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

Range ostream manipulator.

## Parameters

<i>first</i>	Iterator to the first element of the range
<i>last</i>	Iterator to the last element of the range
<i>separator</i>	Separator
<i>start</i>	Left marking
<i>end</i>	Right marking

## Returns

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

**6.1.2.27** `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.

## Parameters

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

## Returns

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

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

C-style pointer ostream manipulator.

## Parameters

<i>x</i>	Pointer to the first element
<i>n</i>	Number of elements to be displayed

<i>separator</i>	Separator
<i>start</i>	Left marking
<i>end</i>	Right marking

**Returns**

Instance of `qpp::internal::internal::LOManipPointer`

**6.1.2.29** `template<typename Derived > std::pair<dyn_col_vect<cplx>, cmat> qpp::eig ( const Eigen::MatrixBase< Derived > & A )`

Full eigen decomposition.

**Parameters**

<i>A</i>	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

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

Entanglement of the bi-partite pure state *A*.

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

**See also**

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

**Parameters**

<i>A</i>	Eigen expression
<i>dims</i>	Dimensions of the bi-partite system

**Returns**

Entanglement, with the logarithm in base 2

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

Eigenvalues.

**Parameters**

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

**Returns**

Eigenvalues of *A*, as a complex dynamic column vector

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

Eigenvectors.

## Parameters

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

## Returns

Eigenvectors of  $A$ , as columns of a complex dynamic matrix

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

Matrix exponential.

## Parameters

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

## Returns

Matrix exponential of  $A$

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

Functional calculus  $f(A)$

## Parameters

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

## Returns

$f(A)$

6.1.2.35 `idx qpp::gcd ( idx  $m$ , idx  $n$  )`

Greatest common divisor of two non-negative integers.

## Parameters

$m$	Non-negative integer
$n$	Non-negative integer

## Returns

Greatest common divisor of  $m$  and  $n$

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

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

## Parameters

---

<i>ns</i>	List of non-negative integers
-----------	-------------------------------

**Returns**

Greatest common divisor of all numbers in *ns*

**6.1.2.37** `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**

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

**Returns**

G-concurrence

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

Gram-Schmidt orthogonalization.

**Parameters**

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

**Returns**

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

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

Gram-Schmidt orthogonalization.

**Parameters**

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

**Returns**

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

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

Gram-Schmidt orthogonalization.

## Parameters

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

## Returns

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

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

Full eigen decomposition of Hermitian expression.

## Parameters

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

## Returns

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

**6.1.2.42** `template<typename Derived> dyn_col_vect<double> qpp::hevals ( const Eigen::MatrixBase<Derived> & A )`

Hermitian eigenvalues.

## Parameters

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

## Returns

Eigenvalues of Hermitian  $A$ , as a real dynamic column vector

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

Hermitian eigenvectors.

## Parameters

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

## Returns

Eigenvectors of Hermitian  $A$ , as columns of a complex matrix

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

Inverse.

## Parameters

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

## Returns

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

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

Inverse permutation.

## Parameters

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

## Returns

Inverse of the permutation *perm*

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

Kronecker product.

Used to stop the recursion for the variadic template version of [qpp::kron\(\)](#)

## Parameters

<i>head</i>	Eigen expression
-------------	------------------

## Returns

Its argument *head*

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

Kronecker product.

## Parameters

<i>head</i>	Eigen expression
<i>tail</i>	Variadic Eigen expression (zero or more parameters)

## Returns

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

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

Kronecker product.

## Parameters

<i>As</i>	std::vector of Eigen expressions
-----------	----------------------------------

## Returns

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

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

Kronecker product.

## Parameters

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

## Returns

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

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

Kronecker power.

## Parameters

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

## Returns

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

6.1.2.51 `idx qpp::lcm ( idx m, idx n )`

Least common multiple of two positive integers.

## Parameters

<i>m</i>	Positive integer
<i>n</i>	Positive integer

## Returns

Least common multiple of *m* and *n*

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

Least common multiple of a list of positive integers.



## Parameters

<i>ns</i>	List of positive integers
-----------	---------------------------

## Returns

Least common multiple of all numbers in *ns*

**6.1.2.53** `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.

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

Example:

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

## See also

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

## Parameters

<i>A</i>	Eigen expression
<i>fname</i>	Output file name

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

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

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

**6.1.2.55** `template<> dmat qpp::loadMATLABmatrix ( const std::string & mat_file, const std::string & var_name )`  
`[inline]`

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

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

Example:

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

## Note

If *var\_name* is a complex matrix, only the real part is loaded

## Parameters

<i>mat_file</i>	MATALB .mat file
<i>var_name</i>	Variable name in the .mat file representing the matrix to be loaded

## Returns

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

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

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

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

Example:

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

## Parameters

<i>mat_file</i>	MATALB .mat file
<i>var_name</i>	Variable name in the .mat file representing the matrix to be loaded

## Returns

Eigen complex dynamic matrix ([qpp::cmat](#))

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

Logarithm of the determinant.

Useful when the determinant overflows/underflows

## Parameters

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

## Returns

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

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

Matrix logarithm.

## Parameters

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

## Returns

Matrix logarithm of *A*

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

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

## Parameters

<i>A</i>	Eigen expression
<i>dims</i>	Dimensions of the bi-partite system

## Returns

Logarithmic negativity, with the logarithm in base 2

6.1.2.60 `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

<i>A</i>	Eigen expression
<i>Ks</i>	Set of Kraus operators
<i>subsys</i>	Subsystem indexes that are measured
<i>dims</i>	Dimensions of the multi-partite system

## Returns

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

6.1.2.61 `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

<i>A</i>	Eigen expression
<i>subsys</i>	Subsystem indexes that are measured
<i>dims</i>	Dimensions of the multi-partite system
<i>Ks</i>	Set of Kraus operators

## Returns

Tuple consisting of 1. Result of the measurement,

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

**6.1.2.62** `template<typename Derived> std::tuple<idx, std::vector<double>, std::vector<cmat>> qpp::measure ( const Eigen::MatrixBase< Derived> & A, const std::vector< cmat> & Ks, const std::vector< idx> & subsys, const 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

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

#### Returns

Tuple consisting of 1. Result of the measurement,

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

**6.1.2.63** `template<typename Derived> std::tuple<idx, std::vector<double>, std::vector<cmat>> qpp::measure ( const Eigen::MatrixBase< Derived> & A, const std::initializer_list< cmat> & Ks, const std::vector< idx> & subsys, const 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

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

#### Returns

Tuple consisting of 1. Result of the measurement,

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

**6.1.2.64** `template<typename Derived> std::tuple<idx, std::vector<double>, std::vector<cmat>> qpp::measure ( const Eigen::MatrixBase< Derived> & A, const cmat & U, const std::vector< idx> & subsys, const std::vector< idx> & dims )`

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

#### Note

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

## Parameters

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

## Returns

Tuple consisting of 1. Result of the measurement,

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

**6.1.2.65** `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

<i>A</i>	Eigen expression
<i>subsys</i>	Subsystem indexes that are measured
<i>d</i>	Subsystem dimensions
<i>U</i>	Unitary matrix whose columns represent the measurement basis vectors

## Returns

Tuple consisting of 1. Result of the measurement,

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

**6.1.2.66** `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

<i>A</i>	Eigen expression
<i>Ks</i>	Set of Kraus operators

## Returns

Tuple consisting of 1. Result of the measurement,

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

**6.1.2.67** `template<typename Derived> std::tuple<idx, std::vector<double>, std::vector<cmat>> qpp::measure ( const Eigen::MatrixBase< Derived> & A, const std::initializer_list< cmat> & Ks )`

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

## Parameters

$A$	Eigen expression
$Ks$	Set of Kraus operators

## Returns

Tuple consisting of 1. Result of the measurement,

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

**6.1.2.68** `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

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

## Returns

Tuple consisting of 1. Result of the measurement,

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

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

Multi-partite qudit ket.

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

## Parameters

$\text{mask}$	<code>std::vector</code> of non-negative integers
$\text{dims}$	Dimensions of the multi-partite system

## Returns

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

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

Multi-partite qudit ket.

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

## Parameters

$\text{mask}$	<code>std::vector</code> of non-negative integers
$d$	Subsystem dimensions

## Returns

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

**6.1.2.71** `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  $|\text{mask}\rangle$ , where *mask* is a `std::vector` of non-negative integers. Each element in *mask* has to be smaller than the corresponding element in *dims*.

**Parameters**

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

**Returns**

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

**6.1.2.72** `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  $|\text{mask}\rangle$ , 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**

<i>mask</i>	<code>std::vector</code> of non-negative integers
<i>d</i>	Subsystem dimensions

**Returns**

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

**6.1.2.73** `idx qpp::multiidx2n ( const std::vector< idx > & midx, const std::vector< idx > & dims )`

Multi-index to non-negative integer index.

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

**Parameters**

<i>midx</i>	Multi-index
<i>dims</i>	Dimensions of the multi-partite system

**Returns**

Non-negative integer index

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

Non-negative integer index to multi-index.

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

**Parameters**

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

**Returns**

Multi-index of the same size as  $dims$

**6.1.2.75** `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**

$A$	Eigen expression
$dims$	Dimensions of the bi-partite system

**Returns**

Negativity

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

Frobenius norm.

**Parameters**

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

**Returns**

Frobenius norm of  $A$ , as a real number

**6.1.2.77** `cplx qpp::omega ( idx D )`

$D$ -th root of unity.

**Parameters**

$D$	Non-negative integer
-----	----------------------

**Returns**

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

**6.1.2.78** `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>
```



6.1.2.79 `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>
```

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

Matrix power.

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

Parameters

$A$	Eigen expression
$n$	Non-negative integer

Returns

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

6.1.2.81 `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  $A$

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

Element-wise product of  $A$ .

Parameters

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

Returns

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

6.1.2.83 `template<typename InputIterator > InputIterator::value_type qpp::prod ( InputIterator first, InputIterator last )`

Element-wise product of a range.

## Parameters

<i>first</i>	Iterator to the first element of the range
<i>last</i>	Iterator to the last element of the range

## Returns

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

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

Partial trace.

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

## Parameters

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

## Returns

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

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

Partial trace.

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

## Parameters

<i>A</i>	Eigen expression
<i>subsys</i>	Subsystem indexes
<i>d</i>	Subsystem dimensions

## Returns

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

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

Partial trace.

Partial trace of density matrix over the first subsystem in a bi-partite system

## Parameters

<i>A</i>	Eigen expression
<i>dims</i>	Dimensions of the bi-partite system (must be a <code>std::vector</code> with 2 elements)

**Returns**

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

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

Partial trace.

**Parameters**

<i>A</i>	Eigen expression
<i>dims</i>	Dimensions of the bi-partite system (must be a <code>std::vector</code> with 2 elements)

**Returns**

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

6.1.2.88 `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**

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

**Returns**

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

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

Partial transpose.

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

**Parameters**

<i>A</i>	Eigen expression
<i>subsys</i>	Subsystem indexes
<i>d</i>	Subsystem dimensions

**Returns**

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

6.1.2.90 `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

<i>A</i>	Eigen expression
<i>subsysA</i>	Indexes of the first subsystem
<i>subsysB</i>	Indexes of the second subsystem
<i>dims</i>	Dimensions of the multi-partite system

## Returns

Mutual information between the 2 subsystems

6.1.2.91 `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

<i>A</i>	Eigen expression
<i>subsysA</i>	Indexes of the first subsystem
<i>subsysB</i>	Indexes of the second subsystem
<i>d</i>	Subsystem dimensions

## Returns

Mutual information between the 2 subsystems

6.1.2.92 `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`

6.1.2.93 `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

<i>rows</i>	Number of rows of the random generated matrix
<i>cols</i>	Number of columns of the random generated matrix
<i>a</i>	Beginning of the interval, belongs to it

<i>b</i>	End of the interval, does not belong to it
----------	--

**Returns**

Random real matrix

#### 6.1.2.94 `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**

<i>rows</i>	Number of rows of the random generated matrix
<i>cols</i>	Number of columns of the random generated matrix
<i>a</i>	Beginning of the interval, belongs to it
<i>b</i>	End of the interval, does not belong to it

**Returns**

Random complex matrix

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

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

**Parameters**

<i>a</i>	Beginning of the interval, belongs to it
<i>b</i>	End of the interval, does not belong to it

**Returns**

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

#### 6.1.2.96 `cmat qpp::randH ( idx D )`

Generates a random Hermitian matrix.

**Parameters**

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

**Returns**

Random Hermitian matrix

#### 6.1.2.97 `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].

## Parameters

<i>a</i>	Beginning of the interval, belongs to it
<i>b</i>	End of the interval, belongs to it

## Returns

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

6.1.2.98 ket qpp::randket ( idx *D* )

Generates a random normalized ket (pure state vector)

## Parameters

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

## Returns

Random normalized ket

6.1.2.99 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

<i>N</i>	Number of Kraus operators
<i>D</i>	Dimension of the Hilbert space

## Returns

Set of *N* Kraus operators satisfying the closure condition

6.1.2.100 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](#)

6.1.2.101 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

<i>rows</i>	Number of rows of the random generated matrix
<i>cols</i>	Number of columns of the random generated matrix
<i>mean</i>	Mean
<i>sigma</i>	Standard deviation

## Returns

Random real matrix

**6.1.2.102** `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  $N(\text{mean}, \text{sigma})$ , specialization for complex matrices ([qpp::cmat](#))

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

Example:

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

## Parameters

<i>rows</i>	Number of rows of the random generated matrix
<i>cols</i>	Number of columns of the random generated matrix
<i>mean</i>	Mean
<i>sigma</i>	Standard deviation

## Returns

Random complex matrix

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

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

## Parameters

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

## Returns

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

**6.1.2.104** `std::vector<idx> qpp::randperm ( idx n )`

Generates a random uniformly distributed permutation.

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



## Parameters

$n$	Size of the permutation
-----	-------------------------

## Returns

Random permutation of size  $n$

6.1.2.105 `cmat qpp::randrho ( idx  $D$  )`

Generates a random density matrix.

## Parameters

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

## Returns

Random density matrix

6.1.2.106 `cmat qpp::randU ( idx  $D$  )`

Generates a random unitary matrix.

## Parameters

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

## Returns

Random unitary

6.1.2.107 `cmat qpp::randV ( idx  $D_{in}$ , idx  $D_{out}$  )`

Generates a random isometry matrix.

## Parameters

$D_{in}$	Size of the input Hilbert space
$D_{out}$	Size of the output Hilbert space

## Returns

Random isometry matrix

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

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

## Parameters

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

<i>alpha</i>	Non-negative real number, use <a href="#">qpp::infy</a> for $\alpha = \infty$
--------------	---

**Returns**

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

6.1.2.109 `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**

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

**Returns**

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

6.1.2.110 `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 returns the first non-zero eigenvector of *A*

**Parameters**

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

**Returns**

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

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

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

**See also**

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

## Parameters

<i>A</i>	Eigen expression
<i>fname</i>	Output file name

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

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

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

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

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

## Parameters

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

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

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

## Parameters

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

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

Schatten norm.

## Parameters

<i>A</i>	Eigen expression
<i>p</i>	Integer, greater or equal to 1

## Returns

Schatten-*p* norm of *A*, as a real number

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

Schmidt basis on Alice's side.

## Parameters

<i>A</i>	Eigen expression
<i>dims</i>	Dimensions of the bi-partite system

## Returns

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

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

Schmidt basis on Bob's side.

## Parameters

<i>A</i>	Eigen expression
<i>dims</i>	Dimensions of the bi-partite system

## Returns

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

6.1.2.118 `template<typename Derived > dyn_col_vect<cplx> qpp::schmidtcoeff ( const Eigen::MatrixBase< Derived > & A, const std::vector< idx > & dims )`

Schmidt coefficients of the bi-partite pure state  $A$ .

## Note

The sum of the squares of the Schmidt coefficients equals 1

## See also

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

## Parameters

<i>A</i>	Eigen expression
<i>dims</i>	Dimensions of the bi-partite system

## Returns

Schmidt coefficients of  $A$ , as a complex dynamic column vector

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

Schmidt probabilities of the bi-partite pure state  $A$ .

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

## See also

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

## Parameters

$A$	Eigen expression
$dims$	Dimensions of the bi-partite system

## Returns

Schmidt probabilities of  $A$ , as a real dynamic column vector

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

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

## Parameters

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

## Returns

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

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

Matrix sin.

## Parameters

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

## Returns

Matrix sine of  $A$

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

Matrix power.

Uses the spectral decomposition of  $A$  to compute the matrix power. By convention  $A^0 = I$ .

## Parameters

$A$	Eigen expression
$z$	Complex number

## Returns

Matrix power  $A^z$

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

Matrix square root.

## Parameters

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

## Returns

Matrix square root of  $A$

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

Element-wise sum of  $A$ .

## Parameters

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

## Returns

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

6.1.2.125 `template<typename InputIterator > InputIterator::value_type qpp::sum ( InputIterator first, InputIterator last )`

Element-wise sum of a range.

## Parameters

<i>first</i>	Iterator to the first element of the range
<i>last</i>	Iterator to the last element of the range

## Returns

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

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

Superoperator matrix representation.

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

## Parameters

$K_s$	Set of Kraus operators
-------	------------------------

## Returns

Superoperator matrix representation

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

Singular values.

## Parameters

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

## Returns

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

6.1.2.128 `template<typename Derived > std::tuple<cmat, dyn_mat<double>, cmat> qpp::svd ( const Eigen::MatrixBase< Derived > & A )`

Full singular value decomposition.

## Parameters

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

## Returns

Tuple of: 1. Left singular 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

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

Left singular vectors.

## Parameters

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

## Returns

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

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

Right singular vectors.

## Parameters

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

## Returns

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

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

System permutation.

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

## Parameters

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

## Returns

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

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

System permutation.

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

## Parameters

$A$	Eigen expression
$perm$	Permutation
$d$	Subsystem dimensions

## Returns

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

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

Trace.

## Parameters

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

## Returns

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

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

Transpose.

## Parameters

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

## Returns

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

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

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

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



## Parameters

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

## Returns

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

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

Simple continued fraction expansion.

## Parameters

$x$	Real number
$n$	Number of terms in the expansion
$cut$	Stop the expansion when the next term is greater than $cut$

## Returns

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

## 6.1.3 Variable Documentation

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

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

6.1.3.2 `const Codes& qpp::codes = Codes::get_instance()`

[qpp::Codes](#) const Singleton

Initializes the codes, see the class [qpp::Codes](#)

6.1.3.3 `constexpr double qpp::ee = 2.718281828459045235360287471352662497`

Base of natural logarithm,  $e$ .

6.1.3.4 `constexpr double qpp::eps = 1e-12`

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

Example:

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

6.1.3.5 `const Gates& qpp::gt = Gates::get_instance()`

[qpp::Gates](#) const Singleton

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

#### 6.1.3.6 constexpr idx qpp::infy = -1

Used to denote infinity.

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

[qpp::Init](#) const Singleton

Additional initializations/cleanups

#### 6.1.3.8 constexpr idx qpp::maxn = 64

Maximum number of qubits.

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

#### 6.1.3.9 constexpr double qpp::pi = 3.141592653589793238462643383279502884

$\pi$

#### 6.1.3.10 RandomDevices& qpp::rdevs = RandomDevices::get\_instance()

[qpp::RandomDevices](#) Singleton

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

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

[qpp::States](#) const Singleton

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

## 6.2 qpp::experimental Namespace Reference

### 6.2.1 Detailed Description

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

## 6.3 qpp::internal Namespace Reference

### Classes

- class [IOManipEigen](#)
- class [IOManipPointer](#)
- class [IOManipRange](#)
- class [Singleton](#)

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

## Functions

- void `_n2multiidx` (`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 `_check_row_vector` (const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >  
bool `_check_col_vector` (const Eigen::MatrixBase< Derived > &A)
- template<typename T >  
bool `_check_nonzero_size` (const T &x)
- bool `_check_dims` (const std::vector< `idx` > &dims)
- template<typename Derived >  
bool `_check_dims_match_mat` (const std::vector< `idx` > &dims, const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >  
bool `_check_dims_match_cvect` (const std::vector< `idx` > &dims, const Eigen::MatrixBase< Derived > &V)
- template<typename Derived >  
bool `_check_dims_match_rvect` (const std::vector< `idx` > &dims, const Eigen::MatrixBase< Derived > &V)
- bool `_check_eq_dims` (const std::vector< `idx` > &dims, `idx` dim)
- bool `_check_subsys_match_dims` (const std::vector< `idx` > &subsys, const std::vector< `idx` > &dims)
- bool `_check_perm` (const std::vector< `idx` > &perm)
- template<typename Derived1 , typename Derived2 >  
`dyn_mat`< typename  
Derived1::Scalar > `_kron2` (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2  
> &B)
- template<typename T >  
void `variadic_vector_emplace` (std::vector< T > &)
- template<typename T , typename First , typename... Args>  
void `variadic_vector_emplace` (std::vector< T > &v, First &&first, Args &&...args)

### 6.3.1 Detailed Description

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

### 6.3.2 Function Documentation

6.3.2.1 template<typename Derived > bool qpp::internal::\_check\_col\_vector ( const Eigen::MatrixBase< Derived > & A )

6.3.2.2 bool qpp::internal::\_check\_dims ( const std::vector< `idx` > & dims )

6.3.2.3 template<typename Derived > bool qpp::internal::\_check\_dims\_match\_cvect ( const std::vector< `idx` > & dims,  
const Eigen::MatrixBase< Derived > & V )

6.3.2.4 template<typename Derived > bool qpp::internal::\_check\_dims\_match\_mat ( const std::vector< `idx` > & dims, const  
Eigen::MatrixBase< Derived > & A )

6.3.2.5 template<typename Derived > bool qpp::internal::\_check\_dims\_match\_rvect ( const std::vector< `idx` > & dims,  
const Eigen::MatrixBase< Derived > & V )

6.3.2.6 bool qpp::internal::\_check\_eq\_dims ( const std::vector< `idx` > & dims, `idx` dim )

- 6.3.2.7 `template<typename T > bool qpp::internal::_check_nonzero_size ( const T & x )`
- 6.3.2.8 `bool qpp::internal::_check_perm ( const std::vector< idx > & perm )`
- 6.3.2.9 `template<typename Derived > bool qpp::internal::_check_row_vector ( const Eigen::MatrixBase< Derived > & A )`
- 6.3.2.10 `template<typename Derived > bool qpp::internal::_check_square_mat ( const Eigen::MatrixBase< Derived > & A )`
- 6.3.2.11 `bool qpp::internal::_check_subsys_match_dims ( const std::vector< idx > & subsys, const std::vector< idx > & dims )`
- 6.3.2.12 `template<typename Derived > bool qpp::internal::_check_vector ( const Eigen::MatrixBase< Derived > & A )`
- 6.3.2.13 `template<typename Derived1 , typename Derived2 > dyn_mat<typename Derived1::Scalar> qpp::internal::_kron2 ( const Eigen::MatrixBase< Derived1 > & A, const Eigen::MatrixBase< Derived2 > & B )`
- 6.3.2.14 `idx qpp::internal::_multiidx2n ( const idx * midx, idx numdims, const idx * dims )` `[inline]`
- 6.3.2.15 `void qpp::internal::_n2multiidx ( idx n, idx numdims, const idx * dims, idx * result )` `[inline]`
- 6.3.2.16 `template<typename T > void qpp::internal::variadic_vector_emplace ( std::vector< T > & )`
- 6.3.2.17 `template<typename T , typename First , typename... Args> void qpp::internal::variadic_vector_emplace ( std::vector< T > & v, First && first, Args &&... args )`

## Chapter 7

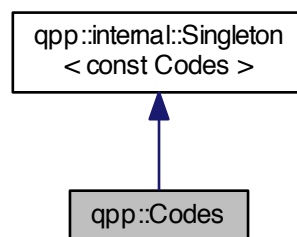
# Class Documentation

### 7.1 qpp::Codes Class Reference

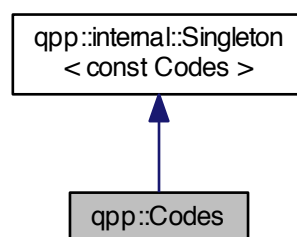
const Singleton class that defines quantum error correcting codes

```
#include <classes/codes.h>
```

Inheritance diagram for qpp::Codes:



Collaboration diagram for qpp::Codes:



## Public Types

- enum `Type` { `Type::FIVE_QUBIT` = 1, `Type::SEVEN_QUBIT_STEANE`, `Type::NINE_QUBIT_SHOR` }  
*Code types, add more codes here if needed.*

## Public Member Functions

- `ket codeword` (`Type` type, `idx` i) const  
*Returns the codeword of the specified code.*

## Private Member Functions

- `Codes` ()  
*Default constructor.*

## Friends

- class `internal::Singleton< const Codes >`

## Additional Inherited Members

### 7.1.1 Detailed Description

const Singleton class that defines quantum error correcting codes

### 7.1.2 Member Enumeration Documentation

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

Code types, add more codes here if needed.

See also

`qpp::Codes::codeword()`

Enumerator

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

### 7.1.3 Constructor & Destructor Documentation

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

Default constructor.

### 7.1.4 Member Function Documentation

#### 7.1.4.1 `ket qpp::Codes::codeword ( Type type, idx i ) const` [inline]

Returns the codeword of the specified code.

## Parameters

<i>type</i>	Code type, defined in the enum <code>qpp::Codes::Types</code>
<i>i</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:

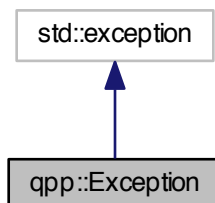
- [classes/codes.h](#)

## 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_CVECTOR`, `Type::MATRIX_NOT_SQUARE_OR_RVECTOR`,  
`Type::MATRIX_NOT_SQUARE_OR_VECTOR`, `Type::MATRIX_MISMATCH_SUBSYS`, `Type::DIMS_INVALID`, `Type::DIMS_NOT_EQUAL`,  
`Type::DIMS_MISMATCH_MATRIX`, `Type::DIMS_MISMATCH_CVECTOR`, `Type::DIMS_MISMATCH_RVECTOR`, `Type::DIMS_MISMATCH_VECTOR`,  
`Type::SUBSYS_MISMATCH_DIMS`, `Type::NOT_QUBIT_GATE`, `Type::NOT_QUBIT_SUBSYS`, `Type::NOT_BIPARTITE`,  
`Type::NO_CODEWORD`, `Type::PERM_INVALID`, `Type::OUT_OF_RANGE`, `Type::TYPE_MISMATCH`,  
`Type::UNDEFINED_TYPE`, `Type::CUSTOM_EXCEPTION` }  
*Exception types, add more exceptions here if needed.*

## Public Member Functions

- `Exception` (const `std::string` &where, const `Type` &type)  
*Constructs an exception.*
- `Exception` (const `std::string` &where, const `std::string` &custom)  
*Constructs an exception.*
- virtual const char \* `what` () const noexcept override  
*Overrides `std::exception::what()`*

## Private Member Functions

- `std::string` `_construct_exception_msg` ()  
*Constructs the exception's description from its type.*



## Private Attributes

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

### 7.2.1 Detailed Description

Generates custom exceptions, used when validating function parameters.

Customize this class if more exceptions are needed

### 7.2.2 Member Enumeration Documentation

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

[Exception](#) types, add more exceptions here if needed.

See also

`qpp::Exception::_construct_exception_msg()`

Enumerator

**UNKNOWN\_EXCEPTION** Unknown exception

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

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

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

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

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

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

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

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

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

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

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

**DIMS\_MISMATCH\_MATRIX** Product of the dimensions of `std::vector<idx>` is not equal to the number of rows of `Eigen::Matrix` (assumed to be square)

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

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

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

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

**NOT\_QUBIT\_GATE** `Eigen::Matrix` is not 2 x 2

**NOT\_QUBIT\_SUBSYS** Subsystems are not 2-dimensional

**NOT\_BIPARTITE** `std::vector<idx>` representing the dimensions has size different from 2

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

**PERM\_INVALID** Invalid `std::vector<idx>` permutation

**OUT\_OF\_RANGE** Parameter out of range

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

**UNDEFINED\_TYPE** Templated function not defined for this type

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

### 7.2.3 Constructor & Destructor Documentation

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

Constructs an exception.

Parameters

<i>where</i>	Text representing where the exception occurred
<i>type</i>	<a href="#">Exception</a> 's type, see the strong enumeration <code>qpp::Exception::Type</code>

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

<i>where</i>	Text representing where the exception occurred
<i>custom</i>	<a href="#">Exception</a> 's description

### 7.2.4 Member Function Documentation

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

Constructs the exception's description from its type.

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

Returns

[Exception](#)'s description

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

Overrides `std::exception::what()`

Returns

[Exception](#)'s description

### 7.2.5 Member Data Documentation

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

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

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

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

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

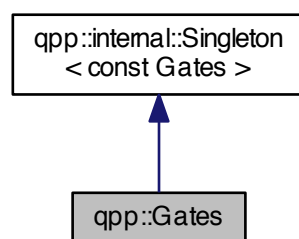
- [classes/exception.h](#)

## 7.3 qpp::Gates Class Reference

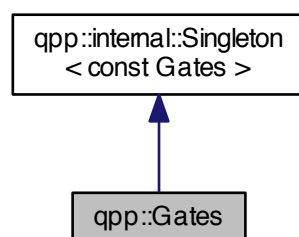
const Singleton class that implements most commonly used gates

```
#include <classes/gates.h>
```

Inheritance diagram for qpp::Gates:



Collaboration diagram for qpp::Gates:



### Public Member Functions

- [cmat 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.*
- `template<typename 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 CNOTab {cmat::Identity(4, 4)}`  
*Controlled-NOT control target gate.*
- `cmat CZ {cmat::Identity(4, 4)}`  
*Controlled-Phase gate.*
- `cmat CNOTba {cmat::Zero(4, 4)}`  
*Controlled-NOT target control gate.*
- `cmat SWAP {cmat::Identity(4, 4)}`  
*SWAP gate.*
- `cmat TOF {cmat::Identity(8, 8)}`  
*Toffoli gate.*
- `cmat FRED {cmat::Identity(8, 8)}`  
*Fredkin gate.*

## Private Member Functions

- `Gates ()`  
*Initializes the gates.*

## Friends

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

## Additional Inherited Members

### 7.3.1 Detailed Description

const Singleton class that implements most commonly used gates

### 7.3.2 Constructor & Destructor Documentation

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

Initializes the gates.

### 7.3.3 Member Function Documentation

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

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

#### Note

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

#### Parameters

<i>A</i>	Eigen expression
<i>ctrl</i>	Control subsystem indexes
<i>subsys</i>	Subsystem indexes where the gate <i>A</i> is applied
<i>n</i>	Total number of subsystems
<i>d</i>	Subsystem dimensions

#### Returns

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

#### 7.3.3.2 template<typename Derived > dyn\_mat<typename Derived::Scalar> qpp::Gates::expandout ( const Eigen::MatrixBase< Derived > & A, idx pos, const std::vector< idx > & dims ) const [inline]

Expands out.

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

#### Parameters

<i>A</i>	Eigen expression
<i>pos</i>	Position

<i>dims</i>	Dimensions of the multi-partite system
-------------	--

**Returns**

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

<i>D</i>	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**

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

**Returns**

Identity gate

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

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

**Parameters**

<i>theta</i>	Rotation angle
<i>n</i>	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::CNOTab {cmat::Identity(4, 4)}`

Controlled-NOT control target gate.

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

Controlled-NOT target control gate.

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

Controlled-Phase gate.

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

Fredkin gate.

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

Hadamard gate.

7.3.5.6 `cmat qpp::Gates::Id2 {cmat::Identity(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::Init::~~Init ( )` `[inline]`, `[private]`

Cleanups.

## 7.4.3 Friends And Related Function Documentation

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

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

- [classes/init.h](#)

## 7.5 `qpp::internal::IOManipEigen` Class Reference

```
#include <internal/classes/iomanip.h>
```

### Public Member Functions

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

### Private Attributes

- `cmat_A`
- `double _chop`

### Friends

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

## 7.5.1 Constructor & Destructor Documentation

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

### 7.5.1.2 `qpp::internal::IOManipEigen::IOManipEigen ( const cplx z, double chop = qpp::chop )` `[inline]`, `[explicit]`

## 7.5.2 Friends And Related Function Documentation

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

### 7.5.3 Member Data Documentation

7.5.3.1 `cmat qpp::internal::IOManipEigen::_A [private]`

7.5.3.2 `double qpp::internal::IOManipEigen::_chop [private]`

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

- [internal/classes/iomanip.h](#)

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

`#include <internal/classes/iomanip.h>`

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



### Public Member Functions

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

### Private Attributes

- `const PointerType * _p`
- `idx _n`

- `std::string _separator`
- `std::string _start`
- `std::string _end`

## Friends

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

## 7.6.1 Constructor & Destructor Documentation

- 7.6.1.1 `template<typename PointerType> qpp::internal::IManipPointer< PointerType >::IManipPointer ( 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::IManipPointer< PointerType >::IManipPointer ( const IManipPointer< PointerType > & ) [default]`

## 7.6.2 Member Function Documentation

- 7.6.2.1 `template<typename PointerType> IManipPointer& qpp::internal::IManipPointer< PointerType >::operator= ( const IManipPointer< 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 IManipPointer< PointerType > & rhs ) [friend]`

## 7.6.4 Member Data Documentation

- 7.6.4.1 `template<typename PointerType> std::string qpp::internal::IManipPointer< PointerType >::_end [private]`
- 7.6.4.2 `template<typename PointerType> idx qpp::internal::IManipPointer< PointerType >::_n [private]`
- 7.6.4.3 `template<typename PointerType> const PointerType* qpp::internal::IManipPointer< PointerType >::_p [private]`
- 7.6.4.4 `template<typename PointerType> std::string qpp::internal::IManipPointer< PointerType >::_separator [private]`
- 7.6.4.5 `template<typename PointerType> std::string qpp::internal::IManipPointer< PointerType >::_start [private]`

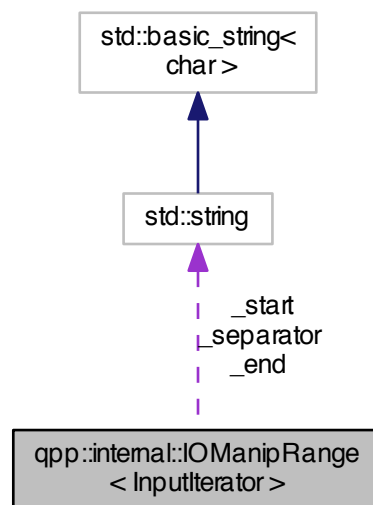
The documentation for this class was generated from the following file:

- `internal/classes/iomanip.h`

## 7.7 qpp::internal::IManipRange< 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

- 7.7.1.1 template<typename InputIterator > qpp::internal::IOManipRange< InputIterator >::IOManipRange ( InputIterator *first*, InputIterator *last*, const std::string & *separator*, const std::string & *start* = " [, const std::string & *end* = "]" ) [inline], [explicit]

## 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>& operator<< ( std::basic_ostream< charT, traits > & os, const IManipRange< InputIterator > & rhs )`  
`[friend]`

### 7.7.3 Member Data Documentation

7.7.3.1 `template<typename InputIterator > std::string qpp::internal::IManipRange< InputIterator >::_end`  
`[private]`

7.7.3.2 `template<typename InputIterator > InputIterator qpp::internal::IManipRange< InputIterator >::_first`  
`[private]`

7.7.3.3 `template<typename InputIterator > InputIterator qpp::internal::IManipRange< InputIterator >::_last`  
`[private]`

7.7.3.4 `template<typename InputIterator > std::string qpp::internal::IManipRange< InputIterator >::_separator`  
`[private]`

7.7.3.5 `template<typename InputIterator > std::string qpp::internal::IManipRange< InputIterator >::_start`  
`[private]`

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

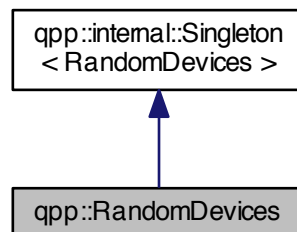
- [internal/classes/iomanip.h](#)

## 7.8 qpp::RandomDevices Class Reference

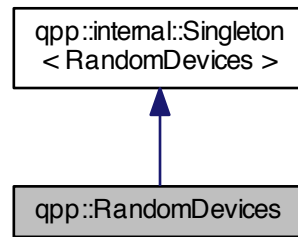
Singleton class that manages the source of randomness in the library.

```
#include <classes/random_devices.h>
```

Inheritance diagram for qpp::RandomDevices:



Collaboration diagram for qpp::RandomDevices:



### Public Attributes

- `std::mt19937 _rng`  
*Mersenne twister random number generator.*

### Private Member Functions

- `RandomDevices ()`  
*Initializes and seeds the random number generators.*

### Private Attributes

- `std::random_device _rd`  
*used to seed std::mt19937 \_rng*

### Friends

- class `internal::Singleton< RandomDevices >`

### Additional Inherited Members

#### 7.8.1 Detailed Description

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

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

#### 7.8.2 Constructor & Destructor Documentation

##### 7.8.2.1 `qpp::RandomDevices::RandomDevices ( )` `[inline]`, `[private]`

Initializes and seeds the random number generators.

### 7.8.3 Friends And Related Function Documentation

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

### 7.8.4 Member Data Documentation

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

used to seed `std::mt19937 _rng`

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

Mersenne twister random number generator.

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

- [classes/random\\_devices.h](#)

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

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

```
#include <internal/classes/singleton.h>
```

### Static Public Member Functions

- static `T & get_instance ()`

### Protected Member Functions

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

#### 7.9.1 Detailed Description

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

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

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

Example:

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



```

    {
        // Implement the constructor here
    }
};

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

```

See also

Code of [qpp::Codes](#), [qpp::Gates](#), [qpp::RandomDevices](#), [qpp::States](#) or [qpp.h](#) for real world examples of usage.

## 7.9.2 Constructor & Destructor Documentation

**7.9.2.1** `template<typename T> qpp::internal::Singleton< T >::Singleton ( )` `[inline]`, `[protected]`

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

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

## 7.9.3 Member Function Documentation

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

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

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

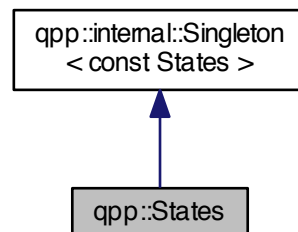
- [internal/classes/singleton.h](#)

## 7.10 qpp::States Class Reference

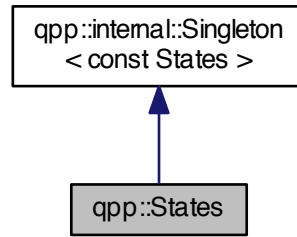
const Singleton class that implements most commonly used states

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

Inheritance diagram for qpp::States:



Collaboration diagram for `qpp::States`:



## Public Attributes

- `ket x0` {ket::Zero(2)}  
Pauli Sigma-X 0-eigenstate  $|+\rangle$
- `ket x1` {ket::Zero(2)}  
Pauli Sigma-X 1-eigenstate  $|-\rangle$
- `ket y0` {ket::Zero(2)}  
Pauli Sigma-Y 0-eigenstate  $|y+\rangle$
- `ket y1` {ket::Zero(2)}  
Pauli Sigma-Y 1-eigenstate  $|y-\rangle$
- `ket z0` {ket::Zero(2)}  
Pauli Sigma-Z 0-eigenstate  $|0\rangle$
- `ket z1` {ket::Zero(2)}  
Pauli Sigma-Z 1-eigenstate  $|1\rangle$
- `cmat px0` {cmat::Zero(2, 2)}  
Projector onto the Pauli Sigma-X 0-eigenstate  $|+\rangle\langle+|$ .
- `cmat px1` {cmat::Zero(2, 2)}  
Projector onto the Pauli Sigma-X 1-eigenstate  $|-\rangle\langle-|$ .
- `cmat py0` {cmat::Zero(2, 2)}  
Projector onto the Pauli Sigma-Y 0-eigenstate  $|y+\rangle\langle y+|$ .
- `cmat py1` {cmat::Zero(2, 2)}  
Projector onto the Pauli Sigma-Y 1-eigenstate  $|y-\rangle\langle y-|$ .
- `cmat pz0` {cmat::Zero(2, 2)}  
Projector onto the Pauli Sigma-Z 0-eigenstate  $|0\rangle\langle 0|$ .
- `cmat pz1` {cmat::Zero(2, 2)}  
Projector onto the Pauli Sigma-Z 1-eigenstate  $|1\rangle\langle 1|$ .
- `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)

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  $|+\rangle\langle+|$ .

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

Projector onto the Pauli Sigma-X 1-eigenstate  $|-\rangle\langle-|$ .

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

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

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

Projector onto the Pauli Sigma-Y 1-eigenstate  $|y-\rangle\langle y-|$ .

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

Projector onto the Pauli Sigma-Z 0-eigenstate  $|0\rangle\langle 0|$ .

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

Projector onto the Pauli Sigma-Z 1-eigenstate  $|1\rangle\langle 1|$ .

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  $|+\rangle$

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

Pauli Sigma-X 1-eigenstate  $|-\rangle$

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

Pauli Sigma-Y 0-eigenstate  $|y+\rangle$

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

Pauli Sigma-Y 1-eigenstate  $|y-\rangle$

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

Pauli Sigma-Z 0-eigenstate  $|0\rangle$

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

Pauli Sigma-Z 1-eigenstate  $|1\rangle$

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

- [classes/states.h](#)

## 7.11 qpp::Timer Class Reference

Measures time.

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

## Public Member Functions

- [Timer](#) ()  
*Constructs an instance with the current time as the starting point.*
- void [tic](#) ()  
*Resets the chronometer.*
- const [Timer](#) & [toc](#) ()  
*Stops the chronometer.*
- double [seconds](#) () const  
*Time passed in seconds.*

## Protected Attributes

- std::chrono::steady\_clock::time\_point [\\_start](#)
- std::chrono::steady\_clock::time\_point [\\_end](#)

## Friends

- template<typename charT , typename traits >  
std::basic\_ostream< charT,  
traits > & [operator<<](#) (std::basic\_ostream< charT, traits > &os, const [Timer](#) &rhs)  
*Overload for std::ostream operators.*

### 7.11.1 Detailed Description

Measures time.

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

### 7.11.2 Constructor & Destructor Documentation

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

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

### 7.11.3 Member Function Documentation

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

Time passed in seconds.

#### Returns

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

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

Resets the chronometer.

Resets the starting/ending point to the current time

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

Stops the chronometer.

Set the current time as the ending point

#### Returns

Current instance

## 7.11.4 Friends And Related Function Documentation

### 7.11.4.1 template<typename charT , typename traits > std::basic\_ostream<charT, traits>& operator<< ( std::basic\_ostream< charT, traits > & os, const Timer & rhs ) [friend]

Overload for std::ostream operators.

#### Parameters

<i>os</i>	Output stream
<i>rhs</i>	<a href="#">Timer</a> 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](#)



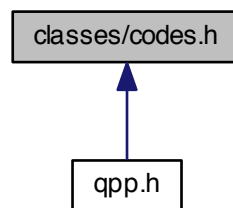


## Chapter 8

# File Documentation

### 8.1 classes/codes.h File Reference

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



#### Classes

- class [qpp::Codes](#)

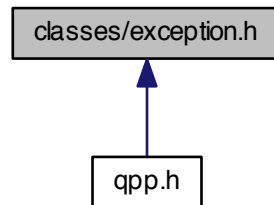
*const Singleton class that defines quantum error correcting codes*

#### Namespaces

- [qpp](#)

## 8.2 classes/exception.h File Reference

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



### Classes

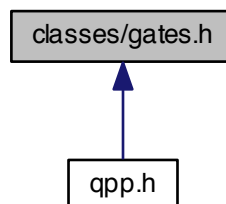
- class [qpp::Exception](#)  
*Generates custom exceptions, used when validating function parameters.*

### Namespaces

- [qpp](#)

## 8.3 classes/gates.h File Reference

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



### Classes

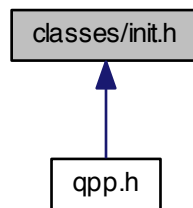
- class [qpp::Gates](#)  
*const Singleton class that implements most commonly used gates*

## Namespaces

- [qpp](#)

## 8.4 classes/init.h File Reference

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



## Classes

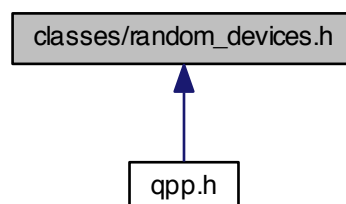
- class [qpp::Init](#)  
*const Singleton class that performs additional initializations/cleanups*

## Namespaces

- [qpp](#)

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

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



## Classes

- class [qpp::RandomDevices](#)

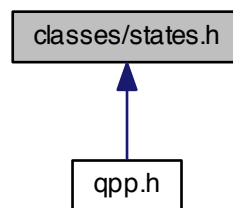
*Singleton class that manages the source of randomness in the library.*

## Namespaces

- [qpp](#)

## 8.6 classes/states.h File Reference

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



## Classes

- class [qpp::States](#)

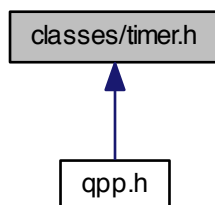
*const Singleton class that implements most commonly used states*

## Namespaces

- [qpp](#)

## 8.7 classes/timer.h File Reference

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



### Classes

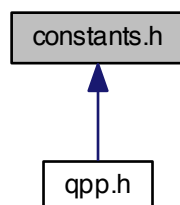
- class [qpp::Timer](#)  
*Measures time.*

### Namespaces

- [qpp](#)

## 8.8 constants.h File Reference

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



### Namespaces

- [qpp](#)

## Functions

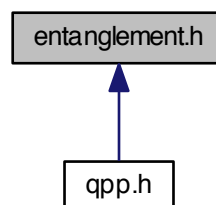
- constexpr cplx [qpp::operator""\\_i](#) (unsigned long long int x)  
*User-defined literal for complex  $i = \sqrt{-1}$  (integer overload)*
- constexpr cplx [qpp::operator""\\_i](#) (long double x)  
*User-defined literal for complex  $i = \sqrt{-1}$  (real overload)*
- cplx [qpp::omega](#) (idx D)  
*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 qubits.*
- constexpr double [qpp::pi](#) = 3.141592653589793238462643383279502884  
 $\pi$
- constexpr double [qpp::ee](#) = 2.718281828459045235360287471352662497  
*Base of natural logarithm,  $e$ .*
- constexpr idx [qpp::infy](#) = -1  
*Used to denote infinity.*

## 8.9 entanglement.h File Reference

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



## Namespaces

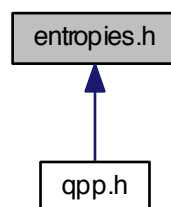
- [qpp](#)

## Functions

- `template<typename Derived >`  
`dyn_col_vect< cplx > qpp::schmidtcoeff (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)`  
*Schmidt coefficients of the bi-partite pure state A.*
- `template<typename Derived >`  
`cmat qpp::schmidtA (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)`  
*Schmidt basis on Alice's side.*
- `template<typename Derived >`  
`cmat qpp::schmidtB (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)`  
*Schmidt basis on Bob's side.*
- `template<typename Derived >`  
`dyn_col_vect< double > qpp::schmidtprob (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)`  
*Schmidt probabilities of the bi-partite pure state A.*
- `template<typename Derived >`  
`double qpp::entanglement (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)`  
*Entanglement of the bi-partite pure state A.*
- `template<typename Derived >`  
`double qpp::gconcurrence (const Eigen::MatrixBase< Derived > &A)`  
*G-concurrence of the bi-partite pure state A.*
- `template<typename Derived >`  
`double qpp::negativity (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)`  
*Negativity of the bi-partite mixed state A.*
- `template<typename Derived >`  
`double qpp::lognegativity (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)`  
*Logarithmic negativity of the bi-partite mixed state A.*
- `template<typename Derived >`  
`double qpp::concurrence (const Eigen::MatrixBase< Derived > &A)`  
*Wootters concurrence of the bi-partite qubit mixed state A.*

## 8.10 entropies.h File Reference

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



## Namespaces

- [qpp](#)

## Functions

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

## 8.11 experimental/test.h File Reference

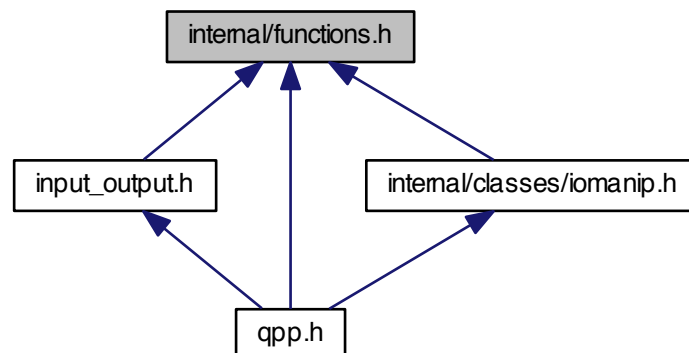
### Namespaces

- `qpp::experimental`
- `qpp`

## 8.12 functions.h File Reference

## 8.13 internal/functions.h File Reference

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





## Namespaces

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

## 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 qpp::trace (const Eigen::MatrixBase< Derived > &A)`  
*Trace.*
- `template<typename Derived >`  
`Derived::Scalar qpp::det (const Eigen::MatrixBase< Derived > &A)`  
*Determinant.*
- `template<typename Derived >`  
`Derived::Scalar qpp::logdet (const Eigen::MatrixBase< Derived > &A)`  
*Logarithm of the determinant.*
- `template<typename Derived >`  
`Derived::Scalar qpp::sum (const Eigen::MatrixBase< Derived > &A)`  
*Element-wise sum of A.*
- `template<typename Derived >`  
`Derived::Scalar qpp::prod (const Eigen::MatrixBase< Derived > &A)`  
*Element-wise product of A.*
- `template<typename Derived >`  
`double qpp::norm (const Eigen::MatrixBase< Derived > &A)`  
*Frobenius norm.*
- `template<typename Derived >`  
`std::pair< dyn_col_vect< cplx >`  
`, cmat > qpp::eig (const Eigen::MatrixBase< Derived > &A)`  
*Full eigen decomposition.*
- `template<typename Derived >`  
`dyn_col_vect< cplx > qpp::evals (const Eigen::MatrixBase< Derived > &A)`  
*Eigenvalues.*
- `template<typename Derived >`  
`cmat qpp::evecs (const Eigen::MatrixBase< Derived > &A)`  
*Eigenvectors.*
- `template<typename Derived >`  
`std::pair< dyn_col_vect`  
`< double >, cmat > qpp::heig (const Eigen::MatrixBase< Derived > &A)`  
*Full eigen decomposition of Hermitian expression.*
- `template<typename Derived >`  
`dyn_col_vect< double > qpp::hevals (const Eigen::MatrixBase< Derived > &A)`

*Hermitian eigenvalues.*

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

*Hermitian eigenvectors.*

- `template<typename Derived >`  
`std::tuple< cmat, dyn_mat`  
`< double >, cmat > qpp::svd (const Eigen::MatrixBase< Derived > &A)`

*Full singular value decomposition.*

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

*Singular values.*

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

*Left singular vectors.*

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

*Right singular vectors.*

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

*Functional calculus  $f(A)$*

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

*Matrix square root.*

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

*Matrix absolut value.*

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

*Matrix exponential.*

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

*Matrix logarithm.*

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

*Matrix sin.*

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

*Matrix cos.*

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

*Matrix power.*

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

*Matrix power.*

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

*Schatten norm.*

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

*Functor.*

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

- Kronecker product.*

  - template<typename T, typename... Args>  
dyn\_mat< typename T::Scalar > [qpp::kron](#) (const T &head, const Args &...tail)
- Kronecker product.*

  - template<typename Derived >  
dyn\_mat< typename Derived::Scalar > [qpp::kron](#) (const std::vector< Derived > &As)
- Kronecker product.*

  - 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 Derived >  
dyn\_mat< typename Derived::Scalar > [qpp::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< Derived2 > &B)
- Commutator.*

  - template<typename Derived1, typename Derived2 >  
dyn\_mat< typename  
Derived1::Scalar > [qpp::anticomm](#) (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)
- Anti-commutator.*

  - template<typename Derived >  
dyn\_mat< typename Derived::Scalar > [qpp::prj](#) (const Eigen::MatrixBase< Derived > &V)
- Projector.*

  - template<typename Derived >  
dyn\_mat< typename Derived::Scalar > [qpp::grams](#) (const std::vector< Derived > &Vs)
- Gram-Schmidt orthogonalization.*

  - template<typename Derived >  
dyn\_mat< typename Derived::Scalar > [qpp::grams](#) (const std::initializer\_list< Derived > &Vs)
- Gram-Schmidt orthogonalization.*

  - template<typename Derived >  
dyn\_mat< typename Derived::Scalar > [qpp::grams](#) (const Eigen::MatrixBase< Derived > &A)
- Gram-Schmidt orthogonalization.*

  - std::vector< idx > [qpp::n2multiidx](#) (idx n, const std::vector< idx > &dims)
- Non-negative integer index to multi-index.*

  - idx [qpp::multiidx2n](#) (const std::vector< idx > &midx, const std::vector< idx > &dims)
- Multi-index to non-negative integer index.*

  - ket [qpp::mket](#) (const std::vector< idx > &mask, const std::vector< idx > &dims)
- Multi-partite qudit ket.*

  - ket [qpp::mket](#) (const std::vector< idx > &mask, idx d=2)
- Multi-partite qudit ket.*

  - cmat [qpp::mprj](#) (const std::vector< idx > &mask, const std::vector< idx > &dims)
- Projector onto multi-partite qudit ket.*

  - cmat [qpp::mprj](#) (const std::vector< idx > &mask, idx d=2)
- Projector onto multi-partite qudit ket.*

  - template<typename InputIterator >  
std::vector< double > [qpp::abssq](#) (InputIterator first, InputIterator last)

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

- `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)`

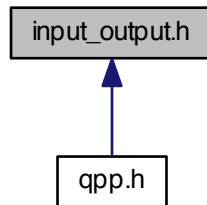
*Finds the pure state representation of a matrix proportional to a projector onto a pure state.*

- `void qpp::internal::\_n2multiidx (idx n, idx numdims, const idx *dims, idx *result)`
- `idx qpp::internal::\_multiidx2n (const idx *midx, idx numdims, const idx *dims)`
- `template<typename Derived >`  
`bool qpp::internal::\_check\_square\_mat (const Eigen::MatrixBase< Derived > &A)`
- `template<typename Derived >`  
`bool qpp::internal::\_check\_vector (const Eigen::MatrixBase< Derived > &A)`
- `template<typename Derived >`  
`bool qpp::internal::\_check\_row\_vector (const Eigen::MatrixBase< Derived > &A)`
- `template<typename Derived >`  
`bool qpp::internal::\_check\_col\_vector (const Eigen::MatrixBase< Derived > &A)`
- `template<typename T >`  
`bool qpp::internal::\_check\_nonzero\_size (const T &x)`
- `bool qpp::internal::\_check\_dims (const std::vector< idx > &dims)`
- `template<typename Derived >`  
`bool qpp::internal::\_check\_dims\_match\_mat (const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &A)`
- `template<typename Derived >`  
`bool qpp::internal::\_check\_dims\_match\_cvect (const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &V)`
- `template<typename Derived >`  
`bool qpp::internal::\_check\_dims\_match\_rvect (const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &V)`
- `bool qpp::internal::\_check\_eq\_dims (const std::vector< idx > &dims, idx dim)`
- `bool qpp::internal::\_check\_subsys\_match\_dims (const std::vector< idx > &subsys, const std::vector< idx > &dims)`
- `bool qpp::internal::\_check\_perm (const std::vector< idx > &perm)`
- `template<typename Derived1 , typename Derived2 >`  
`dyn_mat< typename`  
`Derived1::Scalar > qpp::internal::\_kron2 (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)`
- `template<typename T >`  
`void qpp::internal::variadic\_vector\_emplace (std::vector< T > &)`
- `template<typename T , typename First , typename... Args>`  
`void qpp::internal::variadic\_vector\_emplace (std::vector< T > &v, First &&first, Args &&...args)`

## 8.14 input\_output.h File Reference

```
#include <internal/functions.h>
```

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



### Namespaces

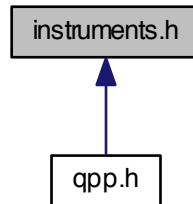
- [qpp](#)

### Functions

- `template<typename Derived >`  
`internal::IOManipEigen qpp::disp (const Eigen::MatrixBase< Derived > &A, double chop=qpp::chop)`  
*Eigen expression ostream manipulator.*
- `internal::IOManipEigen qpp::disp (cplx z, double chop=qpp::chop)`  
*Complex number ostream manipulator.*
- `template<typename InputIterator >`  
`internal::IOManipRange`  
`< InputIterator > qpp::disp (const InputIterator &first, const InputIterator &last, const std::string &separator,`  
`const std::string &start="["`, `const std::string &end=""]")`  
*Range ostream manipulator.*
- `template<typename Container >`  
`internal::IOManipRange`  
`< typename`  
`Container::const_iterator > qpp::disp (const Container &c, const std::string &separator, const std::string`  
`&start="["`, `const std::string &end=""]")`  
*Standard container ostream manipulator. The container must support std::begin(), std::end() and forward iteration.*
- `template<typename PointerType >`  
`internal::IOManipPointer`  
`< PointerType > qpp::disp (const PointerType *p, 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.*

## 8.15 instruments.h File Reference

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



### Namespaces

- [qpp](#)

### 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::vector< 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::vector< 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.*

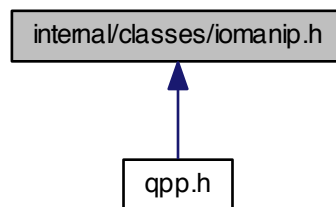
- `template<typename Derived >`  
`std::tuple< idx, std::vector`  
`< double >, std::vector< cmat > >` [qpp::measure](#) (const Eigen::MatrixBase< Derived > &A, const cmat &U)

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

## 8.16 internal/classes/iomanip.h File Reference

```
#include <internal/functions.h>
```

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



### Classes

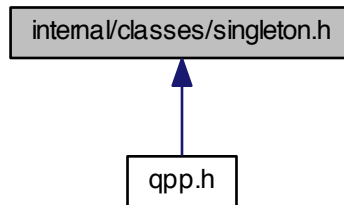
- class [qpp::internal::IOManipRange< InputIterator >](#)
- class [qpp::internal::IOManipPointer< PointerType >](#)
- class [qpp::internal::IOManipEigen](#)

### Namespaces

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

## 8.17 internal/classes/singleton.h File Reference

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



### Classes

- class [qpp::internal::Singleton< T >](#)  
*[Singleton](#) policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern)*

### Namespaces

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

## 8.18 MATLAB/matlab.h File Reference

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

### Namespaces

- [qpp](#)

### Functions

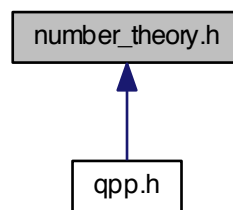
- template<typename Derived >  
 Derived [qpp::loadMATLABmatrix](#) (const std::string &mat\_file, const std::string &var\_name)  
*Loads an Eigen dynamic matrix from a MATLAB .mat file, generic version.*
- template<>  
 dmat [qpp::loadMATLABmatrix](#) (const std::string &mat\_file, const std::string &var\_name)  
*Loads an Eigen dynamic matrix from a MATLAB .mat file, specialization for double matrices ([qpp::dmat](#))*
- template<>  
 cmat [qpp::loadMATLABmatrix](#) (const std::string &mat\_file, const std::string &var\_name)  
*Loads an Eigen dynamic matrix from a MATLAB .mat file, specialization for complex matrices ([qpp::cmat](#))*



- `template<typename Derived >`  
`void qpp::saveMATLABmatrix (const Eigen::MatrixBase< Derived > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)`  
*Saves an Eigen dynamic matrix to a MATLAB .mat file, generic version.*
- `template<>`  
`void qpp::saveMATLABmatrix (const Eigen::MatrixBase< dmat > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)`  
*Saves an Eigen dynamic matrix to a MATLAB .mat file, specialization for double matrices ([qpp::dmat](#))*
- `template<>`  
`void qpp::saveMATLABmatrix (const Eigen::MatrixBase< cmat > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)`  
*Saves an Eigen dynamic matrix to a MATLAB .mat file, specialization for complex matrices ([qpp::cmat](#))*

## 8.19 number\_theory.h File Reference

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



### Namespaces

- [qpp](#)

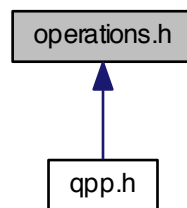
### Functions

- `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.*

## 8.20 operations.h File Reference

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



### Namespaces

- [qpp](#)

### Functions

- `template<typename Derived1 , typename Derived2 >`  
`dyn_mat< typename`  
`Derived1::Scalar > qpp::applyCTRL` (const `Eigen::MatrixBase< Derived1 > &state`, const `Eigen::MatrixBase< Derived2 > &A`, const `std::vector< idx > &ctrl`, const `std::vector< idx > &subsys`, const `std::vector< idx > &dims`)  
*Applies the controlled-gate A to the part subsys of the multi-partite state vector or density matrix state.*
- `template<typename Derived1 , typename Derived2 >`  
`dyn_mat< typename`  
`Derived1::Scalar > qpp::applyCTRL` (const `Eigen::MatrixBase< Derived1 > &state`, const `Eigen::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 > qpp::apply` (const `Eigen::MatrixBase< Derived1 > &state`, const `Eigen::MatrixBase< Derived2 > &A`, const `std::vector< idx > &subsys`, const `std::vector< idx > &dims`)  
*Applies the gate A to the part subsys of the multi-partite state vector or density matrix state.*
- `template<typename Derived1 , typename Derived2 >`  
`dyn_mat< typename`  
`Derived1::Scalar > qpp::apply` (const `Eigen::MatrixBase< Derived1 > &state`, const `Eigen::MatrixBase< Derived2 > &A`, const `std::vector< idx > &subsys`, `idx d=2`)  
*Applies the gate A to the part subsys of the multi-partite state vector or density matrix state.*
- `template<typename Derived >`  
`cmat qpp::channel` (const `Eigen::MatrixBase< Derived > &rho`, const `std::vector< cmat > &Ks`)

*Applies the channel specified by the set of Kraus operators  $K_s$  to the density matrix  $\rho$ .*

- `template<typename Derived >`  
`cmat qpp::channel (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  $K_s$  to the part subsys of the multi-partite density matrix  $\rho$ .*

- `template<typename Derived >`  
`cmat qpp::channel (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  $K_s$  to the part subsys of the multi-partite density matrix  $\rho$ .*

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

*Superoperator matrix representation.*

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

*Choi matrix representation.*

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

*Extracts orthogonal Kraus operators from Choi matrix.*

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

*Partial trace.*

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

*Partial trace.*

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

*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 >`  
`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)`

*System permutation.*

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

*System permutation.*

## 8.21 qpp.h File Reference

```
#include <algorithm>
#include <chrono>
#include <cmath>
#include <complex>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <exception>
#include <fstream>
#include <functional>
#include <initializer_list>
#include <iomanip>
#include <iostream>
#include <iterator>
#include <limits>
#include <numeric>
#include <ostream>
#include <random>
#include <sstream>
#include <stdexcept>
#include <string>
#include <tuple>
#include <type_traits>
#include <utility>
#include <vector>
#include <Eigen/Dense>
#include <Eigen/SVD>
#include "types.h"
#include "constants.h"
#include "classes/exception.h"
#include "internal/classes/iomanip.h"
#include "input_output.h"
#include "internal/classes/singleton.h"
#include "classes/init.h"
#include "internal/functions.h"
#include "functions.h"
#include "classes/codes.h"
#include "classes/gates.h"
#include "classes/states.h"
#include "classes/random_devices.h"
#include "operations.h"
#include "entropies.h"
#include "entanglement.h"
#include "random.h"
#include "classes/timer.h"
#include "instruments.h"
#include "number_theory.h"
```

### Namespaces

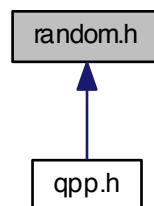
- [qpp](#)

## Variables

- 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*

## 8.22 random.h File Reference

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



## Namespaces

- [qpp](#)

## Functions

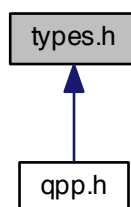
- template<typename Derived >  
Derived [qpp::rand](#) (idx rows, idx cols, double a=0, double b=1)  
*Generates a random matrix with entries uniformly distributed in the interval [a, b)*
- template<>  
dmat [qpp::rand](#) (idx rows, idx cols, double a, double b)  
*Generates a random real matrix with 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  $N(\text{mean}, \text{sigma})$*
  - `template<>`  
`dmat qpp::randn (idx rows, idx cols, double mean, double sigma)`  
*Generates a random real matrix with entries normally distributed in  $N(\text{mean}, \text{sigma})$ , 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  $N(\text{mean}, \text{sigma})$ , specialization for complex matrices ([qpp::cmat](#))*
  - `double qpp::randn (double mean=0, double sigma=1)`  
*Generates a random real number (double) normally distributed in  $N(\text{mean}, \text{sigma})$*
  - `cmat qpp::randU (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.*

## 8.23 types.h File Reference

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



## Namespaces

- [qpp](#)

## Typedefs

- using `qpp::cplx` = `std::complex< double >`  
*Complex number in double precision.*
- template<typename Scalar >  
using `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.*
- template<typename Scalar >  
using `qpp::dyn_row_vect` = `Eigen::Matrix< Scalar, 1, Eigen::Dynamic >`  
*Dynamic Eigen row vector over the field specified by Scalar.*
- using `qpp::ket` = `dyn_col_vect< cplx >`  
*Complex (double precision) dynamic Eigen column vector.*
- using `qpp::bra` = `dyn_row_vect< cplx >`  
*Complex (double precision) dynamic Eigen row vector.*
- using `qpp::cmat` = `dyn_mat< cplx >`  
*Complex (double precision) dynamic Eigen matrix.*
- using `qpp::dmat` = `dyn_mat< double >`  
*Real (double precision) dynamic Eigen matrix.*
- using `qpp::idx` = `std::size_t`  
*Non-negative integer index.*

## 8.24 /Users/vlad/Dropbox/programming/cpp/qpp\_clion/mainpage.dox File Reference

# Index

absm  
    qpp, [22](#)  
abssq  
    qpp, [22](#)  
adjoint  
    qpp, [22](#)  
anticomm  
    qpp, [23](#)  
apply  
    qpp, [23](#)  
  
bra  
    qpp, [21](#)  
  
CUSTOM\_EXCEPTION  
    qpp::Exception, [70](#)  
channel  
    qpp, [24](#), [26](#)  
choi  
    qpp, [26](#)  
choi2kraus  
    qpp, [27](#)  
chop  
    qpp, [61](#)  
cmat  
    qpp, [21](#)  
codes  
    qpp, [61](#)  
comm  
    qpp, [27](#)  
compperm  
    qpp, [27](#)  
concurrence  
    qpp, [27](#)  
conjugate  
    qpp, [28](#)  
contrac2x  
    qpp, [28](#)  
cosm  
    qpp, [28](#)  
cplx  
    qpp, [21](#)  
cwise  
    qpp, [29](#)  
  
DIMS\_INVALID  
    qpp::Exception, [69](#)  
DIMS\_MISMATCH\_CVECTOR  
    qpp::Exception, [69](#)  
DIMS\_MISMATCH\_MATRIX  
    qpp::Exception, [69](#)  
DIMS\_MISMATCH\_RVECTOR  
    qpp::Exception, [69](#)  
DIMS\_MISMATCH\_VECTOR  
    qpp::Exception, [69](#)  
DIMS\_NOT\_EQUAL  
    qpp::Exception, [69](#)  
det  
    qpp, [29](#)  
disp  
    qpp, [29](#), [30](#)  
dmat  
    qpp, [21](#)  
  
ee  
    qpp, [61](#)  
eig  
    qpp, [31](#)  
entanglement  
    qpp, [31](#)  
eps  
    qpp, [61](#)  
evals  
    qpp, [31](#)  
evects  
    qpp, [31](#)  
expm  
    qpp, [32](#)  
  
FIVE\_QUBIT  
    qpp::Codes, [66](#)  
funm  
    qpp, [32](#)  
  
gcd  
    qpp, [32](#)  
gconcurrence  
    qpp, [33](#)  
grams  
    qpp, [33](#)  
gt  
    qpp, [61](#)  
  
heig  
    qpp, [34](#)  
hevals  
    qpp, [34](#)  
hevects  
    qpp, [34](#)  
  
idx



- qpp, [21](#)
- infty
  - qpp, [61](#)
- init
  - qpp, [62](#)
- inverse
  - qpp, [34](#)
- invperm
  - qpp, [35](#)
- ket
  - qpp, [22](#)
- kron
  - qpp, [35](#), [36](#)
- kronpow
  - qpp, [36](#)
- lcm
  - qpp, [36](#)
- load
  - qpp, [37](#)
- logdet
  - qpp, [38](#)
- logm
  - qpp, [38](#)
- lognegativity
  - qpp, [38](#)
- MATRIX\_MISMATCH\_SUBSYS
  - qpp::Exception, [69](#)
- MATRIX\_NOT\_CVECTOR
  - qpp::Exception, [69](#)
- MATRIX\_NOT\_RVECTOR
  - qpp::Exception, [69](#)
- MATRIX\_NOT\_SQUARE
  - qpp::Exception, [69](#)
- MATRIX\_NOT\_SQUARE\_OR\_CVECTOR
  - qpp::Exception, [69](#)
- MATRIX\_NOT\_SQUARE\_OR\_RVECTOR
  - qpp::Exception, [69](#)
- MATRIX\_NOT\_SQUARE\_OR\_VECTOR
  - qpp::Exception, [69](#)
- MATRIX\_NOT\_VECTOR
  - qpp::Exception, [69](#)
- maxn
  - qpp, [62](#)
- measure
  - qpp, [39–42](#)
- mket
  - qpp, [42](#)
- mprj
  - qpp, [42](#), [43](#)
- multiidx2n
  - qpp, [43](#)
- n2multiidx
  - qpp, [43](#)
- NINE\_QUBIT\_SHOR
  - qpp::Codes, [66](#)
- NO\_CODEWORD
  - qpp::Exception, [69](#)
- NOT\_BIPARTITE
  - qpp::Exception, [69](#)
- NOT\_QUBIT\_GATE
  - qpp::Exception, [69](#)
- NOT\_QUBIT\_SUBSYS
  - qpp::Exception, [69](#)
- negativity
  - qpp, [44](#)
- norm
  - qpp, [44](#)
- OUT\_OF\_RANGE
  - qpp::Exception, [70](#)
- omega
  - qpp, [44](#)
- PERM\_INVALID
  - qpp::Exception, [70](#)
- pi
  - qpp, [62](#)
- powm
  - qpp, [45](#)
- prj
  - qpp, [45](#)
- prod
  - qpp, [45](#)
- ptrace
  - qpp, [46](#)
- ptrace1
  - qpp, [46](#)
- ptrace2
  - qpp, [47](#)
- ptranspose
  - qpp, [47](#)
- qmutualinfo
  - qpp, [47](#), [49](#)
- qpp, [11](#)
  - absm, [22](#)
  - abssq, [22](#)
  - adjoint, [22](#)
  - anticomm, [23](#)
  - apply, [23](#)
  - bra, [21](#)
  - channel, [24](#), [26](#)
  - choi, [26](#)
  - choi2kraus, [27](#)
  - chop, [61](#)
  - cmat, [21](#)
  - codes, [61](#)
  - comm, [27](#)
  - compperm, [27](#)
  - concurrence, [27](#)
  - conjugate, [28](#)
  - contfrac2x, [28](#)
  - cosm, [28](#)
  - cplx, [21](#)

- cwise, 29
- det, 29
- disp, 29, 30
- dmat, 21
- ee, 61
- eig, 31
- entanglement, 31
- eps, 61
- evals, 31
- evects, 31
- expm, 32
- funm, 32
- gcd, 32
- gconcurrence, 33
- grams, 33
- gt, 61
- heig, 34
- hevals, 34
- hevects, 34
- idx, 21
- infty, 61
- init, 62
- inverse, 34
- invperm, 35
- ket, 22
- kron, 35, 36
- kronpow, 36
- lcm, 36
- load, 37
- logdet, 38
- logm, 38
- lognegativity, 38
- maxn, 62
- measure, 39–42
- mket, 42
- mprj, 42, 43
- multiidx2n, 43
- n2multiidx, 43
- negativity, 44
- norm, 44
- omega, 44
- pi, 62
- powm, 45
- prj, 45
- prod, 45
- ptrace, 46
- ptrace1, 46
- ptrace2, 47
- ptranspose, 47
- qmutualinfo, 47, 49
- rand, 49, 50
- randidx, 50
- randket, 51
- randkraus, 51
- randn, 51, 52
- randperm, 52
- randrho, 53
- rdevs, 62
- renyi, 53
- reshape, 54
- rho2pure, 54
- save, 54
- schatten, 55
- schmidtcoeff, 56
- schmidtprob, 56
- shannon, 57
- sinm, 57
- spectralpowm, 57
- sqrtn, 57
- st, 62
- sum, 58
- super, 58
- svals, 58
- svd, 59
- syspermute, 59, 60
- trace, 60
- transpose, 60
- tsallis, 60
- x2confrac, 61
- qpp::Codes
  - FIVE\_QUBIT, 66
  - NINE\_QUBIT\_SHOR, 66
  - SEVEN\_QUBIT\_STEANE, 66
- qpp::Exception
  - CUSTOM\_EXCEPTION, 70
  - DIMS\_INVALID, 69
  - DIMS\_MISMATCH\_CVECTOR, 69
  - DIMS\_MISMATCH\_MATRIX, 69
  - DIMS\_MISMATCH\_RVECTOR, 69
  - DIMS\_MISMATCH\_VECTOR, 69
  - DIMS\_NOT\_EQUAL, 69
  - MATRIX\_MISMATCH\_SUBSYS, 69
  - MATRIX\_NOT\_CVECTOR, 69
  - MATRIX\_NOT\_RVECTOR, 69
  - MATRIX\_NOT\_SQUARE, 69
  - MATRIX\_NOT\_SQUARE\_OR\_CVECTOR, 69
  - MATRIX\_NOT\_SQUARE\_OR\_RVECTOR, 69
  - MATRIX\_NOT\_SQUARE\_OR\_VECTOR, 69
  - MATRIX\_NOT\_VECTOR, 69
  - NO\_CODEWORD, 69
  - NOT\_BIPARTITE, 69
  - NOT\_QUBIT\_GATE, 69
  - NOT\_QUBIT\_SUBSYS, 69
  - OUT\_OF\_RANGE, 70
  - PERM\_INVALID, 70
  - SUBSYS\_MISMATCH\_DIMS, 69
  - TYPE\_MISMATCH, 70
  - UNDEFINED\_TYPE, 70
  - UNKNOWN\_EXCEPTION, 69
  - ZERO\_SIZE, 69
- rand
  - qpp, 49, 50
- randidx
  - qpp, 50
- randket
  - qpp, 51

randkraus  
  qpp, [51](#)  
randn  
  qpp, [51](#), [52](#)  
randperm  
  qpp, [52](#)  
randrho  
  qpp, [53](#)  
rdevs  
  qpp, [62](#)  
renyi  
  qpp, [53](#)  
reshape  
  qpp, [54](#)  
rho2pure  
  qpp, [54](#)  
  
SEVEN\_QUBIT\_STEANE  
  qpp::Codes, [66](#)  
SUBSYS\_MISMATCH\_DIMS  
  qpp::Exception, [69](#)  
save  
  qpp, [54](#)  
schatten  
  qpp, [55](#)  
schmidtcoeff  
  qpp, [56](#)  
schmidtprob  
  qpp, [56](#)  
shannon  
  qpp, [57](#)  
sinm  
  qpp, [57](#)  
spectralpowm  
  qpp, [57](#)  
sqrtm  
  qpp, [57](#)  
st  
  qpp, [62](#)  
sum  
  qpp, [58](#)  
super  
  qpp, [58](#)  
svals  
  qpp, [58](#)  
svd  
  qpp, [59](#)  
syspermute  
  qpp, [59](#), [60](#)  
  
TYPE\_MISMATCH  
  qpp::Exception, [70](#)  
trace  
  qpp, [60](#)  
transpose  
  qpp, [60](#)  
tsallis  
  qpp, [60](#)  
  
UNDEFINED\_TYPE  
  qpp::Exception, [70](#)  
UNKNOWN\_EXCEPTION  
  qpp::Exception, [69](#)  
  
x2contfrac  
  qpp, [61](#)  
  
ZERO\_SIZE  
  qpp::Exception, [69](#)