quantum++

0.1

Generated by Doxygen 1.8.7

Sun Oct 26 2014 14:39:56

Contents

Chapter 1

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

Version

0.1

Author

Vlad Gheorghiu, vgheorgh@gmail.com

Date

October 26, 2014

This is the main page of the documentation. More coming soon.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 qpp Namespace Reference

Namespaces

- internal

Classes

- class DiscreteDistribution
- class DiscreteDistributionAbsSquare
- class Exception

  Generates custom exceptions, used when validating function parameters.
- class Gates

  Singleton class that implements most commonly used gates.
- class NormalDistribution

  Template light wrapper around std::normal_distribution<>, generates normally-distributed random numbers.
- class Qudit
- class RandomDevices

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

  Singleton class that implements most commonly used states.
- class Timer

  Measures time.
- class UniformIntegerDistribution
- class UniformRealDistribution

Typedefs

- using cplx = std::complex< double >

  Complex number in double precision.
- using cmat = Eigen::MatrixXcd

  Complex (double precision) dynamic Eigen matrix.
- using dmat = Eigen::MatrixXd

  Real (double precision) dynamic Eigen matrix.
- using ket = Eigen::Matrix< cplx, Eigen::Dynamic, 1 >

  Complex (double precision) dynamic Eigen column matrix.

- using bra = Eigen::Matrix< cplx, 1, Eigen::Dynamic >

  Complex (double precision) dynamic Eigen row matrix.
- template<typename Scalar >
  using DynMat = Eigen::Matrix< Scalar, Eigen::Dynamic, Eigen::Dynamic >

  Dynamic Eigen matrix over the field specified by Scalar.

Functions

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

  Superoperator matrix representation.
- cmat choi (const std::vector< cmat > &Ks)

  Choi matrix representation.
- std::vector< cmat > choi2kraus (const cmat &A)

  Extracts orthogonal Kraus operators from Choi matrix.
- template<typename Derived >
  cmat channel (const Eigen::MatrixBase< Derived > &rho, const std::vector< cmat > &Ks)

  Applies the channel specified by the set of Kraus operators Ks to the density matrix rho.
- template<typename Derived >
  cmat channel (const Eigen::MatrixBase< Derived > &rho, const std::vector< cmat > &Ks, const std::vector< std::size_t > &subsys, const std::vector< std::size_t > &dims)

  Applies the channel specified by the set of Kraus operators Ks to the part of the density matrix rho specified by subsys.
- constexpr std::complex< double > operator""_i (unsigned long long int x)

  User-defined literal for complex $i = \sqrt{-1}$ (integer overload)
- constexpr std::complex< double > operator""_i (long double x)

  User-defined literal for complex $i = \sqrt{-1}$ (real overload)
- std::complex< double > omega (std::size_t D)

  D-th root of unity.
- template<typename Derived >
  cmat schmidtcoeff (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &dims)

  Schmidt coefficients of the bi-partite pure state A.
- template<typename Derived >
  cmat schmidtU (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &dims)

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

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

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

  Entanglement of the bi-partite pure state A.
- template<typename Derived >
  double gconcurrence (const Eigen::MatrixBase< Derived > &A)

  G-concurrence of the bi-partite pure state A.
- template<typename Derived >
  double shannon (const Eigen::MatrixBase< Derived > &A)

  Shannon/von-Neumann entropy of the probability distribution/density matrix A.
- template<typename Derived >
  double renyi (const double alpha, const Eigen::MatrixBase< Derived > &A)

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

- template<typename Derived >
  double renyi_inf (const Eigen::MatrixBase< Derived > &A)

  Renyi- ∞ entropy (min entropy) of the probability distribution/density matrix A.
- template<typename Derived >
  double tsallis (const double alpha, const Eigen::MatrixBase< Derived > &A)

  Tsallis- α entropy of the probability distribution/density matrix A, for $\alpha \geq 0$
  .
- template<typename Derived >
  double qmutualinfo (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &subsysA, const std::vector< std::size_t > &subsysB, const std::vector< std::size_t > &dims)

  Quantum mutual information between 2 subsystems of a composite system.
- template<typename Derived >
  DynMat< typename Derived::Scalar > transpose (const Eigen::MatrixBase< Derived > &A)

  Transpose.
- template<typename Derived >
  DynMat< typename Derived::Scalar > conjugate (const Eigen::MatrixBase< Derived > &A)

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

  Adjoint.
- template<typename Derived >
  DynMat< typename Derived::Scalar > inverse (const Eigen::MatrixBase< Derived > &A)

  Inverse.
- template<typename Derived >
  Derived::Scalar trace (const Eigen::MatrixBase< Derived > &A)

  Trace.
- template<typename Derived >
  Derived::Scalar det (const Eigen::MatrixBase< Derived > &A)

  Determinant.
- template<typename Derived >
  Derived::Scalar logdet (const Eigen::MatrixBase< Derived > &A)

  Logarithm of the determinant.
- template<typename Derived >
  Derived::Scalar sum (const Eigen::MatrixBase< Derived > &A)

  Element-wise sum.
- template<typename Derived >
  double norm (const Eigen::MatrixBase< Derived > &A)

  Trace norm.
- template<typename Derived >
  cmat evals (const Eigen::MatrixBase< Derived > &A)

  Eigenvalues.
- template<typename Derived >
  cmat evects (const Eigen::MatrixBase< Derived > &A)

  Eigenvectors.
- template<typename Derived >
  dmat hevals (const Eigen::MatrixBase< Derived > &A)

  Hermitian eigenvalues.
- template<typename Derived >
  cmat hevects (const Eigen::MatrixBase< Derived > &A)

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

  Functional calculus f(A)

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

  Matrix square root.
- template<typename Derived >
  cmat absm (const Eigen::MatrixBase< Derived > &A)

  Matrix absolut value.
- template<typename Derived >
  cmat expm (const Eigen::MatrixBase< Derived > &A)

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

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

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

  Matrix cos.
- template<typename Derived >
  cmat spectralpowm (const Eigen::MatrixBase< Derived > &A, const cplx z)

  Matrix power.
- template<typename Derived >
  DynMat< typename Derived::Scalar > powm (const Eigen::MatrixBase< Derived > &A, std::size_t n)

  Matrix power.
- template<typename OutputScalar , typename Derived >
  DynMat< OutputScalar > cwise (const Eigen::MatrixBase< Derived > &A, OutputScalar(*f)(const typename Derived::Scalar &))

  Functor.
- template<typename T >
  DynMat< typename T::Scalar > kron (const T &head)

  Kronecker product (variadic overload)
- template<typename T , typename... Args>
  DynMat< typename T::Scalar > kron (const T &head, const Args &...tail)

  Kronecker product (variadic overload)
- template<typename Derived >
  DynMat< typename Derived::Scalar > kron (const std::vector< Derived > &As)

  Kronecker product (std::vector overload)
- template<typename Derived >
  DynMat< typename Derived::Scalar > kron (const std::initializer_list< Derived > &As)

  Kronecker product (std::initializer_list overload)
- template<typename Derived >
  DynMat< typename Derived::Scalar > kronpow (const Eigen::MatrixBase< Derived > &A, std::size_t n)

  Kronecker power.
- template<typename Derived >
  DynMat< typename Derived::Scalar > reshape (const Eigen::MatrixBase< Derived > &A, std::size_t rows, std::size_t cols)

  Reshape.
- template<typename Derived >
  DynMat< typename Derived::Scalar > syspermute (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &perm, const std::vector< std::size_t > &dims)

  System permutation.
- template<typename Derived >
  DynMat< typename Derived::Scalar > ptrace1 (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &dims)

Partial trace.

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

  Partial trace.
- template<typename Derived >
  DynMat< typename Derived::Scalar > ptrace (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &subsys, const std::vector< std::size_t > &dims)

  Partial trace.
- template<typename Derived >
  DynMat< typename Derived::Scalar > ptranspose (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &subsys, const std::vector< std::size_t > &dims)

  Partial transpose.
- template<typename Derived1 , typename Derived2 >
  DynMat< typename Derived1::Scalar > comm (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)

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

  Anti-commutator.
- template<typename Derived >
  DynMat< typename Derived::Scalar > prj (const Eigen::MatrixBase< Derived > &V)

  Projector.
- template<typename Derived >
  DynMat< typename Derived::Scalar > expandout (const Eigen::MatrixBase< Derived > &A, std::size_t pos, const std::vector< std::size_t > &dims)

  Expand out.
- template<typename Derived >
  DynMat< typename Derived::Scalar > grams (const std::vector< Derived > &Vs)

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

  Gram-Schmidt orthogonalization (std::initializer_list overload)
- template<typename Derived >
  DynMat< typename Derived::Scalar > grams (const Eigen::MatrixBase< Derived > &A)

  Gram-Schmidt orthogonalization (Eigen expression (matrix) overload)
- std::vector< std::size_t > n2multiidx (std::size_t n, const std::vector< std::size_t > &dims)

  Non-negative integer index to multi-index.
- std::size_t multiidx2n (const std::vector< std::size_t > &midx, const std::vector< std::size_t > &dims)

  Multi-index to non-negative integer index.
- ket mket (const std::vector< std::size_t > &mask)

  Multi-partite qubit ket.
- ket mket (const std::vector< std::size_t > &mask, const std::vector< std::size_t > &dims)

  Multi-partite qudit ket (different dimensions overload)
- ket mket (const std::vector< std::size_t > &mask, std::size_t d)

  Multi-partite qudit ket (same dimensions overload)
- std::vector< std::size_t > invperm (const std::vector< std::size_t > &perm)

  Inverse permutation.
- std::vector< std::size_t > compperm (const std::vector< std::size_t > &perm, const std::vector< std::size_t > &sigma)

  Compose permutations.

- template<typename T >
  void disp (const T &x, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

  Displays a standard container that supports std::begin, std::end and forward iteration. Does not add a newline.
- template<typename T >
  void displn (const T &x, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

  Displays a standard container that supports std::begin, std::end and forward iteration. Adds a newline.
- template<typename T >
  void disp (const T *x, const std::size_t n, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

  Displays a C-style array. Does not add a newline.
- template<typename T >
  void displn (const T *x, const std::size_t n, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

  Displays a C-style array. Adds a newline.
- template<typename Derived >
  void disp (const Eigen::MatrixBase< Derived > &A, double chop=qpp::chop, std::ostream &os=std::cout)

  Displays an Eigen expression in matrix friendly form. Does not add a new line.
- template<typename Derived >
  void displn (const Eigen::MatrixBase< Derived > &A, double chop=qpp::chop, std::ostream &os=std::cout)

  Displays an Eigen expression in matrix friendly form. Adds a newline.
- void disp (const cplx z, double chop=qpp::chop, std::ostream &os=std::cout)

  Displays a number (implicitly converted to std::complex<double>) in friendly form. Does not add a new line.
- void displn (const cplx z, double chop=qpp::chop, std::ostream &os=std::cout)

  Displays a number (implicitly converted to std::complex<double>) in friendly form. Adds a new line.
- template<typename Derived >
  void save (const Eigen::MatrixBase< Derived > &A, const std::string &fname)

  Saves Eigen expression to a binary file (internal format) in double precission.
- template<typename Derived >
  DynMat< typename Derived::Scalar > load (const std::string &fname)

  Loads Eigen matrix from a binary file (internal format) in double precission.
- template<typename Derived >
  Derived loadMATLABmatrix (const std::string &mat_file, const std::string &var_name)

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

  Loads an Eigen dynamic matrix from a MATLAB .mat file, specialization for double matrices (qpp::dmat)
- template<>
  cmat loadMATLABmatrix (const std::string &mat_file, const std::string &var_name)

  Loads an Eigen dynamic matrix from a MATLAB .mat file, specialization for complex matrices (qpp::cmat)
- template<typename Derived >
  void saveMATLABmatrix (const Eigen::MatrixBase< Derived > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)

  Saves an Eigen dynamic matrix to a MATLAB .mat file, generic version.
- template<>
  void saveMATLABmatrix (const Eigen::MatrixBase< dmat > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)

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

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

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

  Generates a random matrix with entries uniformly distributed in the interval [a, b)
- template<>
  dmat rand (std::size_t rows, std::size_t cols, double a, double b)

  Generates a random real matrix with entries uniformly distributed in the interval [a, b), specialization for double matrices (qpp::dmat)
- template<>
  cmat rand (std::size_t rows, std::size_t cols, double a, double b)

  Generates a random complex matrix with entries (both real and imaginary) uniformly distributed in the interval [a, b), specialization for complex matrices (qpp::cmat)
- double rand (double a=0, double b=1)

  Generates a random real number uniformly distributed in the interval [a, b)
- int randint (int a=std::numeric_limits< int >::min(), int b=std::numeric_limits< int >::max())

  Generates a random integer (int) uniformly distributed in the interval [a, b].
- template<typename Derived >
  Derived randn (std::size_t rows, std::size_t cols, double mean=0, double sigma=1)

  Generates a random matrix with entries normally distributed in N(mean, sigma)
- template<>
  dmat randn (std::size_t rows, std::size_t cols, double mean, double sigma)

  Generates a random real matrix with entries normally distributed in N(mean, sigma), specialization for double matrices (qpp::dmat)
- template<>
  cmat randn (std::size_t rows, std::size_t cols, double mean, double sigma)

  Generates a random complex matrix with entries (both real and imaginary) normally distributed in N(mean, sigma), specialization for complex matrices (qpp::cmat)
- double randn (double mean=0, double sigma=1)

  Generates a random real number (double) normally distributed in N(mean, sigma)
- cmat randU (std::size_t D)

  Generates a random unitary matrix.
- cmat randV (std::size_t Din, std::size_t Dout)

  Generates a random isometry matrix.
- std::vector< cmat > randkraus (std::size_t n, std::size_t D)

  Generates a set of random Kraus operators.
- cmat randH (std::size_t D)

  Generates a random Hermitian matrix.
- ket randket (std::size_t D)

  Generates a random normalized ket (pure state vector)
- cmat randrho (std::size_t D)

  Generates a random density matrix.
- std::vector< std::size_t > randperm (std::size_t n)

  Generates a random uniformly distributed permutation.

Variables

- constexpr double chop = 1e-10

  Used in qpp::disp() and qpp::displn() for setting to zero numbers that have their absolute value smaller than qpp::ct↩ ::chop.
- constexpr double eps = 1e-12

  Used to decide whether a number or expression in double precision is zero or not.
- constexpr std::size_t maxn = 64

  Maximum number of qubits.

- constexpr double pi = 3.14159265358979323846264338327950288 4

  π
- constexpr double ee = 2.71828182845904523536028747135266249 7

  Base of natural logarithm, e.
- RandomDevices & rdevs = RandomDevices::get_instance()

  qpp::RandomDevices Singleton
- const Gates & gt = Gates::get_instance()

  qpp::Gates const Singleton
- const States & st = States::get_instance()

  qpp::States const Singleton

6.1.1 Typedef Documentation

6.1.1.1 using qpp::bra = typedef Eigen::Matrix<cplx, 1, Eigen::Dynamic>

Complex (double precision) dynamic Eigen row matrix.

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

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 Eigen::MatrixXd

Real (double precision) dynamic Eigen matrix.

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

Dynamic Eigen matrix over the field specified by *Scalar*.

Example:

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

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

Complex (double precision) dynamic Eigen column matrix.

6.1.2 Function Documentation

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

Matrix absolut value.

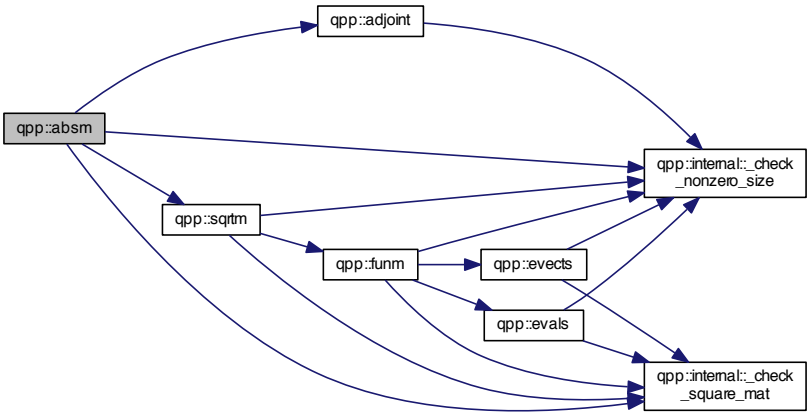
Parameters

A	Eigen expression

Returns

Matrix absolut value of *A*

Here is the call graph for this function:



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

Adjoint.

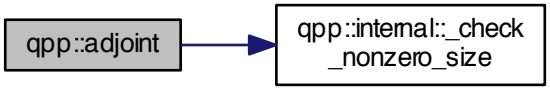
Parameters

A	Eigen expression

Returns

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

Here is the call graph for this function:



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

Anti-commutator.

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

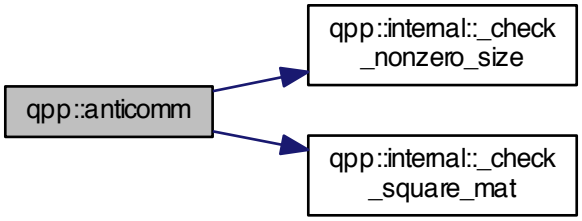
Parameters

A	Eigen expression
B	Eigen expression

Returns

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

Here is the call graph for this function:



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

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

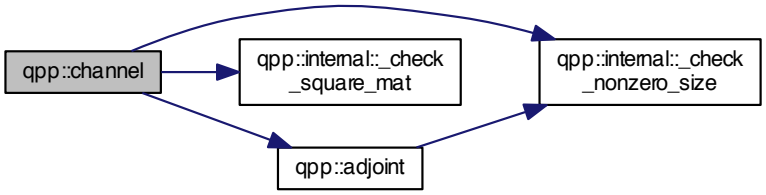
Parameters

rho	Eigen expression
Ks	Set of Kraus operators

Returns

Output density matrix after the action of the channel

Here is the call graph for this function:



6.1.2.5 template<typename Derived > cmat qpp::channel (const Eigen::MatrixBase< Derived > & *rho*, const std::vector< cmat > & *Ks*, const std::vector< std::size_t > & *subsys*, const std::vector< std::size_t > & *dims*)

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

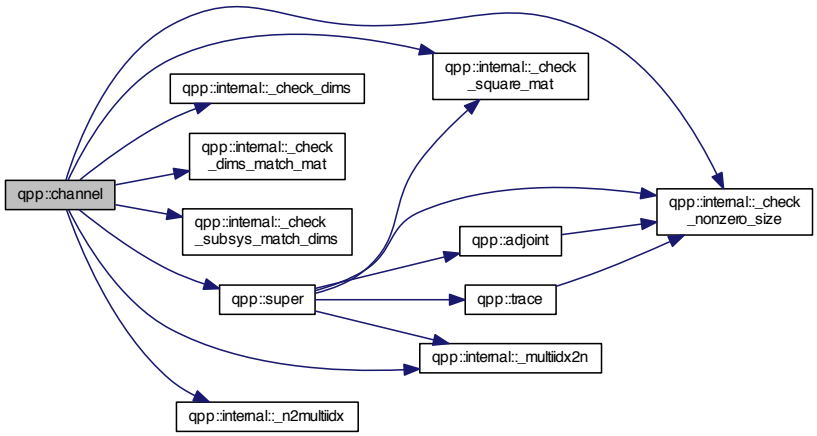
Parameters

rho	Eigen expression
Ks	Set of Kraus operators
subsys	Subsystems' indexes
dims	Dimensions of the multi-partite system

Returns

Output density matrix after the action of the channel

Here is the call graph for this function:



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

Choi matrix representation.

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

Note

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

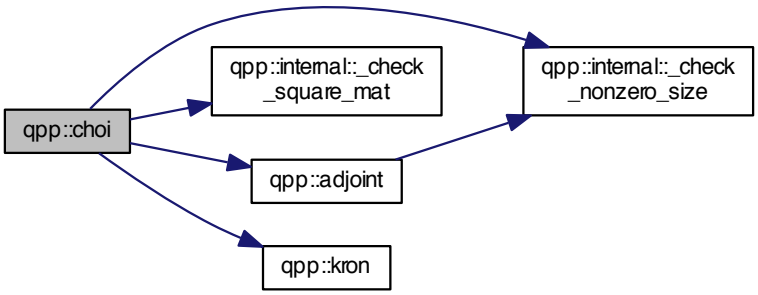
Parameters

Ks	Set of Kraus operators

Returns

Choi matrix representation

Here is the call graph for this function:



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

Extracts orthogonal Kraus operators from Choi matrix.

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

Note

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

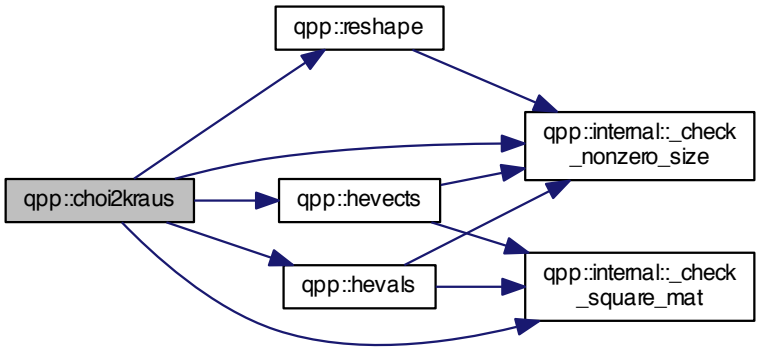
Parameters

A	Choi matrix

Returns

Set of Kraus operators

Here is the call graph for this function:



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

Commutator.

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

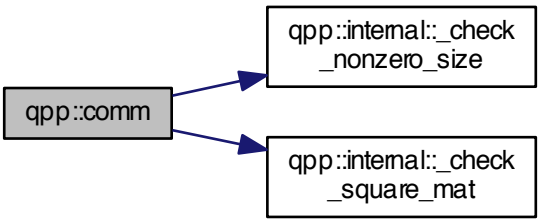
Parameters

A	Eigen expression
B	Eigen expression

Returns

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

Here is the call graph for this function:



6.1.2.9 std::vector<std::size_t> qpp::compperm (const std::vector< std::size_t > & *perm*, const std::vector< std::size_t > & *sigma*)

Compose permutations.

Parameters

perm	Permutation
sigma	Permutation

Returns

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

Here is the call graph for this function:

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

Complex conjugate.

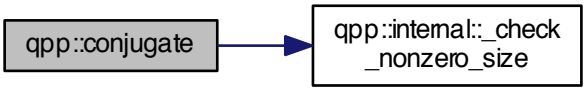
Parameters

A	Eigen expression

Returns

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

Here is the call graph for this function:



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

Matrix cos.

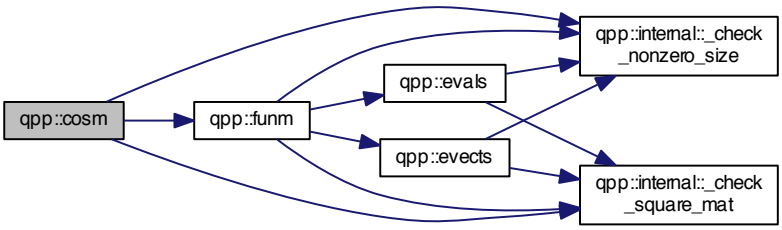
Parameters

A	Eigen expression

Returns

Matrix cosine of *A*

Here is the call graph for this function:



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

Functor.

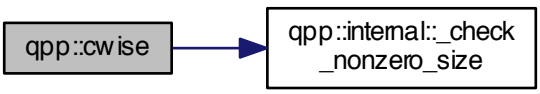
Parameters

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

Returns

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

Here is the call graph for this function:



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

Determinant.

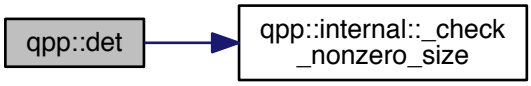
Parameters

A	Eigen expression

Returns

Determinant of *A*, as a dynamic matrix over the same scalar field
Returns $\pm\infty$ when the determinant overflows/underflows

Here is the call graph for this function:



6.1.2.14 template<typename T > void qpp::disp (const T & *x*, const std::string & *separator*, const std::string & *start* = "[", const std::string & *end* = "]", std::ostream & *os* = std::cout)

Displays a standard container that supports std::begin, std::end and forward iteration. Does not add a newline.

See also

qpp::displn()

Parameters

x	Container
separator	Separator
start	Left marking
end	Right marking
os	Output stream

6.1.2.15 template<typename T > void qpp::disp (const T * *x*, const std::size_t *n*, const std::string & *separator*, const std::string & *start* = "[", const std::string & *end* = "]", std::ostream & *os* = std::cout)

Displays a C-style array. Does not add a newline.

See also

qpp::displn()

Parameters

x	Pointer to the first element

n	Number of elements to be displayed
separator	Separator
start	Left marking
end	Right marking
os	Output stream

6.1.2.16 template<typename Derived > void qpp::disp (const Eigen::MatrixBase< Derived > & *A*, double *chop* = qpp::chop, std::ostream & *os* = std::cout)

Displays an Eigen expression in matrix friendly form. Does not add a new line.

See also

qpp::displn()

Parameters

A	Eigen expression
chop	Set to zero the elements smaller in absolute value than *chop*
os	Output stream

6.1.2.17 void qpp::disp (const cplx *z*, double *chop* = qpp::chop, std::ostream & *os* = std::cout)

Displays a number (implicitly converted to std::complex<double>) in friendly form. Does not add a new line.

See also

qpp::displn()

Parameters

z	Real/complex number
chop	Set to zero the elements smaller in absolute value than *chop*
os	Output stream

Here is the call graph for this function:

6.1.2.18 template<typename T > void qpp::displn (const T & *x*, const std::string & *separator*, const std::string & *start* = "[", const std::string & *end* = "]", std::ostream & *os* = std::cout)

Displays a standard container that supports std::begin, std::end and forward iteration. Adds a newline.

See also

qpp::disp()

Parameters

x	Container
separator	Separator
start	Left marking
end	Right marking
os	Output stream

Here is the call graph for this function:

6.1.2.19 template<typename T > void qpp::displn (const T * *x*, const std::size_t *n*, const std::string & *separator*, const std::string & *start* = "[", const std::string & *end* = "]", std::ostream & *os* = std::cout)

Displays a C-style array. Adds a newline.

See also

qpp::disp()

Parameters

x	Pointer to the first element
n	Number of elements to be displayed
separator	Separator
start	Left marking
end	Right marking
os	Output stream

Here is the call graph for this function:

6.1.2.20 template<typename Derived > void qpp::displn (const Eigen::MatrixBase< Derived > & *A*, double *chop* = qpp::chop, std::ostream & *os* = std::cout)

Displays an Eigen expression in matrix friendly form. Adds a newline.

See also

qpp::disp()

Parameters

A	Eigen expression
chop	Set to zero the elements smaller in absolute value than *chop*
os	Output stream

Here is the call graph for this function:

6.1.2.21 void qpp::displn (const cplx *z*, double *chop* = **qpp::chop**, std::ostream & *os* = `std::cout`)

Displays a number (implicitly converted to std::complex<double>) in friendly form. Adds a new line.

See also

qpp::disp()

Parameters

z	Real/complex number
chop	Set to zero the elements smaller in absolute value than *chop*
os	Output stream

Here is the call graph for this function:

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

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

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

See also

qpp::shannon()

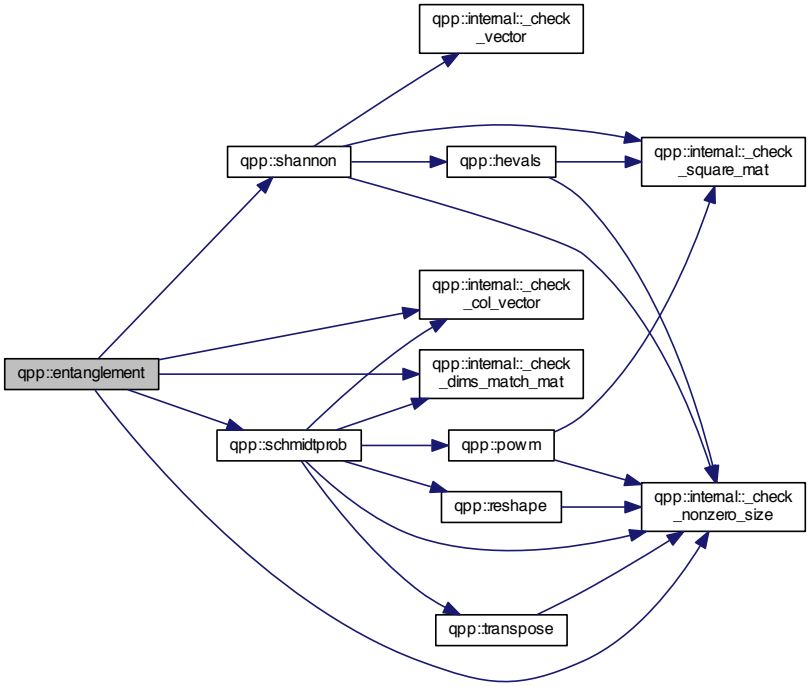
Parameters

A	Eigen expression
dims	Subsystems' dimensions

Returns

Entanglement, with the logarithm in base 2

Here is the call graph for this function:



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

Eigenvalues.

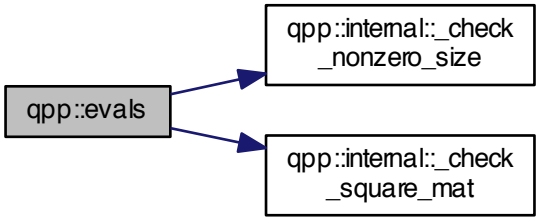
Parameters

A	Eigen expression

Returns

Eigenvalues of *A*, as a diagonal complex matrix

Here is the call graph for this function:



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

Eigenvectors.

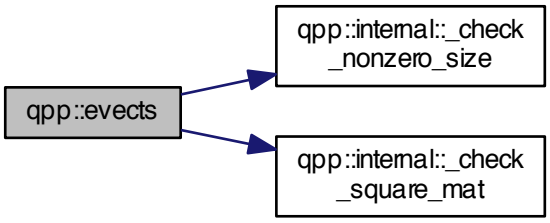
Parameters

A	Eigen expression

Returns

Eigenvectors of *A*, as columns of a complex matrix

Here is the call graph for this function:



6.1.2.25 template<typename Derived > DynMat<typename Derived::Scalar> qpp::expandout (const Eigen::MatrixBase< Derived > & *A*, std::size_t *pos*, const std::vector< std::size_t > & *dims*)

Expand out.

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

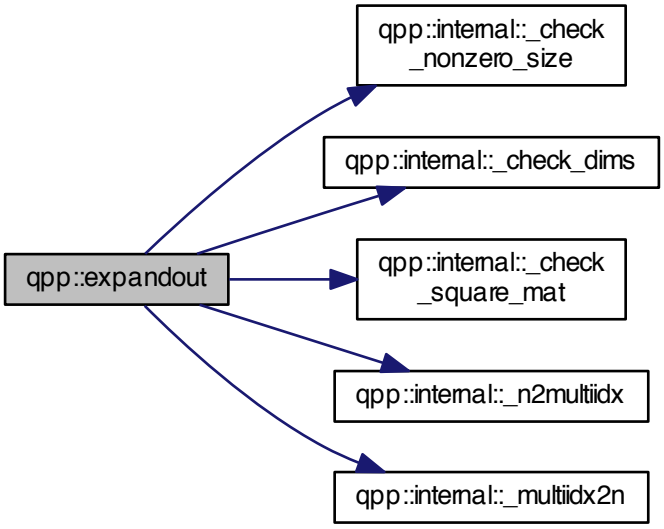
Parameters

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

Returns

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

Here is the call graph for this function:



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

Matrix exponential.

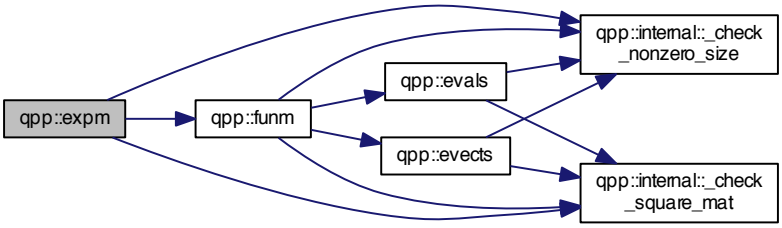
Parameters

A	Eigen expression

Returns

Matrix exponential of *A*

Here is the call graph for this function:



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

Functional calculus f(A)

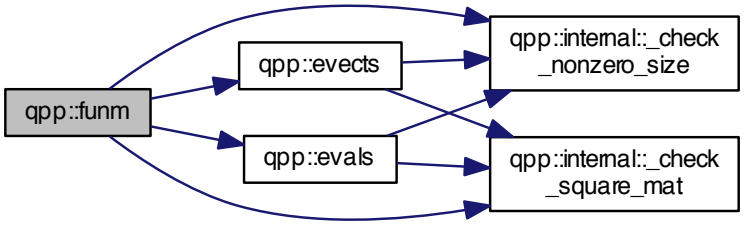
Parameters

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

Returns

$f(A)$

Here is the call graph for this function:



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

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

Uses *qpp::logdet()* to avoid overflows

See also

qpp::logdet()

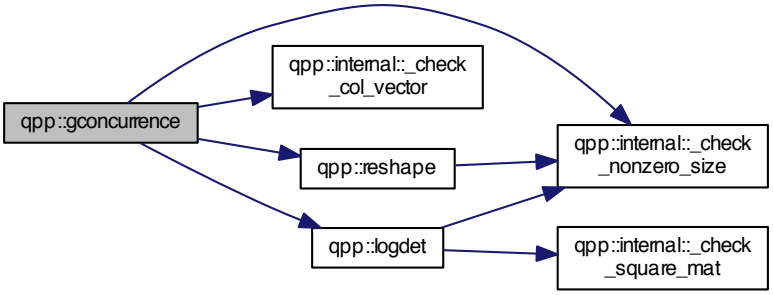
Parameters

A	Eigen expression
dims	Subsystems' dimensions

Returns

G-concurrence

Here is the call graph for this function:



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

Gram-Schmidt orthogonalization (std::vector overload)

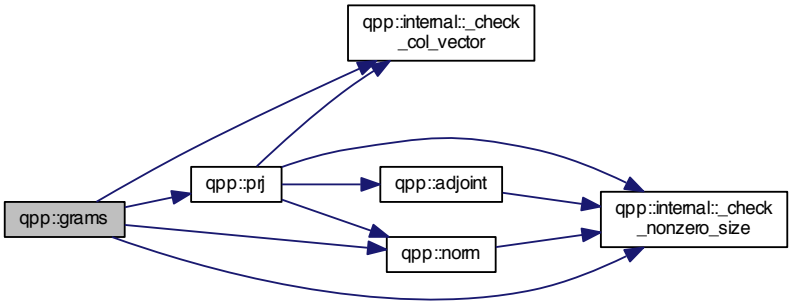
Parameters

Vs	std::vector of Eigen expressions as column vectors

Returns

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

Here is the call graph for this function:



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

Gram-Schmidt orthogonalization (std::initializer_list overload)

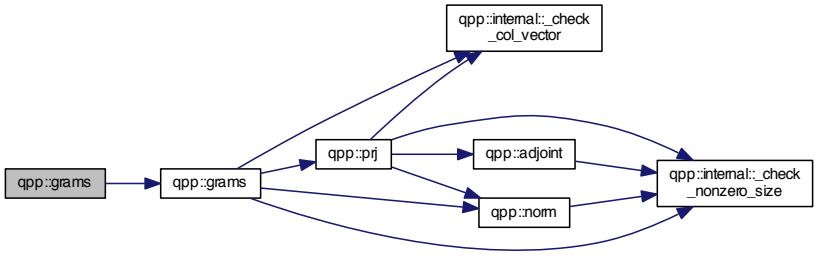
Parameters

Vs	std::initializer_list of Eigen expressions as column vectors

Returns

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

Here is the call graph for this function:



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

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

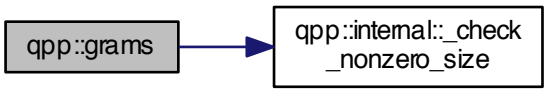
Parameters

A	Eigen expression, the input vectors are the columns of *A*

Returns

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

Here is the call graph for this function:



6.1.2.32 template<typename Derived > dmat qpp::hevals (const Eigen::MatrixBase< Derived > & *A*)

Hermitian eigenvalues.

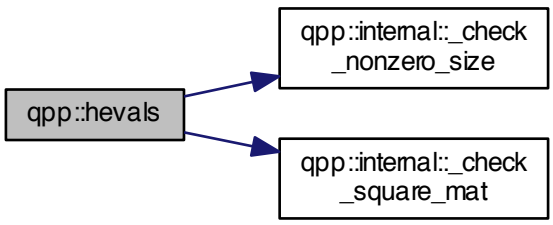
Parameters

A	Eigen expression

Returns

Eigenvalues of Hermitian *A*, as a diagonal real matrix

Here is the call graph for this function:



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

Hermitian eigenvectors.

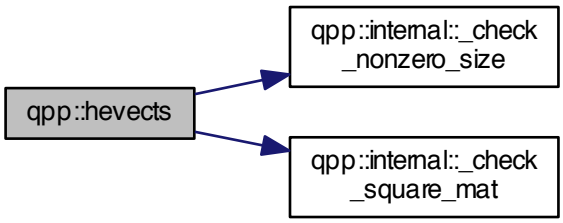
Parameters

A	Eigen expression

Returns

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

Here is the call graph for this function:



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

Inverse.

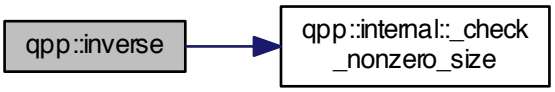
Parameters

A	Eigen expression

Returns

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

Here is the call graph for this function:



6.1.2.35 std::vector<std::size_t> qpp::invperm (const std::vector< std::size_t > & *perm*)

Inverse permutation.

Parameters

perm	Permutation

Returns

Inverse of the permutation *perm*

Here is the call graph for this function:

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

Kronecker product (variadic overload)

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

Parameters

head	Eigen expression

Returns

Its argument *head*

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

Kronecker product (variadic overload)

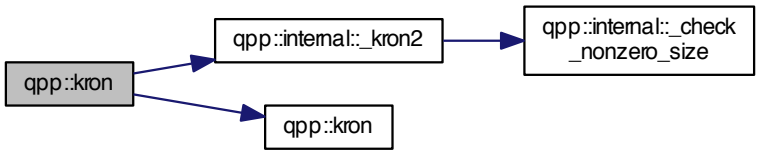
Parameters

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

Returns

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

Here is the call graph for this function:



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

Kronecker product (std::vector overload)

Parameters

As	std::vector of Eigen expressions

Returns

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

Here is the call graph for this function:

6.1.2.39 template<typename Derived > DynMat<typename Derived::Scalar> qpp::kron (const std::initializer_list< Derived > & *As*)

Kronecker product (std::initializer_list overload)

Parameters

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

Returns

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

Here is the call graph for this function:

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

Kronecker power.

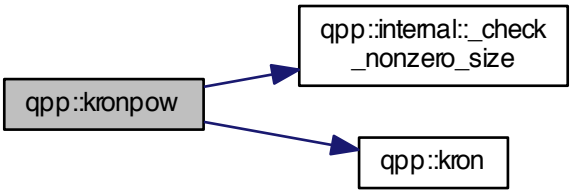
Parameters

A	Eigen expression
n	Non-negative integer

Returns

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

Here is the call graph for this function:



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

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

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

Example:

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

See also

qpp::loadMATLABmatrix()

Parameters

A	Eigen expression
fname	Output file name

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

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

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

6.1.2.43 template<> dmat qpp::loadMATLABmatrix (const std::string & *mat_file,* const std::string & *var_name*)

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

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

Example:

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

Note

If *var_name* is a complex matrix, only the real part is loaded

Parameters

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

Returns

Eigen double dynamic matrix (*qpp::dmat*)

6.1.2.44 template<> cmat qpp::loadMATLABmatrix (const std::string & *mat_file,* const std::string & *var_name*)

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

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

Example:

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

Parameters

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

Returns

Eigen complex dynamic matrix (*qpp::cmat*)

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

Logarithm of the determinant.

Especially useful when the determinant overflows/underflows

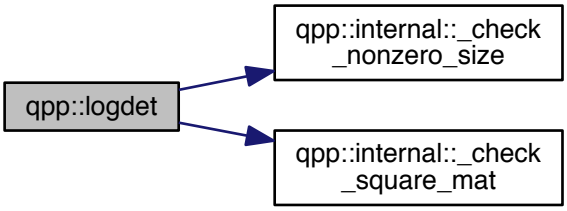
Parameters

A	Eigen expression

Returns

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

Here is the call graph for this function:



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

Matrix logarithm.

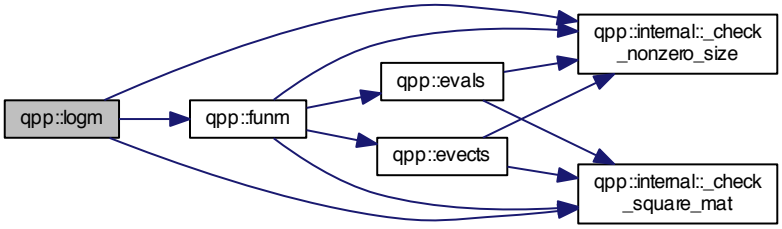
Parameters

A	Eigen expression

Returns

Matrix logarithm of *A*

Here is the call graph for this function:



6.1.2.47 ket qpp::mket (const std::vector< std::size_t > & *mask*)

Multi-partite qubit ket.

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

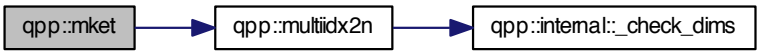
Parameters

mask	std::vector of 0's and 1's

Returns

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

Here is the call graph for this function:



6.1.2.48 ket qpp::mket (const std::vector< std::size_t > & *mask*, const std::vector< std::size_t > & *dims*)

Multi-partite qudit ket (different dimensions overload)

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

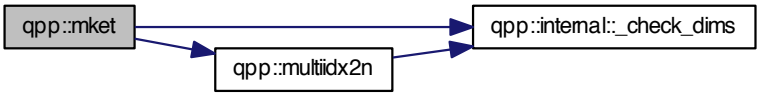
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.49 ket qpp::mket (const std::vector< std::size_t > & *mask*, std::size_t *d*)

Multi-partite qudit ket (same dimensions overload)

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

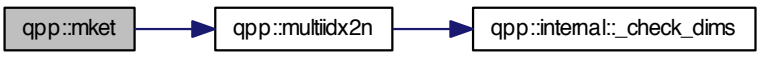
Parameters

mask	std::vector of non-negative integers
d	Subsystems' dimension

Returns

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

Here is the call graph for this function:



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

Multi-index to non-negative integer index.

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

Parameters

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

Returns

Non-negative integer index

Here is the call graph for this function:

6.1.2.51 std::vector<std::size_t> qpp::n2multiidx (std::size_t *n*, const std::vector< std::size_t > & *dims*)

Non-negative integer index to multi-index.

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

Parameters

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

Returns

Multi-index of the same size as *dims*

Here is the call graph for this function:

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

Trace norm.

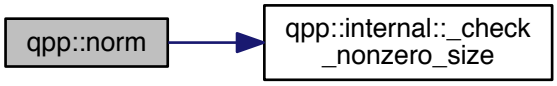
Parameters

A	Eigen expression

Returns

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

Here is the call graph for this function:



6.1.2.53 std::complex<double> qpp::omega (std::size_t *D*)

D-th root of unity.

Parameters

D	Non-negative integer

Returns

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

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

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

Example:

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

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

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

Example:

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

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

Matrix power.

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

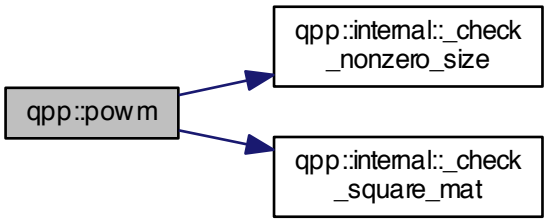
Parameters

A	Eigen expression
n	Non-negative integer

Returns

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

Here is the call graph for this function:



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

Projector.

Normalized projector onto state vector

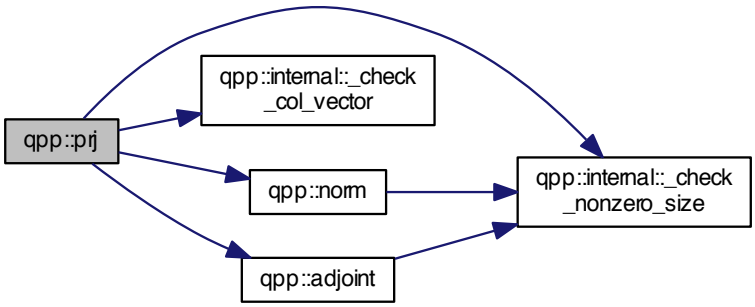
Parameters

V	Eigen expression

Returns

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

Here is the call graph for this function:



6.1.2.58 template<typename Derived > DynMat<typename Derived::Scalar> qpp::ptrace (const Eigen::MatrixBase< Derived > & *A*, const std::vector< std::size_t > & *subsys*, const std::vector< std::size_t > & *dims*)

Partial trace.

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

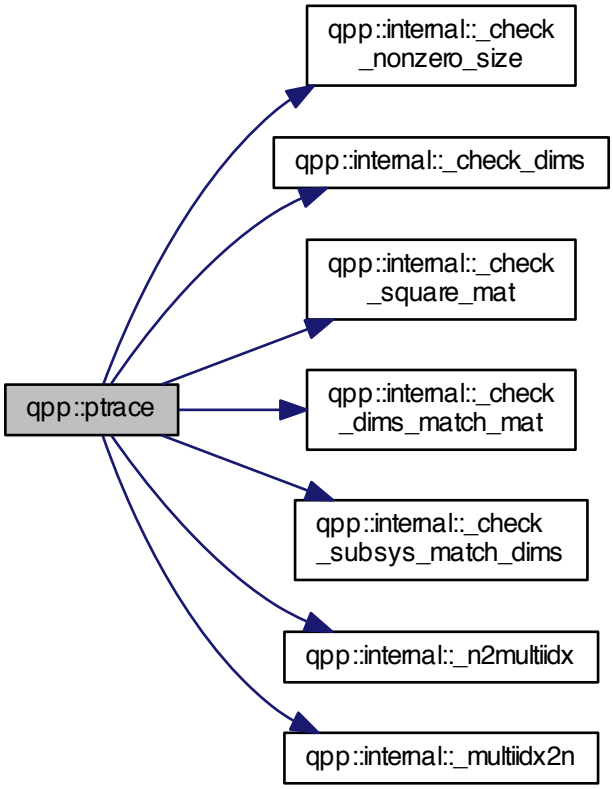
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.59 template<typename Derived > DynMat<typename Derived::Scalar> qpp::ptrace1 (const Eigen::MatrixBase< Derived > & *A*, const std::vector< std::size_t > & *dims*)

Partial trace.

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

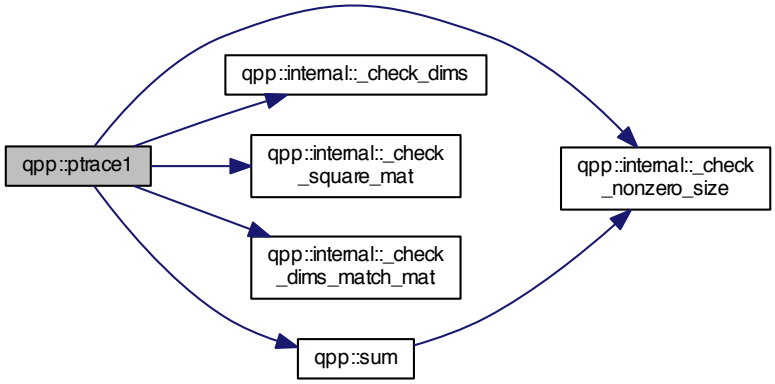
Parameters

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

Returns

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

Here is the call graph for this function:



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

Partial trace.

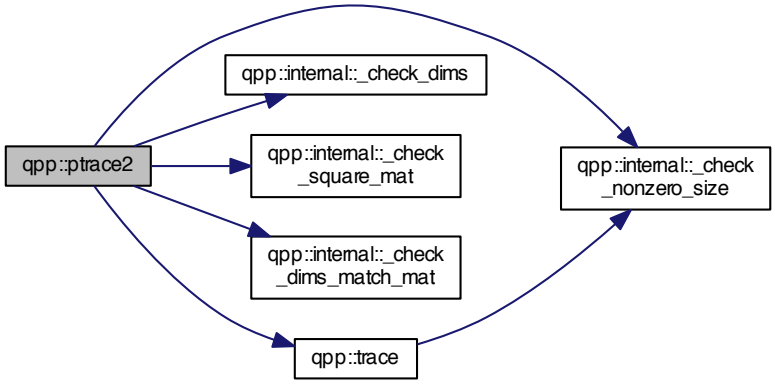
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.61 template<typename Derived > DynMat<typename Derived::Scalar> qpp::ptranspose (const Eigen::MatrixBase< Derived > & *A*, const std::vector< std::size_t > & *subsys*, const std::vector< std::size_t > & *dims*)

Partial transpose.

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

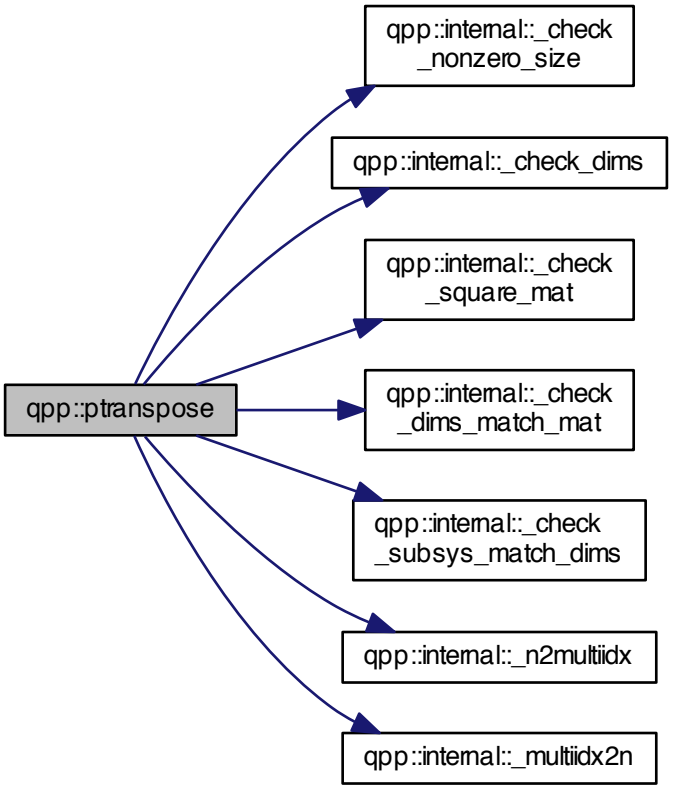
Parameters

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

Returns

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

Here is the call graph for this function:



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

Quantum mutual information between 2 subsystems of a composite system.

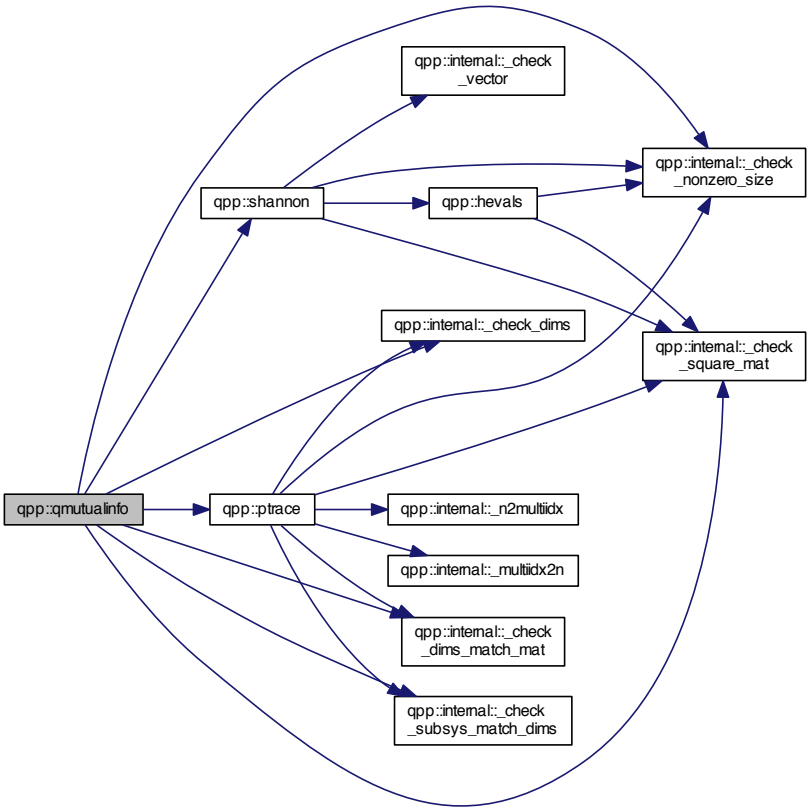
Parameters

A	Eigen expression
subsysA	Indexes of the first subsystem
subsysB	Indexes of the second subsystem
dims	Subsystems' dimensions

Returns

Mutual information between the 2 subsystems

Here is the call graph for this function:



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

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

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

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

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

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

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

Example:

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

Parameters

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

Returns

Random real matrix

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

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

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

Example:

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

Parameters

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

Returns

Random complex matrix

Here is the call graph for this function:

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

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

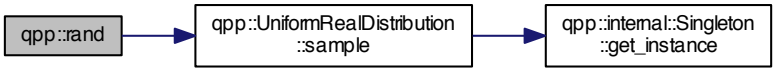
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.67 cmat qpp::randH (std::size_t *D*)

Generates a random Hermitian matrix.

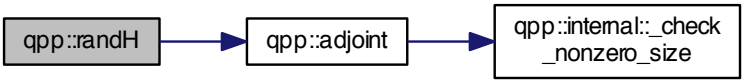
Parameters

D	Dimension of the Hilbert space

Returns

Random Hermitian matrix

Here is the call graph for this function:



6.1.2.68 int qpp::randint (int *a* = `std::numeric_limits<int>::min()`, int *b* = `std::numeric_limits<int>::max()`)

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

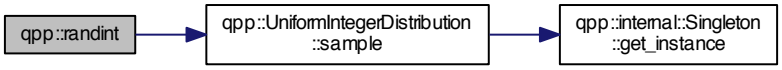
Parameters

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

Returns

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

Here is the call graph for this function:



6.1.2.69 ket qpp::randket (std::size_t *D*)

Generates a random normalized ket (pure state vector)

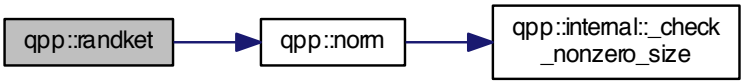
Parameters

D	Dimension of the Hilbert space

Returns

Random normalized ket

Here is the call graph for this function:



6.1.2.70 std::vector<cmat> qpp::randkraus (std::size_t *n*, std::size_t *D*)

Generates a set of random Kraus operators.

Note

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

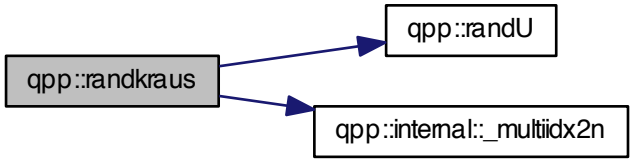
Parameters

n	Number of Kraus operators
D	Dimension of the Hilbert space

Returns

Set of *n* Kraus operators satisfying the closure condition

Here is the call graph for this function:



6.1.2.71 template<typename Derived > Derived qpp::randn (std::size_t *rows*, std::size_t *cols*, double *mean* = 0, double *sigma* = 1)

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

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

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

6.1.2.72 template<> dmat qpp::randn (std::size_t *rows*, std::size_t *cols*, double *mean*, double *sigma*)

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

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

Example:

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

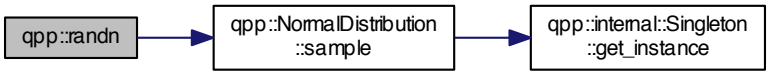
Parameters

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

Returns

Random real matrix

Here is the call graph for this function:



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

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

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

Example:

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

Parameters

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

Returns

Random complex matrix

Here is the call graph for this function:

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

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

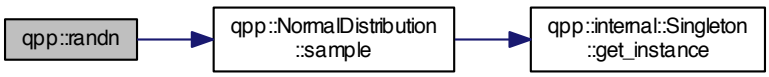
Parameters

mean	Mean
sigma	Standard deviation

Returns

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

Here is the call graph for this function:



6.1.2.75 std::vector<std::size_t> qpp::randperm (std::size_t *n*)

Generates a random uniformly distributed permutation.

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

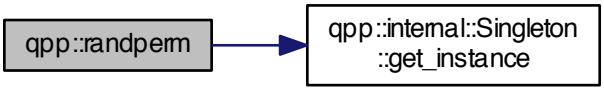
Parameters

n	Size of the permutation

Returns

Random permutation of size *n*

Here is the call graph for this function:



6.1.2.76 cmat qpp::randrho (std::size_t *D*)

Generates a random density matrix.

Parameters

D	Dimension of the Hilbert space

Returns

Random density matrix

6.1.2.77 cmat qpp::randU (std::size_t *D*)

Generates a random unitary matrix.

Parameters

D	Dimension of the Hilbert space

Returns

Random unitary

6.1.2.78 cmat qpp::randV (std::size_t *Din*, std::size_t *Dout*)

Generates a random isometry matrix.

Parameters

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

Returns

Random isometry matrix

Here is the call graph for this function:

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

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

Parameters

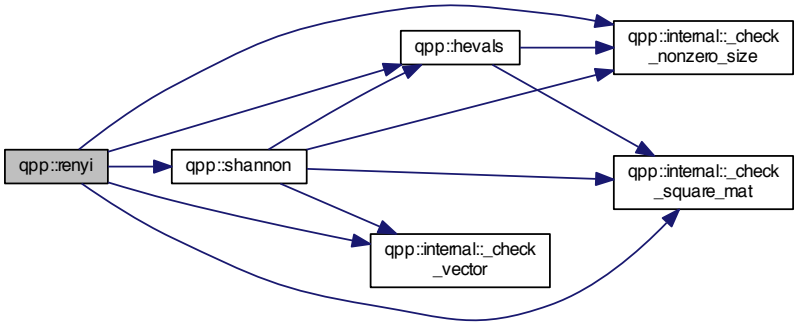
alpha	Non-negative real number

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

Returns

Renyi- α entropy, with the logarithm in base 2

Here is the call graph for this function:



6.1.2.80 template<typename Derived > double qpp::renyi_inf (const Eigen::MatrixBase< Derived > & *A*)

Renyi- ∞ entropy (min entropy) of the probability distribution/density matrix *A*.

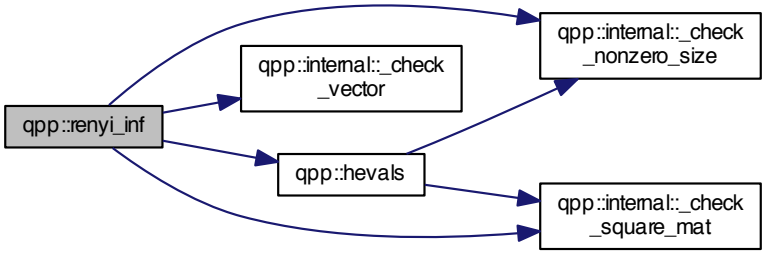
Parameters

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

Returns

Renyi- ∞ entropy (min entropy), with the logarithm in base 2

Here is the call graph for this function:



6.1.2.81 template<typename Derived > DynMat<typename Derived::Scalar> qpp::reshape (const Eigen::MatrixBase< Derived > & *A*, std::size_t *rows*, std::size_t *cols*)

Reshape.

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

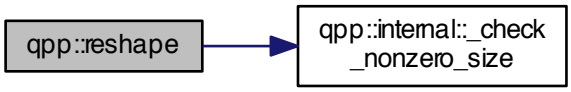
Parameters

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

Returns

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

Here is the call graph for this function:



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

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

See also

qpp::saveMATLABmatrix()

Parameters

A	Eigen expression
fname	Output file name

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

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

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

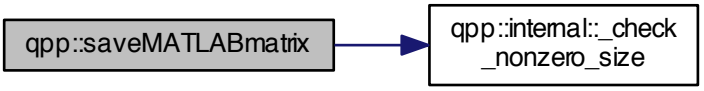
6.1.2.84 template<> void qpp::saveMATLABmatrix (const Eigen::MatrixBase< dmat > & *A*, const std::string & *mat_file*, const std::string & *var_name*, const std::string & *mode*)

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

Parameters

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

Here is the call graph for this function:



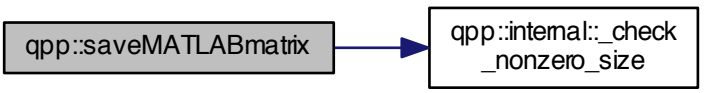
6.1.2.85 template<> void qpp::saveMATLABmatrix (const Eigen::MatrixBase< cmat > & *A*, const std::string & *mat_file*, const std::string & *var_name*, const std::string & *mode*)

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

Parameters

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

Here is the call graph for this function:



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

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

Note

The sum of the squares of the Schmidt coefficients equals 1

See also

qpp::schmidtprob()

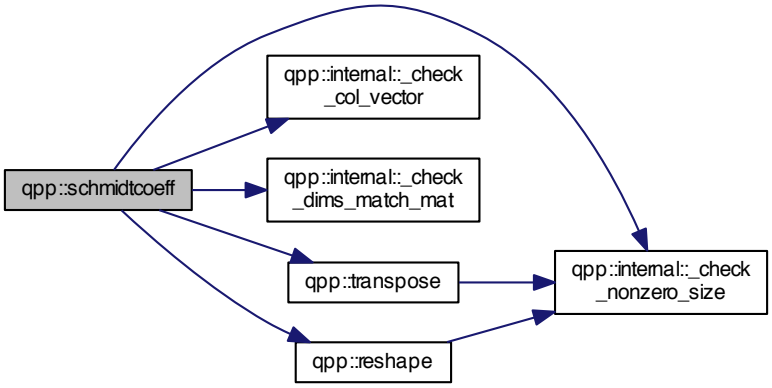
Parameters

A	Eigen expression
dims	Subsystems' dimensions

Returns

Schmidt coefficients of *A*, as a complex dynamic matrix, with the Schmidt coefficients on the diagonal

Here is the call graph for this function:



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

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

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

See also

qpp::schmidtcoeff()

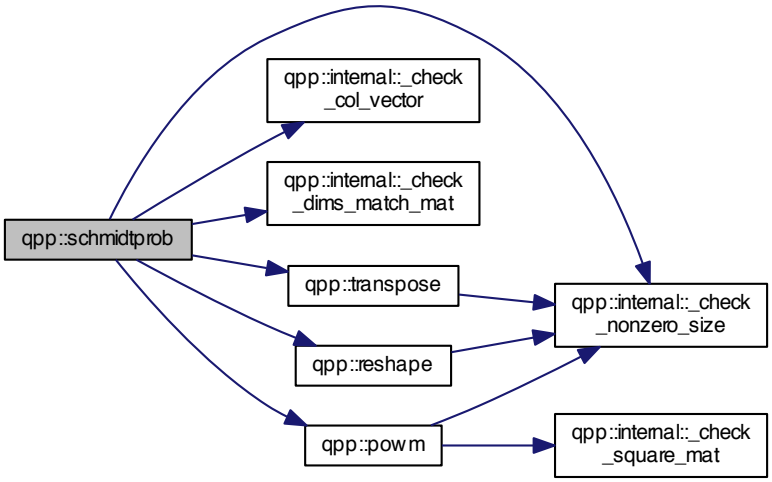
Parameters

A	Eigen expression
dims	Subsystems' dimensions

Returns

Schmidt probabilites of *A*, as a complex dynamic matrix, with the Schmidt probabilities on the diagonal

Here is the call graph for this function:



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

Schmidt basis on Alice's side.

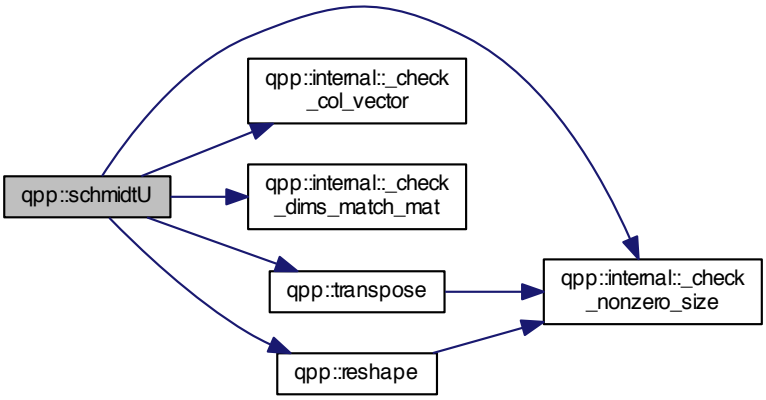
Parameters

A	Eigen expression
dims	Subsystems' dimensions

Returns

Unitary matrix U representing the Schmidt basis on Alice's side, as a complex dynamic matrix, acting on the computational basis as $U|j\rangle = |\bar{j}\rangle$ (Schmidt vector)

Here is the call graph for this function:



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

Schmidt basis on Bob's side.

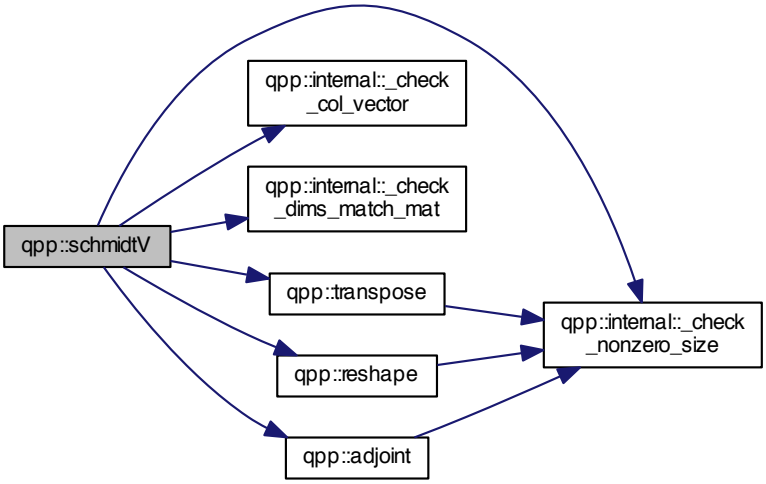
Parameters

A	Eigen expression
dims	Subsystems' dimensions

Returns

Unitary matrix V representing the Schmidt basis on Bob's side, as a complex dynamic matrix, acting on the computational basis as $V|j\rangle = |\bar{j}\rangle$ (Schmidt vector)

Here is the call graph for this function:



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

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

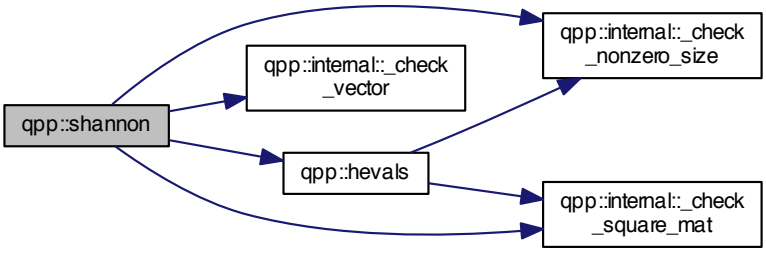
Parameters

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

Returns

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

Here is the call graph for this function:



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

Matrix sin.

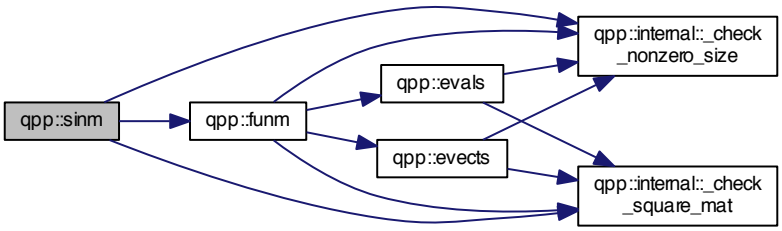
Parameters

A	Eigen expression

Returns

Matrix sine of *A*

Here is the call graph for this function:



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

Matrix power.

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

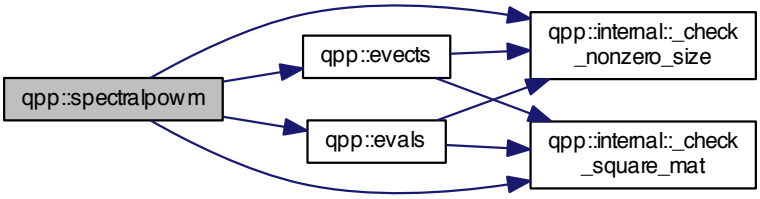
Parameters

A	Eigen expression
z	Complex number

Returns

Matrix power A^z

Here is the call graph for this function:



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

Matrix square root.

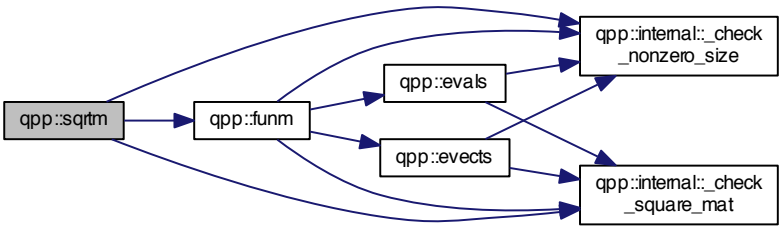
Parameters

A	Eigen expression

Returns

Matrix square root of *A*

Here is the call graph for this function:



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

Element-wise sum.

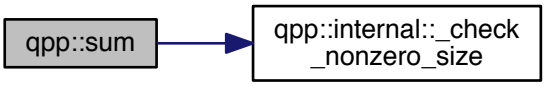
Parameters

A	Eigen expression

Returns

Element-wise sum of *A*, as a dynamic matrix over the same scalar field

Here is the call graph for this function:



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

Superoperator matrix representation.

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

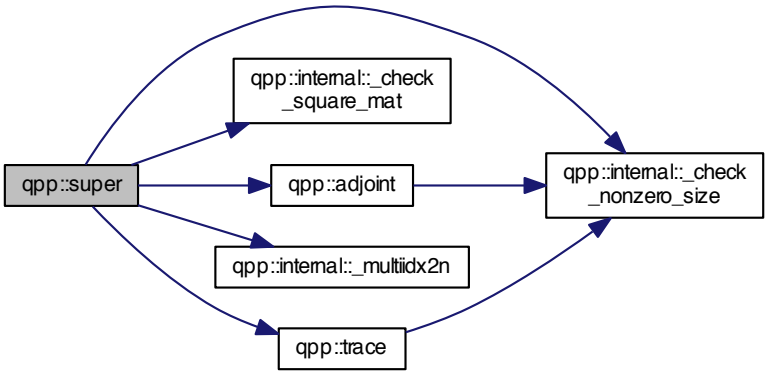
Parameters

Ks	Set of Kraus operators

Returns

Superoperator matrix representation

Here is the call graph for this function:



6.1.2.96 template<typename Derived > DynMat<typename Derived::Scalar> qpp::syspermute (const Eigen::MatrixBase< Derived > & *A*, const std::vector< std::size_t > & *perm*, const std::vector< std::size_t > & *dims*)

System permutation.

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

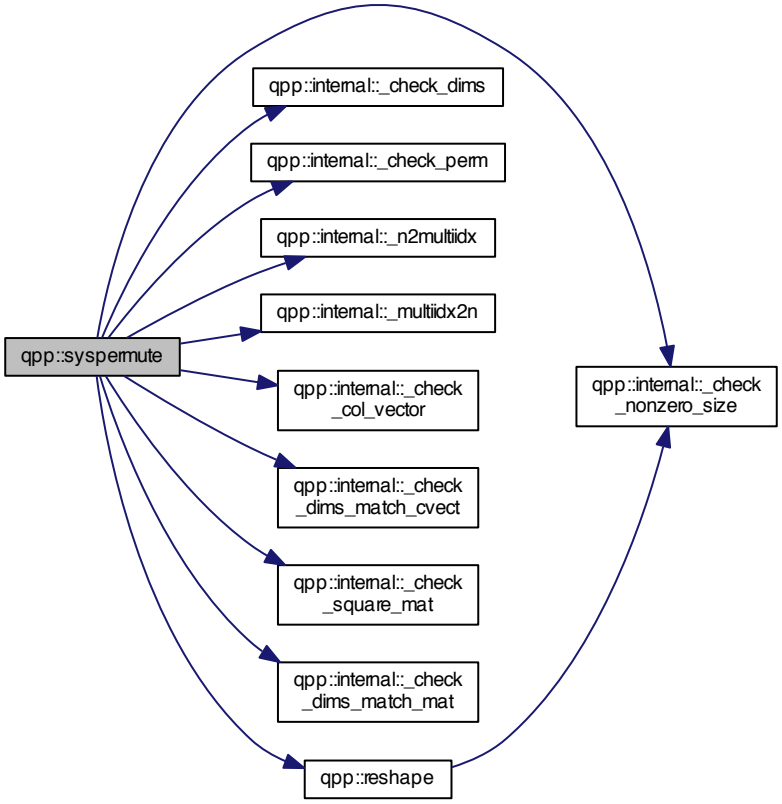
Parameters

A	Eigen expression
perm	Permutation
dims	Subsystems' dimensions

Returns

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

Here is the call graph for this function:



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

Trace.

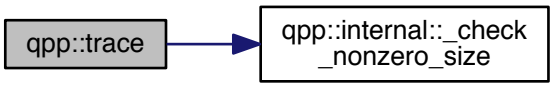
Parameters

A	Eigen expression

Returns

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

Here is the call graph for this function:



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

Transpose.

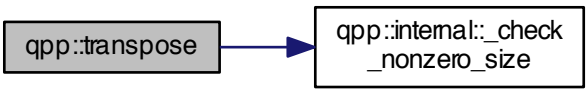
Parameters

A	Eigen expression

Returns

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

Here is the call graph for this function:



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

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

.

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

Parameters

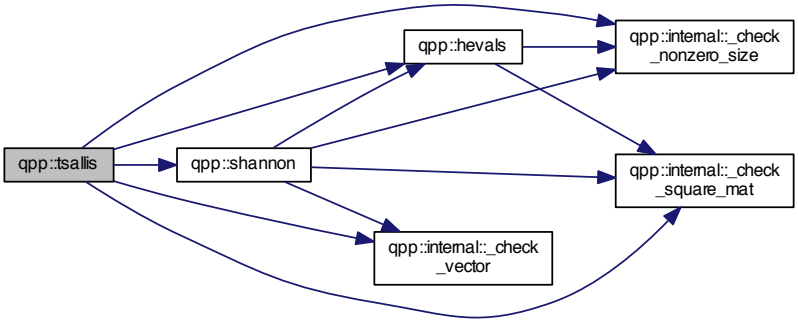
alpha	Non-negative real number

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

Returns

Renyi- α entropy, with the logarithm in base 2

Here is the call graph for this function:



6.1.3 Variable Documentation

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

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

6.1.3.2 constexpr double qpp::ee = 2.71828182845904523536028747135266249 97

Base of natural logarithm, e.

6.1.3.3 constexpr double qpp::eps = 1e-12

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

Example:

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

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

qpp::Gates const Singleton

Initializes the gates, see the class *qpp::Gates*

6.1.3.5 constexpr std::size_t qpp::maxn = 64

Maximum number of qubits.

Used internally to statically allocate arrays (for speed reasons)

6.1.3.6 constexpr double qpp::pi = 3.14159265358979323846264338327950288 4

π

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

qpp::RandomDevices Singleton

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

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

qpp::States const Singleton

Initializes the states, see the class *qpp::States*

6.2 qpp::internal Namespace Reference

Classes

- class Singleton

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

Functions

- void _n2multiidx (std::size_t n, std::size_t numdims, const std::size_t *dims, std::size_t *result)
- std::size_t _multiidx2n (const std::size_t *midx, std::size_t numdims, const std::size_t *dims)
- template<typename Derived >
  bool _check_square_mat (const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >
  bool _check_vector (const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >
  bool _check_row_vector (const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >
  bool _check_col_vector (const Eigen::MatrixBase< Derived > &A)
- template<typename T >
  bool _check_nonzero_size (const T &x)
- bool _check_dims (const std::vector< std::size_t > &dims)
- template<typename Derived >
  bool _check_dims_match_mat (const std::vector< std::size_t > &dims, const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >
  bool _check_dims_match_cvect (const std::vector< std::size_t > &dims, const Eigen::MatrixBase< Derived > &V)
- template<typename Derived >
  bool _check_dims_match_rvect (const std::vector< std::size_t > &dims, const Eigen::MatrixBase< Derived > &V)
- bool _check_eq_dims (const std::vector< std::size_t > &dims, std::size_t dim)
- bool _check_subsys_match_dims (const std::vector< std::size_t > &subsys, const std::vector< std::size_t > &dims)
- bool _check_perm (const std::vector< std::size_t > &perm)

- template<typename Derived1 , typename Derived2 >
  DynMat< typename Derived1::Scalar > _kron2 (const Eigen::MatrixBase< Derived1 > &A, const Eigen::↩
  MatrixBase< Derived2 > &B)

- template<typename T >
  void variadic_vector_emplace (std::vector< T > &)

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

6.2.1 Detailed Description

Internal implementation details, do not modify/use the functions/classes unless you know what you are doing

6.2.2 Function Documentation

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

6.2.2.2 bool qpp::internal::_check_dims (const std::vector< std::size_t > & *dims*)

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

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

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

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

6.2.2.7 template<typename T > bool qpp::internal::_check_nonzero_size (const T & *x*)

6.2.2.8 bool qpp::internal::_check_perm (const std::vector< std::size_t > & *perm*)

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

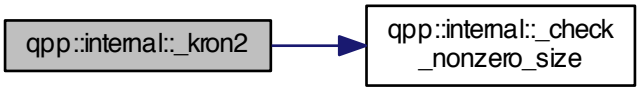
6.2.2.10 template<typename Derived > bool qpp::internal::_check_square_mat (const Eigen::MatrixBase< Derived > & *A*)

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

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

6.2.2.13 template<typename Derived1 , typename Derived2 > DynMat<typename Derived1::Scalar> qpp::internal::_kron2 (const Eigen::MatrixBase< Derived1 > & *A*, const Eigen::MatrixBase< Derived2 > & *B*)

Here is the call graph for this function:



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

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

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

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

Here is the call graph for this function:

Chapter 7

Class Documentation

7.1 qpp::DiscreteDistribution< T > Class Template Reference

`#include <stat.h>`

Public Member Functions

- template<typename InputIterator >
  DiscreteDistribution (InputIterator first, InputIterator last)
- DiscreteDistribution (std::initializer_list< double > weights)
- DiscreteDistribution (std::vector< double > weights)
- T sample ()
- std::vector< double > probabilities () const

Protected Attributes

- std::discrete_distribution< T > _d

7.1.1 Constructor & Destructor Documentation

7.1.1.1 template<typename T = std::size_t> template<typename InputIterator > qpp::DiscreteDistribution< T >::DiscreteDistribution (InputIterator *first,* InputIterator *last*) [inline]

7.1.1.2 template<typename T = std::size_t> qpp::DiscreteDistribution< T >::DiscreteDistribution (std::initializer_list< double > *weights*) [inline]

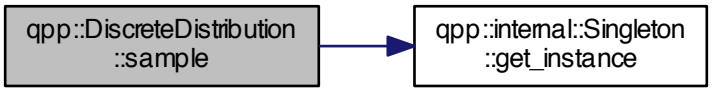
7.1.1.3 template<typename T = std::size_t> qpp::DiscreteDistribution< T >::DiscreteDistribution (std::vector< double > *weights*) [inline]

7.1.2 Member Function Documentation

7.1.2.1 template<typename T = std::size_t> std::vector<double> qpp::DiscreteDistribution< T >::probabilities ( ) const [inline]

7.1.2.2 template<typename T = std::size_t> T qpp::DiscreteDistribution< T >::sample ( ) [inline]

Here is the call graph for this function:



7.1.3 Member Data Documentation

7.1.3.1 template<typename T = std::size_t> std::discrete_distribution<T> qpp::DiscreteDistribution< T >::_d [protected]

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

- include/classes/stat.h

7.2 qpp::DiscreteDistributionAbsSquare< T > Class Template Reference

`#include <stat.h>`

Public Member Functions

- template<typename InputIterator >
  DiscreteDistributionAbsSquare (InputIterator first, InputIterator last)
- DiscreteDistributionAbsSquare (std::initializer_list< cplx > amplitudes)
- DiscreteDistributionAbsSquare (std::vector< cplx > amplitudes)
- template<typename Derived >
  DiscreteDistributionAbsSquare (const Eigen::MatrixBase< Derived > &V)
- T sample ()
- std::vector< double > probabilities () const

Protected Member Functions

- template<typename InputIterator >
  std::vector< double > cplx2weights (InputIterator first, InputIterator last) const

Protected Attributes

- std::discrete_distribution< T > _d

7.2.1 Constructor & Destructor Documentation

7.2.1.1 template<typename T = std::size_t> template<typename InputIterator > qpp::DiscreteDistribution↩ AbsSquare< T >::DiscreteDistributionAbsSquare (InputIterator *first,* InputIterator *last*) [inline]

7.2.1.2 template<typename T = std::size_t> qpp::DiscreteDistributionAbsSquare< T >::DiscreteDistributionAbsSquare (std::initializer_list< cplx > *amplitudes*) [inline]

7.2.1.3 template<typename T = std::size_t> qpp::DiscreteDistributionAbsSquare< T >::DiscreteDistributionAbsSquare (std::vector< cplx > *amplitudes*) [inline]

7.2.1.4 template<typename T = std::size_t> template<typename Derived > qpp::DiscreteDistributionAbsSquare< T >::DiscreteDistributionAbsSquare (const Eigen::MatrixBase< Derived > & *V*) [inline]

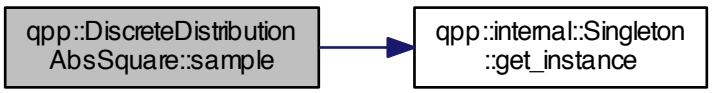
7.2.2 Member Function Documentation

7.2.2.1 template<typename T = std::size_t> template<typename InputIterator > std::vector<double> qpp::DiscreteDistributionAbsSquare< T >::cplx2weights (InputIterator *first,* InputIterator *last*) const [inline], [protected]

7.2.2.2 template<typename T = std::size_t> std::vector<double> qpp::DiscreteDistributionAbsSquare< T >::probabilities ( ) const [inline]

7.2.2.3 template<typename T = std::size_t> T qpp::DiscreteDistributionAbsSquare< T >::sample ( ) [inline]

Here is the call graph for this function:



7.2.3 Member Data Documentation

7.2.3.1 template<typename T = std::size_t> std::discrete_distribution<T> qpp::DiscreteDistributionAbsSquare< T >::_d [protected]

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

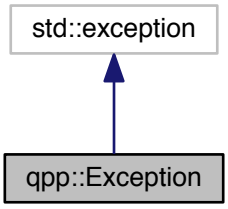
- include/classes/stat.h

7.3 qpp::Exception Class Reference

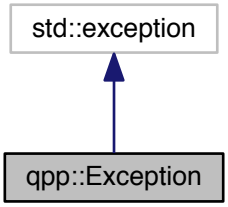
Generates custom exceptions, used when validating function parameters.

`#include <exception.h>`

Inheritance diagram for qpp::Exception:



Collaboration diagram for qpp::Exception:



Public Types

- enum Type {
  Type::UNKNOWN_EXCEPTION = 1, Type::ZERO_SIZE, Type::MATRIX_NOT_SQUARE, Type::MATRIX_NOT_CVECTOR,
  Type::MATRIX_NOT_RVECTOR, Type::MATRIX_NOT_VECTOR, Type::MATRIX_NOT_SQUARE_OR_CVECTOR, Type::MATRIX_NOT_SQUARE_OR_RVECTOR,
  Type::MATRIX_NOT_SQUARE_OR_VECTOR, Type::DIMS_INVALID, Type::DIMS_NOT_EQUAL, Type::DIMS_MISMATCH_MATRIX,
  Type::DIMS_MISMATCH_CVECTOR, Type::DIMS_MISMATCH_RVECTOR, Type::DIMS_MISMATCH_VECTOR, Type::SUBSYS_MISMATCH_DIMS,
  Type::PERM_INVALID, Type::NOT_QUBIT_GATE, Type::NOT_QUBIT_SUBSYS, Type::NOT_BIPARTITE,
  Type::OUT_OF_RANGE, Type::TYPE_MISMATCH, Type::UNDEFINED_TYPE, Type::CUSTOM_EXCEPTION }

  Exception types, add more exceptions here if needed.

Public Member Functions

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

  Constructs an exception.
- Exception (const std::string &where, const std::string &custom)

  Constructs an exception.

- virtual const char * what () const noexceptoverride

  Overrides std::exception::what()

Private Member Functions

- std::string _construct_exception_msg ()

  Constructs the exception's description from its type.

Private Attributes

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

7.3.1 Detailed Description

Generates custom exceptions, used when validating function parameters.

Customize this class if more exceptions are needed

7.3.2 Member Enumeration Documentation

7.3.2.1 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. Unknown exception

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

MATRIX_NOT_SQUARE MATRIX_NOT_SQUARE. Eigen::Matrix is not square

MATRIX_NOT_CVECTOR MATRIX_NOT_CVECTOR. Eigen::Matrix is not a column vector

MATRIX_NOT_RVECTOR MATRIX_NOT_RVECTOR. Eigen::Matrix is not a row vector

MATRIX_NOT_VECTOR MATRIX_NOT_VECTOR. Eigen::Matrix is not a row/column vector

MATRIX_NOT_SQUARE_OR_CVECTOR MATRIX_NOT_SQUARE_OR_CVECTOR. Eigen::Matrix is not square nor a column vector

MATRIX_NOT_SQUARE_OR_RVECTOR MATRIX_NOT_SQUARE_OR_RVECTOR. Eigen::Matrix is not square nor a row vector

MATRIX_NOT_SQUARE_OR_VECTOR MATRIX_NOT_SQUARE_OR_VECTOR. Eigen::Matrix is not square nor a row/column vector

DIMS_INVALID DIMS_INVALID. std::vector<std::size_t> representing the dimensions has zero size or contains zeros

DIMS_NOT_EQUAL DIMS_NOT_EQUAL. std::vector<std::size_t> representing the dimensions contains non-equal elements

DIMS_MISMATCH_MATRIX DIMS_MISMATCH_MATRIX. Product of the dimenisons' std::vector<std↩ ::size_t> is not equal to the number of rows of Eigen::Matrix (assumed to be square)

DIMS_MISMATCH_CVECTOR DIMS_MISMATCH_CVECTOR. Product of the dimenisons' std::vector<std::size_t> is not equal to the number of cols of Eigen::Matrix (assumed to be a column vector)

DIMS_MISMATCH_RVECTOR DIMS_MISMATCH_RVECTOR. Product of the dimenisons' std::vector<std::size_t> is not equal to the number of cols of Eigen::Matrix (assumed to be a row vector)

DIMS_MISMATCH_VECTOR DIMS_MISMATCH_VECTOR. Product of the dimenisons' std::vector<std::size_t> is not equal to the number of cols of Eigen::Matrix (assumed to be a row/column vector)

SUBSYS_MISMATCH_DIMS SUBSYS_MISMATCH_DIMS. std::vector<std::size_t> representing the subsystems' labels has duplicatates, or has entries that are larger than the size of the std::vector<std::size_t> representing the dimensions

PERM_INVALID PERM_INVALID. Invalid std::vector<std::size_t> permutation

NOT_QUBIT_GATE NOT_QUBIT_GATE. Eigen::Matrix is not 2 x 2

NOT_QUBIT_SUBSYS NOT_QUBIT_SUBSYS. Subsystems are not 2-dimensional

NOT_BIPARTITE NOT_BIPARTITE. std::vector<std::size_t> representing the dimensions has size different from 2

OUT_OF_RANGE OUT_OF_RANGE. Parameter out of range

TYPE_MISMATCH TYPE_MISMATCH. Types do not match (i.e. Matrix<double> vs Matrix<cplx>)

UNDEFINED_TYPE UNDEFINED_TYPE. Templated function not defined for this type

CUSTOM_EXCEPTION CUSTOM_EXCEPTION. Custom exception, user must provide a custom message

7.3.3 Constructor & Destructor Documentation

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

Constructs an exception.

Parameters

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

Here is the call graph for this function:

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

Constructs an exception.

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

Parameters

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

Here is the call graph for this function:

7.3.4 Member Function Documentation

7.3.4.1 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

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

Overrides std::exception::what()

Returns

Exception's description

7.3.5 Member Data Documentation

7.3.5.1 std::string qpp::Exception::_custom [private]

7.3.5.2 std::string qpp::Exception::_msg [private]

7.3.5.3 Type qpp::Exception::_type [private]

7.3.5.4 std::string qpp::Exception::_where [private]

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

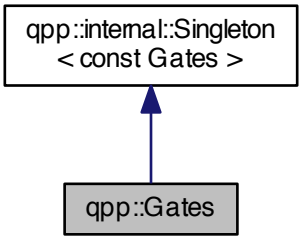
- include/classes/exception.h

7.4 qpp::Gates Class Reference

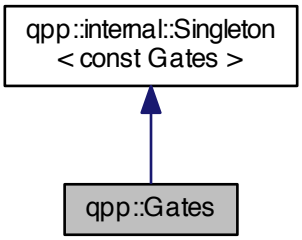
Singleton class that implements most commonly used gates.

`#include <gates.h>`

Inheritance diagram for qpp::Gates:



Collaboration diagram for qpp::Gates:



Public Member Functions

- cmat Rn (double theta, std::vector< double > n) const

  Rotation of theta about the 3-dimensional real unit vector n.
- cmat Zd (std::size_t D) const

  Generalized Z gate for qudits.
- cmat Fd (std::size_t D) const

  Fourier transform gate for qudits.
- cmat Xd (std::size_t D) const

  Generalized X gate for qudits.
- template<typename Derived = Eigen::MatrixXcd>
  Derived Id (std::size_t D) const

  Identity gate.
- template<typename Derived1 , typename Derived2 >
  DynMat< typename Derived1::Scalar > applyCTRL (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< std::size_t > &ctrl, const std::vector< std::size_t > &subsys, std::size_t n, std::size_t d=2) const

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

- template<typename Derived1 , typename Derived2 >
  DynMat< typename Derived1::Scalar > apply (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< std::size_t > &subsys, const std::vector< std::size_t > &dims) const

  Applies the gate A to the part subsys of a multipartite state vector or density matrix.
- template<typename Derived >
  DynMat< typename Derived::Scalar > CTRL (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &ctrl, const std::vector< std::size_t > &subsys, std::size_t n, std::size_t d=2) const

  Generates the multipartite multiple-controlled-A gate in matrix form.

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

7.4.1 Detailed Description

Singleton class that implements most commonly used gates.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 qpp::Gates::Gates ( ) [inline], [private]

Initializes the gates.

7.4.3 Member Function Documentation

7.4.3.1 template<typename Derived1 , typename Derived2 > DynMat<typename Derived1::Scalar> qpp::Gates::apply (const Eigen::MatrixBase< Derived1 > & *state*, const Eigen::MatrixBase< Derived2 > & *A*, const std::vector< std::size_t > & *subsys*, const std::vector< std::size_t > & *dims*) const [inline]

Applies the gate *A* to the part *subsys* of a multipartite state vector or density matrix.

Note

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

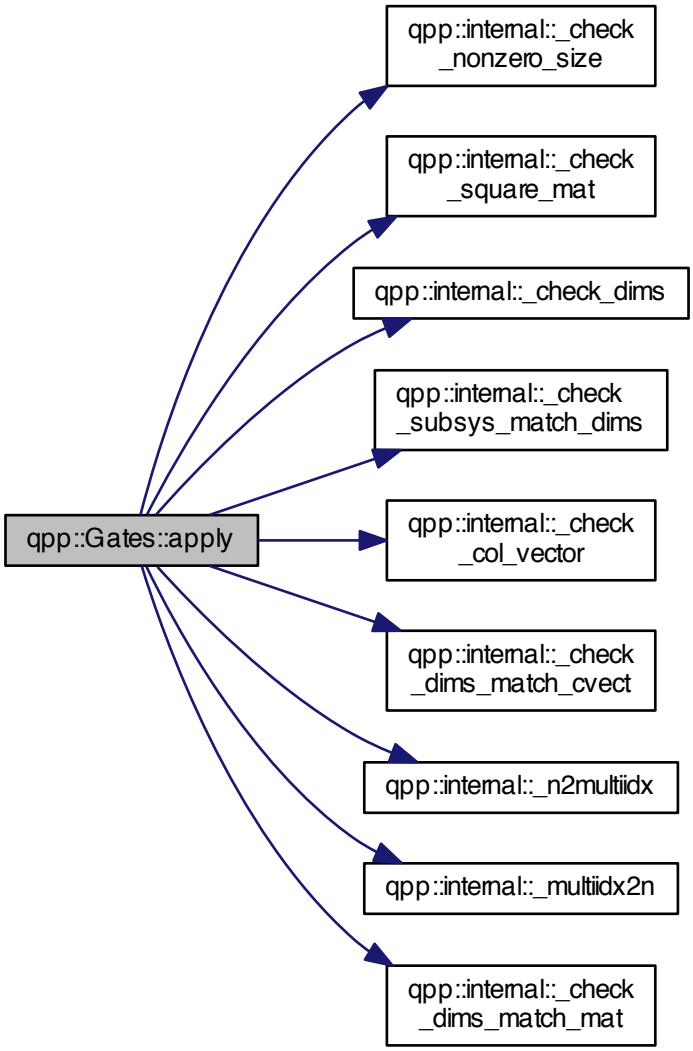
Parameters

state	Eigen expression
A	Eigen expression
subsys	Subsystem indexes where the gate *A* is applied
dims	Local dimensions of all local Hilbert spaces (can be different)

Returns

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

Here is the call graph for this function:



7.4.3.2 template<typename Derived1 , typename Derived2 > DynMat<typename Derived1::Scalar> qpp::Gates::applyCTRL (const Eigen::MatrixBase< Derived1 > & *state*, const Eigen::MatrixBase< Derived2 > & *A*, const std::vector< std::size_t > & *ctrl*, const std::vector< std::size_t > & *subsys*, std::size_t *n*, std::size_t *d* = 2) const [inline]

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

Note

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

Parameters

state	Eigen expression
A	Eigen expression
ctrl	Control subsystem indexes
subsys	Subsystem indexes where the gate *A* is applied
n	Total number of subsystes
d	Local dimensions of all local Hilbert spaces (must all be equal)

Returns

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

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

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

Note

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

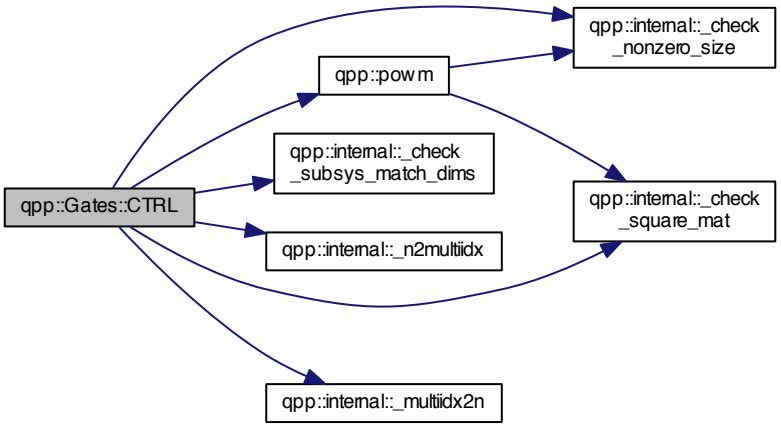
Parameters

A	Eigen expression
ctrl	Control subsystem indexes
subsys	Subsystem indexes where the gate *A* is applied
n	Total number of subsystes
d	Local dimensions of all local Hilbert spaces (must all be equal)

Returns

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

Here is the call graph for this function:



7.4.3.4 cmat qpp::Gates::Fd (std::size_t *D*) const [inline]

Fourier transform gate for qudits.

Note

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

Parameters

D	Dimension of the Hilbert space

Returns

Fourier transform gate for qudits

Here is the call graph for this function:

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

Identity gate.

Note

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

Parameters

D	Dimension of the Hilbert space

Returns

Identity gate

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

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

Parameters

theta	Rotation angle

n	3-dimensional real unit vector

Returns

Rotation gate

7.4.3.7 cmat qpp::Gates::Xd (std::size_t *D*) const [inline]

Generalized X gate for qudits.

Note

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

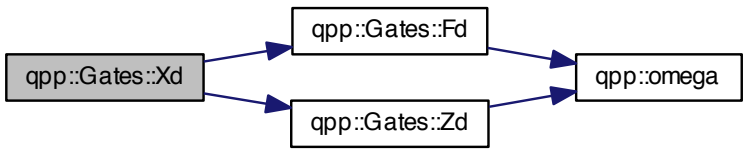
Parameters

D	Dimension of the Hilbert space

Returns

Generalized X gate for qudits

Here is the call graph for this function:



7.4.3.8 cmat qpp::Gates::Zd (std::size_t *D*) const [inline]

Generalized Z gate for qudits.

Note

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

Parameters

D	Dimension of the Hilbert space

Returns

Generalized Z gate for qudits

Here is the call graph for this function:

7.4.4 Friends And Related Function Documentation

7.4.4.1 friend class internal::Singleton< const Gates > [friend]

7.4.5 Member Data Documentation

7.4.5.1 cmat qpp::Gates::CNOTab { cmat::Identity(4, 4) }

Controlled-NOT control target gate.

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

Controlled-NOT target control gate.

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

Controlled-Phase gate.

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

Fredkin gate.

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

Hadamard gate.

7.4.5.6 cmat qpp::Gates::Id2 { cmat::Identity(2, 2) }

Identity gate.

7.4.5.7 cmat qpp::Gates::S { cmat::Zero(2, 2) }

S gate.

7.4.5.8 cmat qpp::Gates::SWAP { cmat::Identity(4, 4) }

SWAP gate.

7.4.5.9 cmat qpp::Gates::T { cmat::Zero(2, 2) }

T gate.

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

Toffoli gate.

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

Pauli Sigma-X gate.

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

Pauli Sigma-Y gate.

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

Pauli Sigma-Z gate.

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

- include/classes/gates.h

7.5 qpp::NormalDistribution< T > Class Template Reference

Template light wrapper around std::normal_distribution<>, generates normally-distributed random numbers.

`#include <stat.h>`

Public Member Functions

- NormalDistribution (T mean=0, T sigma=1)

  Constructs an instance with mean mean and standard deviation sigma.

- T sample ()

  Samples from the distribution.

Protected Attributes

- std::normal_distribution< T > _d

  Wrapper around this.

7.5.1 Detailed Description

template<typename T = double>class qpp::NormalDistribution< T >

Template light wrapper around std::normal_distribution<>, generates normally-distributed random numbers.

Note

Can change the scalar type from double (default) by explicitly specifying the template parameter

7.5.2 Constructor & Destructor Documentation

7.5.2.1 template<typename T = double> qpp::NormalDistribution< T >::NormalDistribution (T *mean* = 0, T *sigma* = 1) [inline]

Constructs an instance with mean *mean* and standard deviation *sigma*.

Parameters

mean	Mean
sigma	Standard deviation

7.5.3 Member Function Documentation

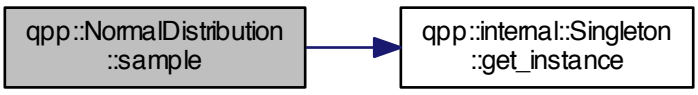
7.5.3.1 template<typename T = double> T qpp::NormalDistribution< T >::sample ( ) [inline]

Samples from the distribution.

Returns

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

Here is the call graph for this function:



7.5.4 Member Data Documentation

7.5.4.1 template<typename T = double> std::normal_distribution<T> qpp::NormalDistribution< T >::_d [protected]

Wrapper around this.

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

- include/classes/stat.h

7.6 qpp::Qudit Class Reference

`#include <qudit.h>`

Public Member Functions

- Qudit (const cmat &rho=States::get_instance().pz0)
- std::size_t measure (const cmat &U, bool destructive=false)
- std::size_t measure (bool destructive=false)
- cmat getRho () const
- std::size_t getD () const

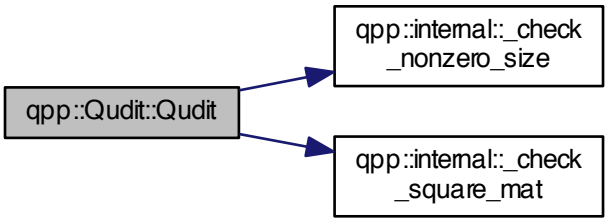
Private Attributes

- cmat _rho
- std::size_t _D

7.6.1 Constructor & Destructor Documentation

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

Here is the call graph for this function:



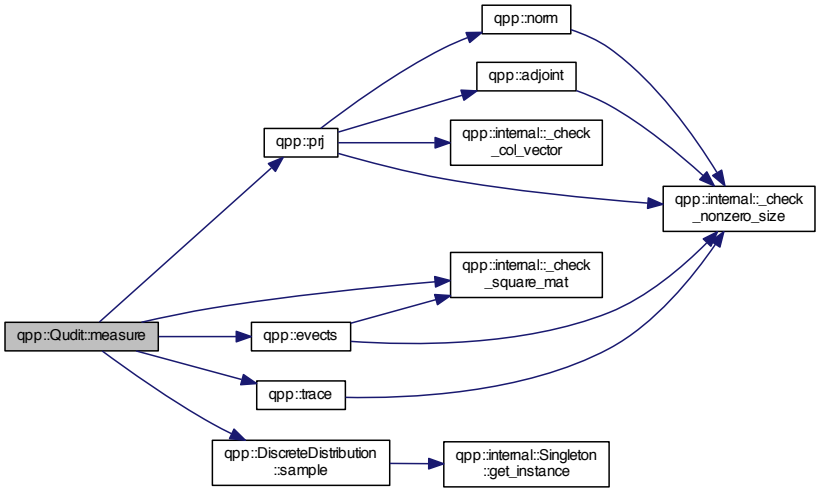
7.6.2 Member Function Documentation

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

7.6.2.2 cmat qpp::Qudit::getRho ( ) const [inline]

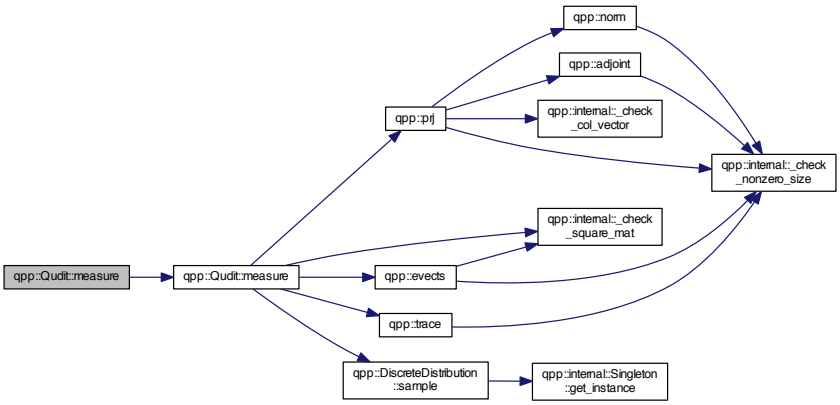
7.6.2.3 std::size_t qpp::Qudit::measure (const cmat & *U,* bool *destructive* = `false`) [inline]

Here is the call graph for this function:



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

Here is the call graph for this function:



7.6.3 Member Data Documentation

7.6.3.1 std::size_t qpp::Qudit::_D [private]

7.6.3.2 cmat qpp::Qudit::_rho [private]

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

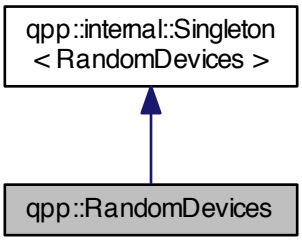
- include/classes/qudit.h

7.7 qpp::RandomDevices Class Reference

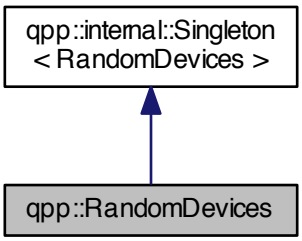
Singeleton class that manages the source of randomness in the library.

`#include <randevs.h>`

Inheritance diagram for qpp::RandomDevices:



Collaboration diagram for qpp::RandomDevices:



Public Attributes

- std::mt19937 _rng

  Mersenne twister random number generator engine.

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

7.7.1 Detailed Description

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

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

7.7.2 Constructor & Destructor Documentation

7.7.2.1 qpp::RandomDevices::RandomDevices ( ) [inline], [private]

Initializes and seeds the random number generators.

7.7.3 Friends And Related Function Documentation

7.7.3.1 friend class internal::Singleton< RandomDevices > [friend]

7.7.4 Member Data Documentation

7.7.4.1 std::random_device qpp::RandomDevices::_rd [private]

used to seed std::mt19937 _rng

7.7.4.2 std::mt19937 qpp::RandomDevices::_rng

Mersenne twister random number generator engine.

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

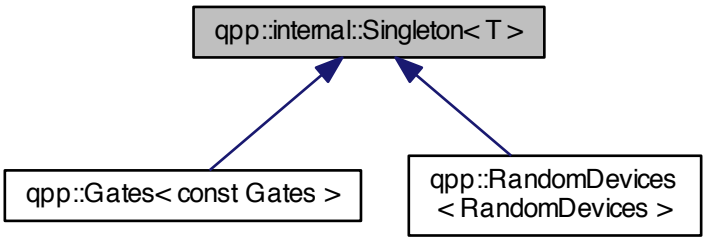
- include/classes/randevs.h

7.8 qpp::internal::Singleton< T > Class Template Reference

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

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

Inheritance diagram for qpp::internal::Singleton< T >:



Static Public Member Functions

- static T & get_instance ()

Protected Member Functions

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

7.8.1 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::Gates*, *qpp::RandomDevices*, *qpp::States* or *qpp.h* for real world examples of usage.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 template<typename T> qpp::internal::Singleton< T >::Singleton ( ) [protected], [default]

7.8.2.2 template<typename T> virtual qpp::internal::Singleton< T >::~Singleton ( ) [protected], [virtual], [default]

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

7.8.3 Member Function Documentation

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

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

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

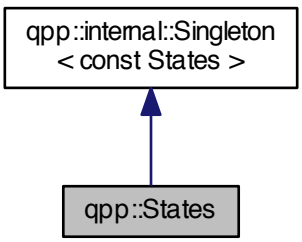
- include/classes/singleton.h

7.9 qpp::States Class Reference

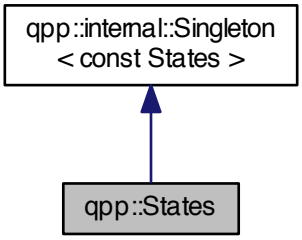
Singleton class that implements most commonly used states.

`#include <states.h>`

Inheritance diagram for qpp::States:



Collaboration diagram for qpp::States:



Public Attributes

- ket x0 { ket::Zero(2) }

  Pauli Sigma-X 0-eigenstate |+>
- ket x1 { ket::Zero(2) }

  Pauli Sigma-X 1-eigenstate |->
- ket y0 { ket::Zero(2) }

  Pauli Sigma-Y 0-eigenstate.
- ket y1 { ket::Zero(2) }

  Pauli Sigma-Y 1-eigenstate.
- ket z0 { ket::Zero(2) }

  Pauli Sigma-Z 0-eigenstate |0>
- ket z1 { ket::Zero(2) }

  Pauli Sigma-Z 1-eigenstate |1>
- cmat px0 { cmat::Zero(2, 2) }

  Projector onto the Pauli Sigma-X 0-eigenstate |+><+|.
- cmat px1 { cmat::Zero(2, 2) }

  Projector onto the Pauli Sigma-X 1-eigenstate |-><-|.
- cmat py0 { cmat::Zero(2, 2) }

  Projector onto the Pauli Sigma-Y 0-eigenstate.
- cmat py1 { cmat::Zero(2, 2) }

  Projector onto the Pauli Sigma-Y 1-eigenstate.
- cmat pz0 { cmat::Zero(2, 2) }

  Projector onto the Pauli Sigma-Z 0-eigenstate |0><0|.
- cmat pz1 { cmat::Zero(2, 2) }

  Projector onto the Pauli Sigma-Z 1-eigenstate |1><1|.
- ket b00 { ket::Zero(4) }

  Bell-00 state (following the convention in Nielsen and Chuang)
- ket b01 { ket::Zero(4) }

  Bell-01 state (following the convention in Nielsen and Chuang)
- ket b10 { ket::Zero(4) }

  Bell-10 state (following the convention in Nielsen and Chuang)
- ket b11 { ket::Zero(4) }

  Bell-11 state (following the convention in Nielsen and Chuang)
- cmat pb00 { cmat::Zero(4, 4) }

Projector onto the Bell-00 state.

- cmat pb01 { cmat::Zero(4, 4) }

  Projector onto the Bell-01 state.

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

  Projector onto the Bell-10 state.

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

  Projector onto the Bell-11 state.

- ket GHZ { ket::Zero(8) }

  GHZ state.

- ket W { ket::Zero(8) }

  W state.

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

  Projector onto the GHZ state.

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

  Projector onto the W state.

Private Member Functions

- States ()

Friends

- class internal::Singleton< const States >

Additional Inherited Members

7.9.1 Detailed Description

Singleton class that implements most commonly used states.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 qpp::States::States ( ) [inline], [private]

Initialize the states

7.9.3 Friends And Related Function Documentation

7.9.3.1 friend class internal::Singleton< const States > [friend]

7.9.4 Member Data Documentation

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

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

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

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

7.9.4.3 ket qpp::States::b10 { ket::Zero(4) }

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

7.9.4.4 ket qpp::States::b11 { ket::Zero(4) }

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

7.9.4.5 ket qpp::States::GHZ { ket::Zero(8) }

GHZ state.

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

Projector onto the Bell-00 state.

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

Projector onto the Bell-01 state.

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

Projector onto the Bell-10 state.

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

Projector onto the Bell-11 state.

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

Projector onto the GHZ state.

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

Projector onto the W state.

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

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

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

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

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

Projector onto the Pauli Sigma-Y 0-eigenstate.

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

Projector onto the Pauli Sigma-Y 1-eigenstate.

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

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

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

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

7.9.4.18 ket qpp::States::W { ket::Zero(8) }

W state.

7.9.4.19 ket qpp::States::x0 { ket::Zero(2) }

Pauli Sigma-X 0-eigenstate $|+>$

7.9.4.20 ket qpp::States::x1 { ket::Zero(2) }

Pauli Sigma-X 1-eigenstate $|->$

7.9.4.21 ket qpp::States::y0 { ket::Zero(2) }

Pauli Sigma-Y 0-eigenstate.

7.9.4.22 ket qpp::States::y1 { ket::Zero(2) }

Pauli Sigma-Y 1-eigenstate.

7.9.4.23 ket qpp::States::z0 { ket::Zero(2) }

Pauli Sigma-Z 0-eigenstate $|0>$

7.9.4.24 ket qpp::States::z1 { ket::Zero(2) }

Pauli Sigma-Z 1-eigenstate $|1>$

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

- include/classes/states.h

7.10 qpp::Timer Class Reference

Measures time.

`#include <timer.h>`

Public Member Functions

- Timer ()

  Constructs an instance with the current time as the starting point.
- void tic ()

  Resets the chronometer.
- void 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

- std::ostream & operator<< (std::ostream &os, const Timer &rhs)

  Overload for std::ostream operators.

7.10.1 Detailed Description

Measures time.

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

7.10.2 Constructor & Destructor Documentation

7.10.2.1 qpp::Timer::Timer ( ) [inline]

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

7.10.3 Member Function Documentation

7.10.3.1 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()

7.10.3.2 void qpp::Timer::tic ( ) [inline]

Resets the chronometer.

Resets the starting/ending point to the current time

7.10.3.3 void qpp::Timer::toc ( ) [inline]

Stops the chronometer.

Set the current time as the ending point

7.10.4 Friends And Related Function Documentation

7.10.4.1 std::ostream& operator<< (std::ostream & *os*, const Timer & *rhs*) [friend]

Overload for std::ostream operators.

Parameters

os	Output stream
rhs	Timer instance

Returns

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

7.10.5 Member Data Documentation

7.10.5.1 std::chrono::steady_clock::time_point qpp::Timer::_end [protected]

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

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

- include/classes/timer.h

7.11 qpp::UniformIntegerDistribution< T > Class Template Reference

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

Public Member Functions

- UniformIntegerDistribution (T a=std::numeric_limits< T >::min(), T b=std::numeric_limits< T >::max())
- T sample ()

Protected Attributes

- std::uniform_int_distribution< T > _d

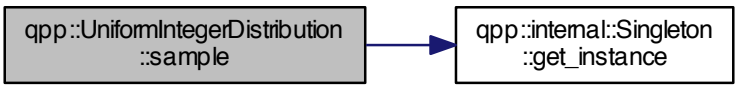
7.11.1 Constructor & Destructor Documentation

7.11.1.1 template<typename T = int> qpp::UniformIntegerDistribution< T >::UniformIntegerDistribution (T *a* = std::numeric_limits<T>::min(), T *b* = std::numeric_limits<T>::max()) [inline]

7.11.2 Member Function Documentation

7.11.2.1 template<typename T = int> T qpp::UniformIntegerDistribution< T >::sample ( ) [inline]

Here is the call graph for this function:



7.11.3 Member Data Documentation

7.11.3.1 template<typename T = int> std::uniform_int_distribution<T> qpp::UniformIntegerDistribution< T >::_d [protected]

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

- include/classes/stat.h

7.12 qpp::UniformRealDistribution< T > Class Template Reference

`#include <stat.h>`

Public Member Functions

- UniformRealDistribution (T a=0, T b=1)
- T sample ()

Protected Attributes

- std::uniform_real_distribution< T > _d

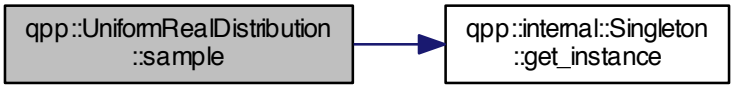
7.12.1 Constructor & Destructor Documentation

7.12.1.1 template<typename T = double> qpp::UniformRealDistribution< T >::UniformRealDistribution (T *a* = 0, T *b* = 1) [inline]

7.12.2 Member Function Documentation

7.12.2.1 template<typename T = double> T qpp::UniformRealDistribution< T >::sample ( ) [inline]

Here is the call graph for this function:



7.12.3 Member Data Documentation

7.12.3.1 template<typename T = double> std::uniform_real_distribution<T> qpp::UniformRealDistribution< T >::_d [protected]

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

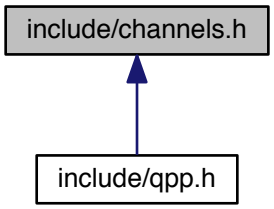
- include/classes/stat.h

Chapter 8

File Documentation

8.1 include/channels.h File Reference

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



Namespaces

- qpp

Functions

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

  Superoperator matrix representation.
- cmat qpp::choi (const std::vector< cmat > &Ks)

  Choi matrix representation.
- std::vector< cmat > qpp::choi2kraus (const cmat &A)

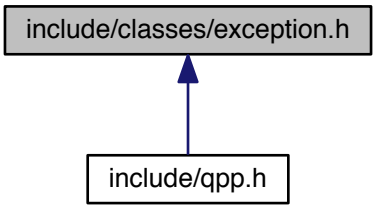
  Extracts orthogonal Kraus operators from Choi matrix.
- template<typename Derived >
  cmat qpp::channel (const Eigen::MatrixBase< Derived > &rho, const std::vector< cmat > &Ks)

  Applies the channel specified by the set of Kraus operators Ks to the density matrix rho.
- template<typename Derived >
  cmat qpp::channel (const Eigen::MatrixBase< Derived > &rho, const std::vector< cmat > &Ks, const std↩
  ::vector< std::size_t > &subsys, const std::vector< std::size_t > &dims)

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

8.2 include/classes/exception.h File Reference

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



Classes

- class qpp::Exception

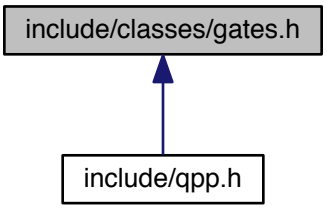
  Generates custom exceptions, used when validating function parameters.

Namespaces

- qpp

8.3 include/classes/gates.h File Reference

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



Classes

- class qpp::Gates

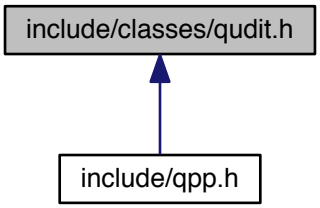
  Singleton class that implements most commonly used gates.

Namespaces

- qpp

8.4 include/classes/qudit.h File Reference

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



Classes

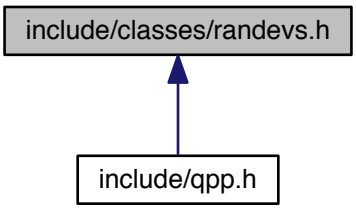
- class qpp::Qudit

Namespaces

- qpp

8.5 include/classes/randevs.h File Reference

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



Classes

- class qpp::RandomDevices

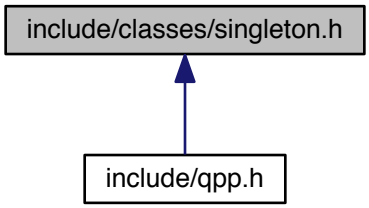
Singeleton class that manages the source of randomness in the library.

Namespaces

- qpp

8.6 include/classes/singleton.h File Reference

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



Classes

- class qpp::internal::Singleton< T >

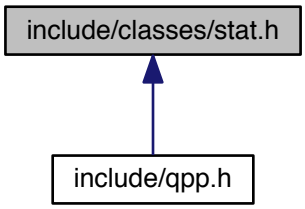
  Singleton policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern)

Namespaces

- qpp
- qpp::internal

8.7 include/classes/stat.h File Reference

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



Classes

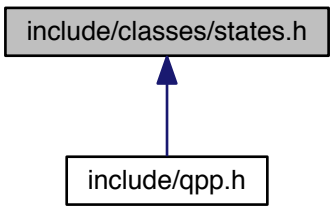
- class qpp::NormalDistribution< T >
  Template light wrapper around std::normal_distribution<>, generates normally-distributed random numbers.
- class qpp::UniformRealDistribution< T >
- class qpp::UniformIntegerDistribution< T >
- class qpp::DiscreteDistribution< T >
- class qpp::DiscreteDistributionAbsSquare< T >

Namespaces

- qpp

8.8 include/classes/states.h File Reference

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



Classes

- class qpp::States

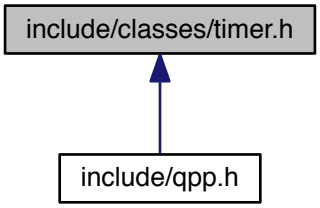
Singleton class that implements most commonly used states.

Namespaces

- qpp

8.9 include/classes/timer.h File Reference

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



Classes

- class qpp::Timer

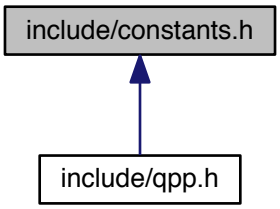
  Measures time.

Namespaces

- qpp

8.10 include/constants.h File Reference

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



Namespaces

- qpp

Functions

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

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

  User-defined literal for complex $i = \sqrt{-1}$ (real overload)
- std::complex< double > qpp::omega (std::size_t D)

  D-th root of unity.

Variables

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

  Used in qpp::disp() and qpp::displn() for setting to zero numbers that have their absolute value smaller than qpp::ct↩ ::chop.
- constexpr double qpp::eps = 1e-12

  Used to decide whether a number or expression in double precision is zero or not.
- constexpr std::size_t qpp::maxn = 64

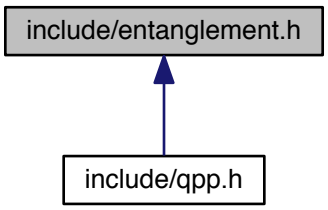
  Maximum number of qubits.
- constexpr double qpp::pi = 3.14159265358979323846264338327950288 4

  π
- constexpr double qpp::ee = 2.71828182845904523536028747135266249 7

  Base of natural logarithm, e.

8.11 include/entanglement.h File Reference

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



Namespaces

- qpp

Functions

- template<typename Derived >
  cmat qpp::schmidtcoeff (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &dims)

  Schmidt coefficients of the bi-partite pure state A.

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

  Schmidt basis on Alice's side.

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

  Schmidt basis on Bob's side.

- template<typename Derived >
  cmat qpp::schmidtprob (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &dims)

  Schmidt probabilities of the bi-partite pure state A.

- template<typename Derived >
  double qpp::entanglement (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &dims)

  Entanglement of the bi-partite pure state A.

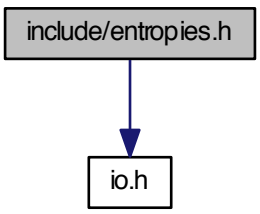
- template<typename Derived >
  double qpp::gconcurrence (const Eigen::MatrixBase< Derived > &A)

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

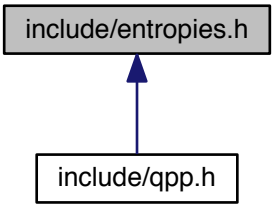
8.12 include/entropies.h File Reference

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

Include dependency graph for entropies.h:



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



Namespaces

- qpp

Functions

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

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

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

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

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

  Renyi- ∞ entropy (min entropy) of the probability distribution/density matrix A.

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

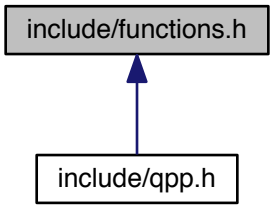
  Tsallis- α entropy of the probability distribution/density matrix A, for $\alpha \geq 0$
  .

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

  Quantum mutual information between 2 subsystems of a composite system.

8.13 include/functions.h File Reference

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



Namespaces

- qpp

Functions

- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::transpose (const Eigen::MatrixBase< Derived > &A)
  Transpose.
- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::conjugate (const Eigen::MatrixBase< Derived > &A)
  Complex conjugate.
- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::adjoint (const Eigen::MatrixBase< Derived > &A)
  Adjoint.
- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::inverse (const Eigen::MatrixBase< Derived > &A)
  Inverse.
- template<typename Derived >
  Derived::Scalar qpp::trace (const Eigen::MatrixBase< Derived > &A)
  Trace.
- template<typename Derived >
  Derived::Scalar qpp::det (const Eigen::MatrixBase< Derived > &A)
  Determinant.
- template<typename Derived >
  Derived::Scalar qpp::logdet (const Eigen::MatrixBase< Derived > &A)
  Logarithm of the determinant.
- template<typename Derived >
  Derived::Scalar qpp::sum (const Eigen::MatrixBase< Derived > &A)
  Element-wise sum.
- template<typename Derived >
  double qpp::norm (const Eigen::MatrixBase< Derived > &A)
  Trace norm.
- template<typename Derived >
  cmat qpp::evals (const Eigen::MatrixBase< Derived > &A)

Eigenvalues.

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

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

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

  Matrix square root.
- template<typename Derived >
  cmat qpp::absm (const Eigen::MatrixBase< Derived > &A)

  Matrix absolut value.
- template<typename Derived >
  cmat qpp::expm (const Eigen::MatrixBase< Derived > &A)

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

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

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

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

  Matrix power.
- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::powm (const Eigen::MatrixBase< Derived > &A, std::size_t n)

  Matrix power.
- template<typename OutputScalar , typename Derived >
  DynMat< OutputScalar > qpp::cwise (const Eigen::MatrixBase< Derived > &A, OutputScalar(*f)(const typename Derived::Scalar &))

  Functor.
- template<typename T >
  DynMat< typename T::Scalar > qpp::kron (const T &head)

  Kronecker product (variadic overload)
- template<typename T , typename... Args>
  DynMat< typename T::Scalar > qpp::kron (const T &head, const Args &...tail)

  Kronecker product (variadic overload)
- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::kron (const std::vector< Derived > &As)

  Kronecker product (std::vector overload)
- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::kron (const std::initializer_list< Derived > &As)

  Kronecker product (std::initializer_list overload)

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

  Kronecker power.
- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::reshape (const Eigen::MatrixBase< Derived > &A, std::size_t rows, std::size_t cols)

  Reshape.
- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::syspermute (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &perm, const std::vector< std::size_t > &dims)

  System permutation.
- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::ptrace1 (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &dims)

  Partial trace.
- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::ptrace2 (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &dims)

  Partial trace.
- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::ptrace (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &subsys, const std::vector< std::size_t > &dims)

  Partial trace.
- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::ptranspose (const Eigen::MatrixBase< Derived > &A, const std::vector< std::size_t > &subsys, const std::vector< std::size_t > &dims)

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

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

  Anti-commutator.
- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::prj (const Eigen::MatrixBase< Derived > &V)

  Projector.
- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::expandout (const Eigen::MatrixBase< Derived > &A, std::size_t pos, const std::vector< std::size_t > &dims)

  Expand out.
- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::grams (const std::vector< Derived > &Vs)

  Gram-Schmidt orthogonalization (std::vector overload)
- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::grams (const std::initializer_list< Derived > &Vs)

  Gram-Schmidt orthogonalization (std::initializer_list overload)
- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::grams (const Eigen::MatrixBase< Derived > &A)

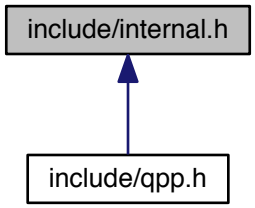
  Gram-Schmidt orthogonalization (Eigen expression (matrix) overload)
- std::vector< std::size_t > qpp::n2multiidx (std::size_t n, const std::vector< std::size_t > &dims)

  Non-negative integer index to multi-index.

- std::size_t qpp::multiidx2n (const std::vector< std::size_t > &midx, const std::vector< std::size_t > &dims)
  Multi-index to non-negative integer index.
- ket qpp::mket (const std::vector< std::size_t > &mask)
  Multi-partite qubit ket.
- ket qpp::mket (const std::vector< std::size_t > &mask, const std::vector< std::size_t > &dims)
  Multi-partite qudit ket (different dimensions overload)
- ket qpp::mket (const std::vector< std::size_t > &mask, std::size_t d)
  Multi-partite qudit ket (same dimensions overload)
- std::vector< std::size_t > qpp::invperm (const std::vector< std::size_t > &perm)
  Inverse permutation.
- std::vector< std::size_t > qpp::compperm (const std::vector< std::size_t > &perm, const std::vector< std::size_t > &sigma)
  Compose permutations.

8.14 include/internal.h File Reference

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



Namespaces

- qpp::internal
- qpp

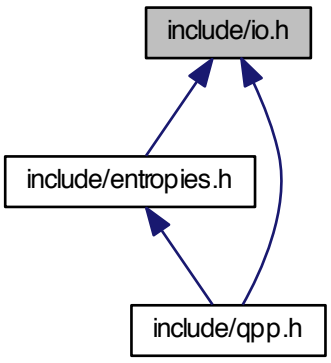
Functions

- void qpp::internal::_n2multiidx (std::size_t n, std::size_t numdims, const std::size_t *dims, std::size_t *result)
- std::size_t qpp::internal::_multiidx2n (const std::size_t *midx, std::size_t numdims, const std::size_t *dims)
- template<typename Derived >
  bool qpp::internal::_check_square_mat (const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >
  bool qpp::internal::_check_vector (const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >
  bool qpp::internal::_check_row_vector (const Eigen::MatrixBase< Derived > &A)
- template<typename Derived >
  bool qpp::internal::_check_col_vector (const Eigen::MatrixBase< Derived > &A)
- template<typename T >
  bool qpp::internal::_check_nonzero_size (const T &x)
- bool qpp::internal::_check_dims (const std::vector< std::size_t > &dims)

- template<typename Derived >
  bool qpp::internal::_check_dims_match_mat (const std::vector< std::size_t > &dims, const Eigen::Matrix↩ Base< Derived > &A)
- template<typename Derived >
  bool qpp::internal::_check_dims_match_cvect (const std::vector< std::size_t > &dims, const Eigen::Matrix↩ Base< Derived > &V)
- template<typename Derived >
  bool qpp::internal::_check_dims_match_rvect (const std::vector< std::size_t > &dims, const Eigen::Matrix↩ Base< Derived > &V)
- bool qpp::internal::_check_eq_dims (const std::vector< std::size_t > &dims, std::size_t dim)
- bool qpp::internal::_check_subsys_match_dims (const std::vector< std::size_t > &subsys, const std↩ ::vector< std::size_t > &dims)
- bool qpp::internal::_check_perm (const std::vector< std::size_t > &perm)
- template<typename Derived1 , typename Derived2 >
  DynMat< typename Derived1::Scalar > qpp::internal::_kron2 (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)
- template<typename T >
  void qpp::internal::variadic_vector_emplace (std::vector< T > &)
- template<typename T , typename First , typename... Args>
  void qpp::internal::variadic_vector_emplace (std::vector< T > &v, First &&first, Args &&...args)

8.15 include/io.h File Reference

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



Namespaces

- qpp

Functions

- template<typename T >
  void qpp::disp (const T &x, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

  Displays a standard container that supports std::begin, std::end and forward iteration. Does not add a newline.

- template<typename T >
  void qpp::displn (const T &x, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

  Displays a standard container that supports std::begin, std::end and forward iteration. Adds a newline.
- template<typename T >
  void qpp::disp (const T *x, const std::size_t n, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

  Displays a C-style array. Does not add a newline.
- template<typename T >
  void qpp::displn (const T *x, const std::size_t n, const std::string &separator, const std::string &start="[", const std::string &end="]", std::ostream &os=std::cout)

  Displays a C-style array. Adds a newline.
- template<typename Derived >
  void qpp::disp (const Eigen::MatrixBase< Derived > &A, double chop=qpp::chop, std::ostream &os=std::cout)

  Displays an Eigen expression in matrix friendly form. Does not add a new line.
- template<typename Derived >
  void qpp::displn (const Eigen::MatrixBase< Derived > &A, double chop=qpp::chop, std::ostream &os=std::cout)

  Displays an Eigen expression in matrix friendly form. Adds a newline.
- void qpp::disp (const cplx z, double chop=qpp::chop, std::ostream &os=std::cout)

  Displays a number (implicitly converted to std::complex<double>) in friendly form. Does not add a new line.
- void qpp::displn (const cplx z, double chop=qpp::chop, std::ostream &os=std::cout)

  Displays a number (implicitly converted to std::complex<double>) in friendly form. Adds a new line.
- template<typename Derived >
  void qpp::save (const Eigen::MatrixBase< Derived > &A, const std::string &fname)

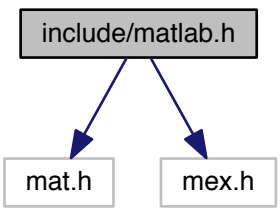
  Saves Eigen expression to a binary file (internal format) in double precission.
- template<typename Derived >
  DynMat< typename Derived::Scalar > qpp::load (const std::string &fname)

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

8.16 include/matlab.h File Reference

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

Include dependency graph for matlab.h:



Namespaces

- qpp

Functions

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

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

- template<>
  dmat qpp::loadMATLABmatrix (const std::string &mat_file, const std::string &var_name)

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

- template<>
  cmat qpp::loadMATLABmatrix (const std::string &mat_file, const std::string &var_name)

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

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

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

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

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

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

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

8.17 include/qpp.h File Reference

```
#include <algorithm>
#include <chrono>
#include <cmath>
#include <complex>
#include <cstdlib>
#include <cstring>
#include <exception>
#include <fstream>
#include <functional>
#include <iomanip>
#include <iostream>
#include <iterator>
#include <limits>
#include <numeric>
#include <ostream>
#include <random>
#include <sstream>
#include <stdexcept>
#include <string>
#include <type_traits>
#include <utility>
#include <vector>
#include <Eigen/Dense>
#include <Eigen/SVD>
#include "constants.h"
#include "types.h"
#include "classes/exception.h"
#include "classes/singleton.h"
#include "classes/states.h"
#include "classes/randevs.h"
#include "internal.h"
#include "functions.h"
#include "classes/gates.h"
#include "classes/stat.h"
#include "entropies.h"
#include "entanglement.h"
#include "channels.h"
#include "io.h"
#include "random.h"
#include "classes/qudit.h"
#include "classes/timer.h"
```

Include dependency graph for qpp.h:

Namespaces

- qpp

Variables

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

qpp::RandomDevices Singleton

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

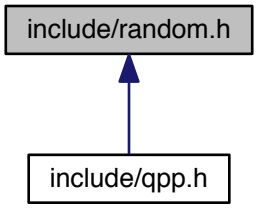
  qpp::Gates const Singleton

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

  qpp::States const Singleton

8.18 include/random.h File Reference

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



Namespaces

- qpp

Functions

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

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

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

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

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

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

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

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

- int qpp::randint (int a=std::numeric_limits< int >::min(), int b=std::numeric_limits< int >::max())

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

- template<typename Derived >
  Derived qpp::randn (std::size_t rows, std::size_t cols, double mean=0, double sigma=1)

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

- template<>
  dmat qpp::randn (std::size_t rows, std::size_t cols, double mean, double sigma)

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

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

  Generates a random complex matrix with entries (both real and imaginary) normally distributed in N(mean, sigma), specialization for complex matrices (qpp::cmat)
- double qpp::randn (double mean=0, double sigma=1)

  Generates a random real number (double) normally distributed in N(mean, sigma)
- cmat qpp::randU (std::size_t D)

  Generates a random unitary matrix.
- cmat qpp::randV (std::size_t Din, std::size_t Dout)

  Generates a random isometry matrix.
- std::vector< cmat > qpp::randkraus (std::size_t n, std::size_t D)

  Generates a set of random Kraus operators.
- cmat qpp::randH (std::size_t D)

  Generates a random Hermitian matrix.
- ket qpp::randket (std::size_t D)

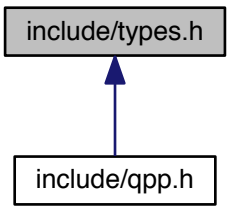
  Generates a random normalized ket (pure state vector)
- cmat qpp::randrho (std::size_t D)

  Generates a random density matrix.
- std::vector< std::size_t > qpp::randperm (std::size_t n)

  Generates a random uniformly distributed permutation.

8.19 include/types.h File Reference

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



Namespaces

- qpp

Typedefs

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

  Complex number in double precision.
- using qpp::cmat = Eigen::MatrixXcd

  Complex (double precision) dynamic Eigen matrix.
- using qpp::dmat = Eigen::MatrixXd

Real (double precision) dynamic Eigen matrix.

- using qpp::ket = Eigen::Matrix< cplx, Eigen::Dynamic, 1 >

  Complex (double precision) dynamic Eigen column matrix.

- using qpp::bra = Eigen::Matrix< cplx, 1, Eigen::Dynamic >

  Complex (double precision) dynamic Eigen row matrix.

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

  Dynamic Eigen matrix over the field specified by Scalar.

Index