CQEDSimulator

1.0.0

Generated by Doxygen 1.6.1

Sun Sep 6 20:54:26 2009

Contents

1 Class Index **1**

1.1 Class Hierarchy 1

2 Class Index **3**

2.1 Class List 3

3 File Index **5**

3.1 File List 5

4 Class Documentation **7**

4.1 Bra Class Reference 7

4.1.1 Detailed Description 8

4.1.2 Constructor & Destructor Documentation 8

4.1.2.1 Bra 8

4.1.2.2 Bra 8

4.1.2.3 Bra 8

4.1.2.4 Bra 8

4.1.3 Member Function Documentation 8

4.1.3.1 getket 8

4.1.3.2 print 9

4.1.4 Friends And Related Function Documentation 9

4.1.4.1 operator* 9

4.1.4.2 operator* 9

4.2 Ket Class Reference 10

4.2.1 Detailed Description 12

4.2.2 Constructor & Destructor Documentation 12

4.2.2.1 Ket 12

4.2.2.2 Ket 12

4.2.2.3 Ket 12

4.2.2.4 ∼Ket . . . 12
4.2.3 Member Function Documentation . . . 12
4.2.3.1 getbra . . . 13
4.2.3.2 getdim . . . 13
4.2.3.3 getnorme . . . 13
4.2.3.4 getscalarproduct . . . 13
4.2.3.5 operator() . . . 13
4.2.3.6 operator() . . . 13
4.2.3.7 operator+ . . . 13
4.2.3.8 operator+= . . . 13
4.2.3.9 operator- . . . 14
4.2.3.10 operator= . . . 14
4.2.3.11 print . . . 14
4.2.3.12 redim . . . 14
4.2.4 Friends And Related Function Documentation . . . 14
4.2.4.1 operator∗ . . . 14
4.2.4.2 operator∗ . . . 14
4.2.4.3 operator∗ . . . 15
4.2.4.4 operator∗ . . . 15
4.2.4.5 operator∗ . . . 15
4.2.4.6 operator∗ . . . 15
4.2.4.7 operator/ . . . 15
4.2.4.8 operator/ . . . 15
4.2.4.9 operator<< . . . 15
4.3 QAnnihilation Class Reference . . . 16
4.3.1 Detailed Description . . . 16
4.3.2 Constructor & Destructor Documentation . . . 16
4.3.2.1 QAnnihilation . . . 16
4.4 QCreation Class Reference . . . 17
4.4.1 Detailed Description . . . 17
4.4.2 Constructor & Destructor Documentation . . . 17
4.4.2.1 QCreation . . . 17
4.5 QDisplacement Class Reference . . . 18
4.5.1 Detailed Description . . . 18
4.5.2 Constructor & Destructor Documentation . . . 18
4.5.2.1 QDisplacement . . . 18

4.5.2.2 ~QDisplacement . . . 19
4.5.3 Member Function Documentation . . . 19
4.5.3.1 operator() . . . 19
4.6 QOperator Class Reference . . . 20
4.6.1 Detailed Description . . . 21
4.6.2 Constructor & Destructor Documentation . . . 21
4.6.2.1 QOperator . . . 21
4.6.2.2 QOperator . . . 21
4.6.2.3 QOperator . . . 21
4.6.2.4 QOperator . . . 21
4.6.2.5 QOperator . . . 21
4.6.2.6 QOperator . . . 21
4.6.3 Friends And Related Function Documentation . . . 21
4.6.3.1 operator* . . . 21
4.6.3.2 operator* . . . 21
4.7 QParity Class Reference . . . 23
4.7.1 Detailed Description . . . 23
4.7.2 Constructor & Destructor Documentation . . . 23
4.7.2.1 QParity . . . 23
4.8 SMatrix Class Reference . . . 24
4.8.1 Detailed Description . . . 27
4.8.2 Constructor & Destructor Documentation . . . 27
4.8.2.1 SMatrix . . . 27
4.8.2.2 SMatrix . . . 27
4.8.2.3 SMatrix . . . 27
4.8.2.4 SMatrix . . . 27
4.8.2.5 SMatrix . . . 28
4.8.2.6 ~SMatrix . . . 28
4.8.3 Member Function Documentation . . . 28
4.8.3.1 decompositionCholesky . . . 28
4.8.3.2 decompositionLU . . . 28
4.8.3.3 getabsmatrix . . . 29
4.8.3.4 getdet . . . 29
4.8.3.5 getdetformal . . . 29
4.8.3.6 getdim . . . 29
4.8.3.7 getexpm . . . 29

4.8.3.8 gethermitiantransposed . . . 29
4.8.3.9 getinversed . . . 30
4.8.3.10 getinversedformal . . . 30
4.8.3.11 getmax . . . 30
4.8.3.12 getnorm . . . 30
4.8.3.13 getreducedmatrix . . . 30
4.8.3.14 gettrace . . . 31
4.8.3.15 gettransposed . . . 31
4.8.3.16 operator() . . . 31
4.8.3.17 operator() . . . 31
4.8.3.18 operator∗ . . . 31
4.8.3.19 operator+ . . . 31
4.8.3.20 operator- . . . 32
4.8.3.21 operator/ . . . 32
4.8.3.22 operator= . . . 32
4.8.3.23 print . . . 32
4.8.3.24 redim . . . 32
4.8.3.25 setdiag . . . 33
4.8.3.26 setdiag . . . 33
4.8.3.27 SolveCrout . . . 33
4.8.3.28 SolveGaussSeidel . . . 33
4.8.3.29 SolveSOR . . . 34
4.8.4 Friends And Related Function Documentation . . . 34
4.8.4.1 operator∗ . . . 34
4.8.4.2 operator∗ . . . 34
4.8.4.3 operator∗ . . . 34
4.8.4.4 operator∗ . . . 34
4.8.4.5 operator∗ . . . 34
4.8.4.6 operator/ . . . 35
4.8.4.7 operator/ . . . 35
4.8.4.8 operator<< . . . 35
4.9 TensorBra Class Reference . . . 36
4.9.1 Detailed Description . . . 36
4.9.2 Constructor & Destructor Documentation . . . 36
4.9.2.1 TensorBra . . . 36
4.9.2.2 TensorBra . . . 36

4.9.2.3 ~TensorBra . . . 37
4.10 TensorKet Class Reference . . . 38
4.10.1 Detailed Description . . . 38
4.10.2 Constructor & Destructor Documentation . . . 38
4.10.2.1 TensorKet . . . 39
4.10.2.2 TensorKet . . . 39
4.10.2.3 ~TensorKet . . . 39
4.10.3 Member Function Documentation . . . 39
4.10.3.1 getHdim . . . 39
4.10.3.2 getHtot . . . 39
4.10.3.3 getKet . . . 39
4.10.3.4 getNb . . . 39
4.10.3.5 getPartialTrace . . . 39
4.11 TensorQOperator Class Reference . . . 40
4.11.1 Detailed Description . . . 40
4.11.2 Constructor & Destructor Documentation . . . 41
4.11.2.1 TensorQOperator . . . 41
4.11.2.2 TensorQOperator . . . 41
4.11.2.3 ~TensorQOperator . . . 41
4.11.3 Member Function Documentation . . . 41
4.11.3.1 getHdim . . . 41
4.11.3.2 getHtot . . . 41
4.11.3.3 getNb . . . 41
4.11.3.4 getPartialTrace . . . 41
4.11.3.5 getQOperator . . . 42

5 File Documentation **43**

5.1 Bra.cpp File Reference . . . 43
5.1.1 Detailed Description . . . 43
5.1.2 Function Documentation . . . 43
5.1.2.1 operator* . . . 43
5.1.2.2 operator* . . . 43
5.2 Bra.h File Reference . . . 44
5.2.1 Detailed Description . . . 44
5.3 CQEDSimulator.cpp File Reference . . . 45
5.3.1 Detailed Description . . . 46
5.3.2 Define Documentation . . . 46

5.3.2.1 _USE_MATH_DEFINES . . . 46
5.3.3 Function Documentation . . . 46
5.3.3.1 GetMatrixFromMatlab . . . 46
5.3.3.2 SendMatrixToMatlab . . . 46
5.3.3.3 SendVariableToMatlab . . . 46
5.3.3.4 SendVariableToMatlab . . . 47
5.3.3.5 Test2Matrix . . . 47
5.3.3.6 TestBra . . . 47
5.3.3.7 TestBraKetOperator . . . 47
5.3.3.8 TestKet . . . 47
5.3.3.9 TestMatrix . . . 47
5.3.3.10 TestMatrixFunctions . . . 47
5.3.3.11 TestQO . . . 47
5.3.3.12 WignerFunctionFock . . . 47
5.4 CQEDSimulator.h File Reference . . . 49
5.4.1 Detailed Description . . . 49
5.4.2 Define Documentation . . . 49
5.4.2.1 EXPORTED_FUNCTION . . . 50
5.4.3 Function Documentation . . . 50
5.4.3.1 Test2Matrix . . . 50
5.4.3.2 TestBra . . . 50
5.4.3.3 TestBraKetOperator . . . 50
5.4.3.4 TestKet . . . 50
5.4.3.5 TestMatrix . . . 50
5.4.3.6 TestMatrixFunctions . . . 50
5.4.3.7 TestQO . . . 50
5.4.3.8 WignerFunctionFock . . . 50
5.5 includes.h File Reference . . . 52
5.5.1 Detailed Description . . . 52
5.6 Ket.cpp File Reference . . . 53
5.6.1 Detailed Description . . . 53
5.6.2 Function Documentation . . . 53
5.6.2.1 operator* . . . 53
5.6.2.2 operator* . . . 53
5.6.2.3 operator* . . . 53
5.6.2.4 operator* . . . 53

5.6.2.5 operator* . . . 54
5.6.2.6 operator* . . . 54
5.6.2.7 operator/ . . . 54
5.6.2.8 operator/ . . . 54
5.6.2.9 operator<< . . . 54
5.7 Ket.h File Reference . . . 55
5.7.1 Detailed Description . . . 55
5.7.2 Define Documentation . . . 55
5.7.2.1 EXPORTED_FUNCTION . . . 55
5.8 QO.cpp File Reference . . . 56
5.8.1 Detailed Description . . . 56
5.8.2 Function Documentation . . . 56
5.8.2.1 basis . . . 56
5.9 QO.h File Reference . . . 57
5.9.1 Detailed Description . . . 57
5.9.2 Define Documentation . . . 57
5.9.2.1 EXPORTED_FUNCTION . . . 57
5.9.3 Function Documentation . . . 57
5.9.3.1 basis . . . 58
5.10 QOperators.cpp File Reference . . . 59
5.10.1 Detailed Description . . . 59
5.10.2 Function Documentation . . . 59
5.10.2.1 operator* . . . 59
5.10.2.2 operator* . . . 59
5.11 QOperators.h File Reference . . . 60
5.11.1 Detailed Description . . . 60
5.12 SMatrix.cpp File Reference . . . 61
5.12.1 Detailed Description . . . 61
5.12.2 Function Documentation . . . 61
5.12.2.1 operator* . . . 61
5.12.2.2 operator* . . . 61
5.12.2.3 operator* . . . 62
5.12.2.4 operator* . . . 62
5.12.2.5 operator* . . . 62
5.12.2.6 operator/ . . . 62
5.12.2.7 operator/ . . . 62

5.12.2.8 operator<< 63
5.13 SMatrix.h File Reference 64
5.13.1 Detailed Description 64
5.13.2 Define Documentation 64
5.13.2.1 EXPORTED_FUNCTION 64

Chapter 1

Class Index

1.1 Class Hierarchy

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

- Ket . . . 10
  - Bra . . . 7
    - TensorBra . . . 36
  - TensorKet . . . 38
- SMatrix . . . 24
  - QOperator . . . 20
    - QAnnihilation . . . 16
    - QCreation . . . 17
    - QDisplacement . . . 18
    - QParity . . . 23
    - TensorQOperator . . . 40

Chapter 2

Class Index

2.1 Class List

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

Bra (Bra, the bra class definition) 7
Ket (Ket, ket class definition) 10
QAnnihilation (QAnnihilation, quantum annihilation operator class definition) 16
QCreation (QCreation, quantum creation operator class definition) 17
QDisplacement (QDisplacement, quantum displacement operator class definition) 18
QOperator (QOperator, quantum operator class definition) 20
QParity (QParity, quantum parity operator class definition. exp(i∗pi∗a'∗a)) 23
SMatrix (SMatrix, square matrix class definition) 24
TensorBra (TensorBra definition class) 36
TensorKet (TensorKet definition class) 38
TensorQOperator (TensorQOperator definition class) 40

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Bra.cpp (Contains the Bra definitions) 43
Bra.h (Contains the Bra Object class) 44
CQEDSimulator.cpp (Contains all fonctions for matlab or win32 program) 45
CQEDSimulator.h (The main file for C and Matlab application) 49
includes.h (Contains all the necessary includes for all classes) 52
Ket.cpp (Contains the Ket definitions) 53
Ket.h (Contains the Ket Object class) 55
QO.cpp (Contains the Quantum Object definitions) 56
QO.h (Contains the Quantum Object class) 57
QOperators.cpp (Contains the Quantum Operators Object definitions) 59
QOperators.h (Contains the Quantum Operators class) 60
SMatrix.cpp (Contains SquareMatrix definitions) 61
SMatrix.h (Contains the SquareMatrix class) 64

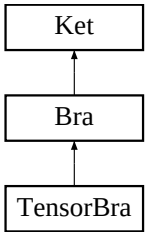
Chapter 4

Class Documentation

4.1 Bra Class Reference

Bra, the bra class definition.

`#include <Bra.h>`Inheritance diagram for Bra::



Public Member Functions

- Bra (int dim_i=1)

  Default bra constructor.

- Bra (int dim_i, complex< double > *Vtab_i)

  Creates a Bra with dimension dim_i, and data Vtab_i.

- Bra (const Ket &v)

  Constructor by copy from a Ket.

- Bra (const Bra &v)

  Constructor by copy.

- void print () const

  Print the vector in a row.

- Ket getket () const

  Send the respective Ket.

Friends

- EXPORTED_FUNCTION Bra operator∗ (complex< double > a, const Bra &v)

  Scalar complex - bra multiplication.

- EXPORTED_FUNCTION Bra operator∗ (double a, const Bra &v)

  Scalar double - bra multiplication.

4.1.1 Detailed Description

Bra, the bra class definition. This class provides all the necessaries proprieties to work with bra and operators.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Bra::Bra (int *dim_i* = 1) [inline]

Default bra constructor.

4.1.2.2 Bra::Bra (int *dim_i*, complex< double > ∗ *Vtab_i*) [inline]

Creates a Bra with dimension dim_i, and data Vtab_i.

4.1.2.3 Bra::Bra (const Ket & *v*) [inline]

Constructor by copy from a Ket.

4.1.2.4 Bra::Bra (const Bra & *v*) [inline]

Constructor by copy.

4.1.3 Member Function Documentation

4.1.3.1 Ket Bra::getket () const

Send the respective Ket.

Returns:

Returns the respective ket.

4.1.3.2 void Bra::print () const

Print the vector in a row. Output the vector as a row vector, i.e.: v.print();

Returns:

Returns the bra output.

Reimplemented from Ket.

4.1.4 Friends And Related Function Documentation

4.1.4.1 EXPORTED_FUNCTION Bra operator∗ (double *a*, const Bra & *v*) [friend]

Scalar double - bra multiplication.

Reimplemented from Ket.

4.1.4.2 EXPORTED_FUNCTION Bra operator∗ (complex< double > *a*, const Bra & *v*) [friend]

Scalar complex - bra multiplication.

Reimplemented from Ket.

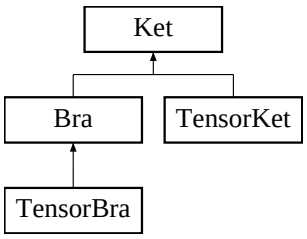
The documentation for this class was generated from the following files:

- Bra.h
- Bra.cpp

4.2 Ket Class Reference

Ket, ket class definition.

`#include <Ket.h>`Inheritance diagram for Ket::



Public Member Functions

- Ket (int dim_i=1)

  Default constructor, creates a Ket with dimension igual 1.

- Ket (int dim_i, complex< double > *Vtab_i)

  Creates a Ket with dimension dim_i, and data Vtab_i.

- Ket (const Ket &v)

  Constructor by copy.

- ~Ket ()

  Destructor.

- int getdim () const

  Send the Ket dimension.

- void redim (int ndim)

  Modify the Ket dimension.

- Bra getbra () const

  Send the respective Bra.

- complex< double > getscalarproduct (Ket right) const

  Get the scalar product.

- double getnorme () const

  Send the vector norm.

- void print () const

  Print the vector in a column.

- complex< double > & operator() (int i)

  Acces to vector elements.

- complex< double > operator() (int i) const
  Read/Write vector elements.
- const Ket & operator= (const Ket &v)
  Affectation operator.
- Ket operator+ (const Ket &v) const
  Addition operator.
- Ket operator- (const Ket &v) const
  Soustraction operator.
- void operator+= (const Ket &v)
  += *operator.*

Friends

- EXPORTED_FUNCTION Ket operator* (complex< double > a, const Ket &v)
  Scalar complex - ket multiplication.
- EXPORTED_FUNCTION Ket operator* (double a, const Ket &v)
  Scalar double - ket multiplication.
- EXPORTED_FUNCTION Ket operator* (const Ket &v, complex< double > a)
  Ket - scalar complex multiplication.
- EXPORTED_FUNCTION Ket operator* (const Ket &v, double a)
  Ket - scalar double multiplication.
- EXPORTED_FUNCTION Ket operator/ (const Ket &v, complex< double > a)
  Ket - scalar complex multiplication.
- EXPORTED_FUNCTION Ket operator/ (const Ket &v, double a)
  Ket - scalar double multiplication.
- EXPORTED_FUNCTION complex< double > operator* (const Bra &bra, const Ket &ket)
  Bra - Ket multiplication.
- EXPORTED_FUNCTION QOperator operator* (const Ket &ket, const Bra &bra)
  Ket - Bra multiplication.
- EXPORTED_FUNCTION std::ostream & operator<< (std::ostream &out, const Ket &v)
  Write to a flux, allow cout<<ket; .

4.2.1 Detailed Description

Ket, ket class definition. This class provides all the necessaries proprieties to work with kets and operators.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Ket::Ket (int *dim_i* = 1)

Default constructor, creates a Ket with dimension igual 1. To create a ket with all elements 0, you need to pass a valid dimension (dim_i > 0).

Parameters:

dim_i Ket column dimension.

4.2.2.2 Ket::Ket (int *dim_i*, complex< double > * *Vtab_i*)

Creates a Ket with dimension dim_i, and data Vtab_i. To create a ket with all elements 0, you need to pass a valid dimension and an array

Parameters:

dim_i Ket column dimension.

Vtab_i Array with data.

4.2.2.3 Ket::Ket (const Ket & *v*)

Constructor by copy. To create a ket by copy, you need to pass the ket to be copied.

Parameters:

v Ket to be copied.

4.2.2.4 Ket::~Ket ()

Destructor. Destroy all ket data.

4.2.3 Member Function Documentation

4.2.3.1 Bra Ket::getbra () const

Send the respective Bra.

Returns:

Returns the respective Bra.

4.2.3.2 int Ket::getdim () const [inline]

Send the Ket dimension.

4.2.3.3 double Ket::getnorme () const

Send the vector norm.

Returns:

Returns the ket norm as the vector norm.

4.2.3.4 complex< double > Ket::getscalarproduct (Ket *right*) const

Get the scalar product.

Returns:

Returns the scalar product between 2 kets as 2 VECTORS, it's not the tensor product.

4.2.3.5 complex<double> Ket::operator() (int *i*) const [inline]

Read/Write vector elements.

4.2.3.6 complex<double>& Ket::operator() (int *i*) [inline]

Acces to vector elements.

4.2.3.7 Ket Ket::operator+ (const Ket & *v*) const

Addition operator. Allow the use of the + operator, i.e.: a = a + b + ... + c;.

4.2.3.8 void Ket::operator+= (const Ket & *v*)

+= operator. Allow the use of += operator, i.e.: a += b;.

4.2.3.9 Ket Ket::operator- (const Ket & *v*) const

Soustraction operator. Allow the use of - operator, i.e.: a = a - b - ... - c;.

4.2.3.10 const Ket & Ket::operator= (const Ket & *v*)

Affectation operator. Allow you to use the = operator, i.e.: Ket v = b; .

4.2.3.11 void Ket::print () const

Print the vector in a column. Allow you to print the ket as a column vector.

Returns:

Returns the ket output

Reimplemented in Bra.

4.2.3.12 void Ket::redim (int *ndim*)

Modify the Ket dimension. To modify the ket dimension you pass the new dimension.

Parameters:

ndim New ket dimension.

Returns:

Returns a modified ket.

4.2.4 Friends And Related Function Documentation

4.2.4.1 EXPORTED_FUNCTION QOperator operator∗ (const Ket & *ket*, const Bra & *bra*) [friend]

Ket - Bra multiplication.

Returns:

Returns a QOperator.

4.2.4.2 EXPORTED_FUNCTION complex<double> operator∗ (const Bra & *bra*, const Ket & *ket*) [friend]

Bra - Ket multiplication.

4.2.4.3 EXPORTED_FUNCTION Ket operator∗ (const Ket & *v*, double *a*) [friend]

Ket - scalar double multiplication.

4.2.4.4 EXPORTED_FUNCTION Ket operator∗ (const Ket & *v*, complex< double > *a*) [friend]

Ket - scalar complex multiplication.

4.2.4.5 EXPORTED_FUNCTION Ket operator∗ (double *a*, const Ket & *v*) [friend]

Scalar double - ket multiplication.

Reimplemented in Bra.

4.2.4.6 EXPORTED_FUNCTION Ket operator∗ (complex< double > *a*, const Ket & *v*) [friend]

Scalar complex - ket multiplication.

Reimplemented in Bra.

4.2.4.7 EXPORTED_FUNCTION Ket operator/ (const Ket & *v*, double *a*) [friend]

Ket - scalar double multiplication.

4.2.4.8 EXPORTED_FUNCTION Ket operator/ (const Ket & *v*, complex< double > *a*) [friend]

Ket - scalar complex multiplication.

4.2.4.9 EXPORTED_FUNCTION std::ostream& operator<< (std::ostream & *out*, const Ket & *v*) [friend]

Write to a flux, allow cout<<ket; .

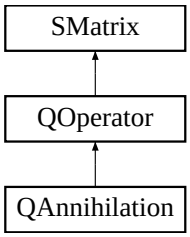
The documentation for this class was generated from the following files:

- Ket.h
- Ket.cpp

4.3 QAnnihilation Class Reference

QAnnihilation, quantum annihilation operator class definition.

`#include <QOperators.h>`Inheritance diagram for QAnnihilation::



Public Member Functions

- QAnnihilation (int dim_i=1)

4.3.1 Detailed Description

QAnnihilation, quantum annihilation operator class definition. This class provides a fast way to create the a operator.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 QAnnihilation::QAnnihilation (int *dim_i* = 1)

Allow you to create a QAnnihilation operator with dimention dim_i.

Parameters:

dim_i Hilbert space dimension.

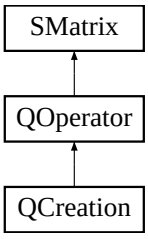
The documentation for this class was generated from the following files:

- QOperators.h
- QOperators.cpp

4.4 QCreation Class Reference

QCreation, quantum creation operator class definition.

`#include <QOperators.h>`Inheritance diagram for QCreation::



Public Member Functions

- QCreation (int dim_i=1)

  Singular constructor.

4.4.1 Detailed Description

QCreation, quantum creation operator class definition. This class provides a fast way to create the a' operator.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 QCreation::QCreation (int *dim_i* = 1)

Singular constructor. Allow you to create a QCreation operator with dimention dim_i.

Parameters:

dim_i Hilbert space dimension.

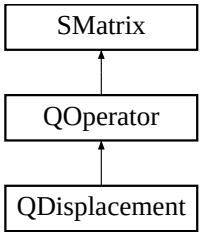
The documentation for this class was generated from the following files:

- QOperators.h
- QOperators.cpp

4.5 QDisplacement Class Reference

QDisplacement, quantum displacement operator class definition.

`#include <QOperators.h>`Inheritance diagram for QDisplacement::



Public Member Functions

- QDisplacement (int dim_i=1)

  Default Constructor.

- ~QDisplacement ()

  Destructor.

- QOperator operator() (complex< double > alpha) const

  Returns the displacement value in a alpha point: D(alpha).

4.5.1 Detailed Description

QDisplacement, quantum displacement operator class definition. This class provides a fast way to create the displacement operator.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 QDisplacement::QDisplacement (int *dim_i* = 1)

Default Constructor. Allow you to create a QDisplacement operator with dimention dim_i.

Parameters:

dim_i Hilbert space dimension.

4.5.2.2 QDisplacement::~QDisplacement ()

Destructor.

4.5.3 Member Function Documentation

4.5.3.1 QOperator QDisplacement::operator() (complex< double > *alpha*) const [inline]

Returns the displacement value in a alpha point: D(alpha).

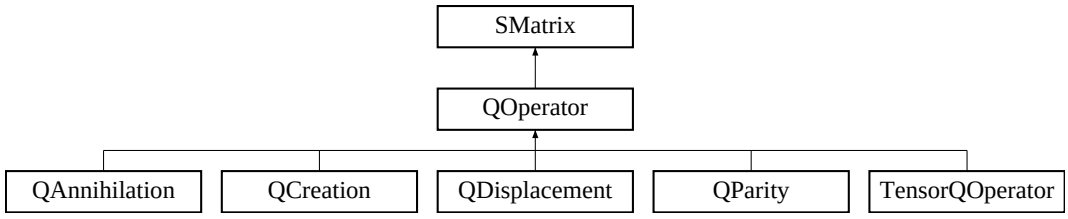
The documentation for this class was generated from the following files:

- QOperators.h
- QOperators.cpp

4.6 QOperator Class Reference

QOperator, quantum operator class definition.

`#include <QOperators.h>`Inheritance diagram for QOperator::



Public Member Functions

- QOperator (int dim_i=1)

  Default constructor, see SMatrix.

- QOperator (int dim_i, complex< double > diag)

  Diagonal constructor, see SMatrix.

- QOperator (int dim_i, const complex< double > *Mtab_i)

  Creation from array, see SMatrix.

- QOperator (int dim_i, double *real, double *imag)

  Creation from real array and imaginary array, see SMatrix.

- QOperator (const SMatrix &m)

  Constructor by copy from SMatrix.

- QOperator (const QOperator &m)

  Constructor by copy.

Friends

- EXPORTED_FUNCTION Ket operator* (const QOperator &m, const Ket &ket)

  QOperator - ket multiplication.

- EXPORTED_FUNCTION Bra operator* (const Bra &bra, const QOperator &m)

  Bra - QOperator multiplication.

4.6.1 Detailed Description

QOperator, quantum operator class definition. This class provides the basical operation between Ket - Bra - Operators.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 QOperator::QOperator (int *dim_i* = 1) [inline]

Default constructor, see SMatrix.

4.6.2.2 QOperator::QOperator (int *dim_i*, complex< double > *diag*) [inline]

Diagonal constructor, see SMatrix.

4.6.2.3 QOperator::QOperator (int *dim_i*, const complex< double > * *Mtab_i*) [inline]

Creation from array, see SMatrix.

4.6.2.4 QOperator::QOperator (int *dim_i*, double * *real*, double * *imag*) [inline]

Creation from real array and imaginary array, see SMatrix.

4.6.2.5 QOperator::QOperator (const SMatrix & *m*) [inline]

Constructor by copy from SMatrix.

4.6.2.6 QOperator::QOperator (const QOperator & *m*) [inline]

Constructor by copy.

4.6.3 Friends And Related Function Documentation

4.6.3.1 EXPORTED_FUNCTION Bra operator* (const Bra & *bra*, const QOperator & *m*) [friend]

Bra - QOperator multiplication.

4.6.3.2 EXPORTED_FUNCTION Ket operator* (const QOperator & *m*, const Ket & *ket*) [friend]

QOperator - ket multiplication.

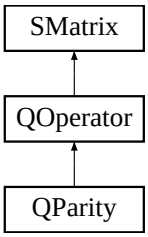
The documentation for this class was generated from the following file:

- QOperators.h

4.7 QParity Class Reference

QParity, quantum parity operator class definition. exp(i∗pi∗a'∗a).

`#include <QOperators.h>`Inheritance diagram for QParity::



Public Member Functions

- QParity (int dim_i=1)
  Default constructor.

4.7.1 Detailed Description

QParity, quantum parity operator class definition. exp(i∗pi∗a'∗a). This class provides a fast way to create the parity operator.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 QParity::QParity (int *dim_i* = 1)

Default constructor. Allow you to create a QParity operator with dimention dim_i.

Parameters:

dim_i Hilbert space dimension.

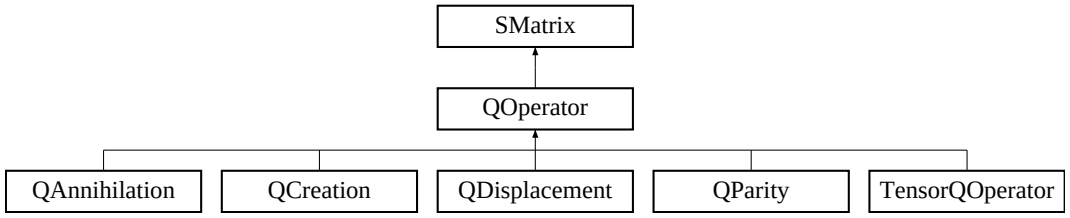
The documentation for this class was generated from the following files:

- QOperators.h
- QOperators.cpp

4.8 SMatrix Class Reference

SMatrix, square matrix class definition.

`#include <SMatrix.h>`Inheritance diagram for SMatrix::



Public Member Functions

- SMatrix (int dim_i=1)
  Default initialisation, matrix dimension is dim_i, all elements are zero.
- SMatrix (int dim_i, complex< double > diag)
  Creates a diagonal matrix with dimension dim_i.
- SMatrix (int dim_i, const complex< double > *Mtab_i)
  Creates a matrix from array, read order from left to right.
- SMatrix (int dim_i, double *real, double *imag)
  Creates a matrix from real and imaginary arrays.
- SMatrix (const SMatrix &m)
  Constructor by copy.
- ~SMatrix ()
  Destructor.
- int getdim () const
  Send matrix dimension (dim).
- void redim (int ndim)
  Modify the matrix dimension.
- void setdiag (complex< double > diag)
  Creates a diagonal matrix, which all elements = diag.
- void setdiag (complex< double > *tab_diag)
  Transform the matrix in a diagonal matrix one, which all elements = array tab_diag.
- void print (const char *option="") const
  Output matrix as C code, ONLY for matlab mexPrintf(), cout<<SMatrix instead.
- void decompositionLU (SMatrix &L, SMatrix &U)

LU decomposition.

- void decompositionCholesky (SMatrix &L, SMatrix &U)

  Cholesky decomposition.

- void SolveCrout (Ket b, Ket &x, SMatrix Lm, SMatrix Um)

  Solve linear system using Crout Method.

- void SolveSOR (Ket b, Ket &x)

  Solve linear system using SOR Method.

- void SolveGaussSeidel (Ket b, Ket &x)

  Solve linear system using Gauss Seidel Method.

- complex< double > getdetformal () const

  Calculate the determinant (formal mode).

- complex< double > getdet () const

  Calculate the determinant with LU decomposition.

- complex< double > gettrace () const

  Calculate the matrix trace.

- double getnorm () const

  Calculate the matrix norm.

- double getmax (double a[ ], int n) const

  Get the maximum value.

- SMatrix gettransposed () const

  Calculate the matrix transposed.

- SMatrix gethermitiantransposed () const

  Calculate the conjugate transposed.

- SMatrix getreducedmatrix (int row, int col) const

  Return a matrix without row i and column j.

- SMatrix getinversedformal () const

  Calculate the inversed matrix (formal mode).

- SMatrix getinversed () const

  Calculate the inversed matrix.

- SMatrix getabsmatrix () const

  Return matrix with the abs of all elements.

- SMatrix getexpm () const

  Calculate the matrix exponential using Pade algoritm (see Matlab for documentation).

- complex< double > & operator() (int i, int j)

  Acces to matrix elements.

- complex< double > operator() (int i, int j) const

  Read/Write matrix elements.

- const SMatrix & operator= (const SMatrix &m)

  Affectation operator.

- SMatrix operator+ (const SMatrix &m) const

  Addition operator.

- SMatrix operator- (const SMatrix &m) const

  Soustraction operator.

- SMatrix operator* (const SMatrix &m) const

  Multiplication operator.

- SMatrix operator/ (const SMatrix &m) const

  Division operator.

Friends

- EXPORTED_FUNCTION SMatrix operator* (complex< double > a, const SMatrix &m)

  Scalar complex - matrix multiplication.

- EXPORTED_FUNCTION SMatrix operator* (double a, const SMatrix &m)

  Scalar double - matrix multiplication.

- EXPORTED_FUNCTION SMatrix operator* (const SMatrix &m, complex< double > a)

  Matrix - scalar complex multiplication.

- EXPORTED_FUNCTION SMatrix operator* (const SMatrix &m, double a)

  Matrix - scalar double multiplication.

- EXPORTED_FUNCTION SMatrix operator* (complex< double > a, const SMatrix *m)

  Scalar complex - matrix pointer multiplication.

- EXPORTED_FUNCTION SMatrix operator/ (const SMatrix &m, double a)

  Matrix - scalar double division.

- EXPORTED_FUNCTION SMatrix operator/ (const SMatrix &m, complex< double > a)

  Matrix - scalar complex division.

- EXPORTED_FUNCTION std::ostream & operator<< (std::ostream &out, const SMatrix &m)

  Write to a flux, allow cout<<matrix; .

4.8.1 Detailed Description

SMatrix, square matrix class definition. SMatrix is the square matrix base class for linear algebra operations. This class is composed by basical matrix operations, like trace, determinant, etc... You can also find numerical methods like LU decomposition, direct and interactives methods to solve linear systems. This class provides a user friend interface for load, create and get information from matrices, this is the mother class of Quantum Operators (QOperator).

4.8.2 Constructor & Destructor Documentation

4.8.2.1 SMatrix::SMatrix (int *dim_i* = 1)

Default initialisation, matrix dimension is dim_i, all elements are zero. To create a matrix with all elements 0, you need to pass a valid dimension (dim_i $>$ 0).

Parameters:

dim_i Matrix row = column dimension.

4.8.2.2 SMatrix::SMatrix (int *dim_i*, complex< double > *diag*)

Creates a diagonal matrix with dimension dim_i. To create a diagonal matrix, you pass the matrix dimension and the non zero element that will be put in the diagonal.

Parameters:

dim_i Matrix row = column dimension.

diag Complex number with double precision that will be put in the diagonal.

4.8.2.3 SMatrix::SMatrix (int *dim_i*, const complex< double > * *Mtab_i*)

Creates a matrix from array, read order from left to right. To create a matrix from a complex<double> array that contains matrix data, you pass the matrix dimension and the data array.

Parameters:

dim_i Matrix row = column dimension.

Mtab_i Complex double precision array with all data. The data is put from left to right, in other words we fix the row and change columns.

4.8.2.4 SMatrix::SMatrix (int *dim_i*, double * *real*, double * *imag*)

Creates a matrix from real and imaginary arrays.

4.8.2.5 SMatrix::SMatrix (const SMatrix & *m*)

Constructor by copy. Allow you to do that: SMatrix A(B), where B is a SMatrix.

Parameters:

m Matrix to be copied.

4.8.2.6 SMatrix::~SMatrix ()

Destructor. Destroy all matrix array elements.

4.8.3 Member Function Documentation

4.8.3.1 void SMatrix::decompositionCholesky (SMatrix & *L*, SMatrix & *U*)

Cholesky decomposition. To elaborate a Cholesky decomposition, you need to pass the lower and upper matrices to be elaborated.

Parameters:

L Lower matrix

U Upper matrix

Returns:

Returns L and U with the correct data.

4.8.3.2 void SMatrix::decompositionLU (SMatrix & *L*, SMatrix & *U*)

LU decomposition. To decompose your matrix into a lower and upper triangular matrix product, you pass to SMatrix L (lower), SMatrix U (upper).

Parameters:

L Lower matrix to be passed.

U Upper matrix to be passed.

Returns:

Returns L and U with the correct data.

4.8.3.3 SMatrix SMatrix::getabsmatrix () const

Return matrix with the abs of all elements.

Returns:

Returns the matrix with all abs elements.

4.8.3.4 complex< double > SMatrix::getdet () const

Calculate the determinant with LU decomposition.

Returns:

Returns the matrix determinant, using LU decomposition.

4.8.3.5 complex< double > SMatrix::getdetformal () const

Calculate the determinant (formal mode).

Returns:

Returns the matrix determinant.

4.8.3.6 int SMatrix::getdim () const [inline]

Send matrix dimension (dim).

4.8.3.7 SMatrix SMatrix::getexpm () const

Calculate the matrix exponential using Pade algoritm (see Matlab for documentation).

Returns:

Returns the matrix exponential using Pade algorithm.

4.8.3.8 SMatrix SMatrix::gethermitiantransposed () const

Calculate the conjugate transposed.

Returns:

Returns the transposed conjugate matrix.

4.8.3.9 SMatrix SMatrix::getinversed () const

Calculate the inversed matrix.

Returns:

Returns the matrix inversed using LAPACK++ libraries if avaiable.

4.8.3.10 SMatrix SMatrix::getinversedformal () const

Calculate the inversed matrix (formal mode).

Returns:

Returns the matrix inversed.

4.8.3.11 double SMatrix::getmax (double *a*[ ], int *n*) const

Get the maximum value.

Returns:

Returns the maximum from an array.

Parameters:

a Your array.

n Array dimension.

4.8.3.12 double SMatrix::getnorm () const

Calculate the matrix norm.

Returns:

Returns the matrix norm = bigger row abs sum (Matlab norm('A',inf)).

4.8.3.13 SMatrix SMatrix::getreducedmatrix (int *row*, int *col*) const

Return a matrix without row i and column j.

Parameters:

row Row to be removed.

col Column to be removed.

Returns:

Returns the reduced matrix.

4.8.3.14 complex< double > SMatrix::gettrace () const

Calculate the matrix trace.

Returns:

Returns the matrix trace.

4.8.3.15 SMatrix SMatrix::gettransposed () const

Calculate the matrix transposed.

Returns:

Returns the transposed matrix.

4.8.3.16 complex<double> SMatrix::operator() (int *i*, int *j*) const [inline]

Read/Write matrix elements.

4.8.3.17 complex<double>& SMatrix::operator() (int *i*, int *j*) [inline]

Acces to matrix elements.

4.8.3.18 SMatrix SMatrix::operator∗ (const SMatrix & *m*) const

Multiplication operator. Allow you to multiply 2 or more operators, i.e: A = A∗B∗...∗C;

Returns:

Returns the matrix multiplication.

4.8.3.19 SMatrix SMatrix::operator+ (const SMatrix & *m*) const

Addition operator. Allow you to sum 2 or more operators, i.e: A = A + B + ... + C;

Returns:

Returns the matrix sum.

4.8.3.20 SMatrix SMatrix::operator- (const SMatrix & *m*) const

Soustraction operator. Allow you to subtracte 2 or more operators, i.e: A = A - B - ... - C;

Returns:

Returns the matrix subtraction.

4.8.3.21 SMatrix SMatrix::operator/ (const SMatrix & *m*) const

Division operator. Allow you to divide 2 or more operators, i.e: A = A/B/.../C;

Returns:

Returns the matrix division.

4.8.3.22 const SMatrix & SMatrix::operator= (const SMatrix & *m*)

Affectation operator. Allow you to use the = operator, i.e: SMatrix A(2); SMatrix B(2); A = B;

4.8.3.23 void SMatrix::print (const char ∗ *option* = "") const

Output matrix as C code, ONLY for matlab mexPrintf(), cout<<SMatrix instead. You can output matrix using this command i.e.: SMatrix A(...); A.print(); or A.print("expo");

Parameters:

option If you pass the option "expo" all matrix elements will be showed with the exponential notation, else data will be showed as doubles.

Returns:

Return the matrix output.

4.8.3.24 void SMatrix::redim (int *ndim*)

Modify the matrix dimension. To modify the dimension of a matrix, you pass the new dimension.

Parameters:

ndim New matrix dimension.

4.8.3.25 void SMatrix::setdiag (complex< double > * *tab_diag*)

Transform the matrix in a diagonal matrix one, which all elements = array tab_diag. To create a diagonal matrix, you need to pass a complex array that will be put in the diagonal.

Parameters:

tab_diag Complex array that contains the elements to be put in the diagonal.

Returns:

Modify you matrix data.

4.8.3.26 void SMatrix::setdiag (complex< double > *diag*)

Creates a diagonal matrix, which all elements = diag. To create a diagonal matrix, you need to pass a complex number to be put in the diagonal.

Parameters:

diag Complex number to be put in the diagonal.

Returns:

Modify your matrix data.

4.8.3.27 void SMatrix::SolveCrout (Ket *b*, Ket & *x*, SMatrix *Lm*, SMatrix *Um*)

Solve linear system using Crout Method. To solve a linear system A∗x=b, you pass b and an empty x, you also need to pass the LU matrix from A.

Parameters:

b The know vector.

x The solution vector.

Lm The lower triangular matrix.

Um The upper triangular matrix.

Returns:

Return x with the solution.

4.8.3.28 void SMatrix::SolveGaussSeidel (Ket *b*, Ket & *x*)

Solve linear system using Gauss Seidel Method. To solve a linear system A∗x=b, you pass b and an empty x.

Parameters:

b The know vector.

x The solution vector.

Returns:

Return x with the solution.

4.8.3.29 void SMatrix::SolveSOR (Ket *b*, Ket & *x*)

Solve linear system using SOR Method. To solve a linear system A∗x=b, you pass b and an empty x.

Parameters:

b The know vector.

x The solution vector.

Returns:

Return x with the solution.

4.8.4 Friends And Related Function Documentation

4.8.4.1 EXPORTED_FUNCTION SMatrix operator∗ (complex< double > *a*, const SMatrix ∗ *m*) [friend]

Scalar complex - matrix pointer multiplication.

4.8.4.2 EXPORTED_FUNCTION SMatrix operator∗ (const SMatrix & *m*, double *a*) [friend]

Matrix - scalar double multiplication.

4.8.4.3 EXPORTED_FUNCTION SMatrix operator∗ (const SMatrix & *m*, complex< double > *a*) [friend]

Matrix - scalar complex multiplication.

4.8.4.4 EXPORTED_FUNCTION SMatrix operator∗ (double *a*, const SMatrix & *m*) [friend]

Scalar double - matrix multiplication.

4.8.4.5 EXPORTED_FUNCTION SMatrix operator∗ (complex< double > *a*, const SMatrix & *m*) [friend]

Scalar complex - matrix multiplication.

4.8.4.6 EXPORTED_FUNCTION SMatrix operator/ (const SMatrix & *m*, complex< double > *a*) [friend]

Matrix - scalar complex division.

4.8.4.7 EXPORTED_FUNCTION SMatrix operator/ (const SMatrix & *m*, double *a*) [friend]

Matrix - scalar double division.

4.8.4.8 EXPORTED_FUNCTION std::ostream& operator<< (std::ostream & *out*, const SMatrix & *m*) [friend]

Write to a flux, allow cout<<matrix; .

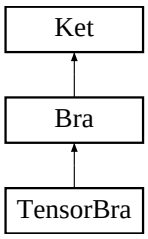
The documentation for this class was generated from the following files:

- SMatrix.h
- SMatrix.cpp

4.9 TensorBra Class Reference

TensorBra definition class.

`#include <QO.h>`Inheritance diagram for TensorBra::



Public Member Functions

- TensorBra (Bra a, Bra b)
  Constructor for 2 bra.
- TensorBra (int Nb, Bra ∗bra)
  Constructor for 3 or more bras.
- ∼TensorBra ()
  Destructor.

4.9.1 Detailed Description

TensorBra definition class. This class provides a easy way to create tensor products between 2 or more Bra.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 TensorBra::TensorBra (Bra *a*, Bra *b*)

Constructor for 2 bra.

4.9.2.2 TensorBra::TensorBra (int *Nb*, Bra ∗ *bra*)

Constructor for 3 or more bras.

4.9.2.3 TensorBra::∼TensorBra ()

Destructor.

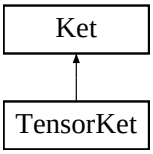
The documentation for this class was generated from the following files:

- QO.h
- QO.cpp

4.10 TensorKet Class Reference

TensorKet definition class.

`#include <QO.h>`Inheritance diagram for TensorKet::



Public Member Functions

- TensorKet (Ket a, Ket b)

  Constructor for 2 kets.

- TensorKet (int Nb, Ket ∗ket)

  Constructor for 3 or more kets.

- ∼TensorKet ()
- int getNb () const

  Send the number of kets in the tensor product.

- int getHtot () const

  Send the total dimension of Hilbert space.

- int getHdim (int object) const

  Send the singular dimension of the Hilbert space.

- Ket getKet (int i) const

  Send the singular ket operator.

- Ket getPartialTrace (int respectto)

  Send the partial trace in respect to another ket.

4.10.1 Detailed Description

TensorKet definition class. This class provides a easy way to create tensor products between 2 or more kets.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 TensorKet::TensorKet (Ket *a*, Ket *b*)

Constructor for 2 kets.

4.10.2.2 TensorKet::TensorKet (int *Nb*, Ket * *ket*)

Constructor for 3 or more kets.

4.10.2.3 TensorKet::~TensorKet ()

4.10.3 Member Function Documentation

4.10.3.1 int TensorKet::getHdim (int *object*) const [inline]

Send the singular dimension of the Hilbert space.

4.10.3.2 int TensorKet::getHtot () const [inline]

Send the total dimension of Hilbert space.

4.10.3.3 Ket TensorKet::getKet (int *i*) const [inline]

Send the singular ket operator.

4.10.3.4 int TensorKet::getNb () const [inline]

Send the number of kets in the tensor product.

4.10.3.5 Ket TensorKet::getPartialTrace (int *respectto*)

Send the partial trace in respect to another ket. Allow you to get the partial trace of a tensor product.

Parameters:

respectto Is the index of the ket to be calculated.

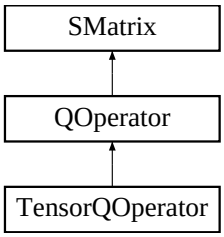
The documentation for this class was generated from the following files:

- QO.h
- QO.cpp

4.11 TensorQOperator Class Reference

TensorQOperator definition class.

`#include <QO.h>`Inheritance diagram for TensorQOperator::



Public Member Functions

- TensorQOperator (QOperator a, QOperator b)

  Constructor for 2 QOperators.

- TensorQOperator (int Nb, QOperator *op)

  Constructor for 3 or more QOperators.

- ~TensorQOperator ()

  Destructor.

- int getNb () const

  Send the number of QOperator inside the tensor product.

- int getHtot () const

  Send the total dimension of the Hilbert space.

- int getHdim (int object) const

  Send the singular dimension of Hilbert space.

- QOperator getQOperator (int i) const

  Send the singular QOperator.

- QOperator getPartialTrace (int respectto)

  Send the partial trace in respect to another QOperator.

4.11.1 Detailed Description

TensorQOperator definition class. This class provides a easy way to create tensor products between 2 or more QOperators.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 TensorQOperator::TensorQOperator (QOperator *a*, QOperator *b*)

Constructor for 2 QOperators.

4.11.2.2 TensorQOperator::TensorQOperator (int *Nb*, QOperator * *op*)

Constructor for 3 or more QOperators.

4.11.2.3 TensorQOperator::~TensorQOperator ()

Destructor.

4.11.3 Member Function Documentation

4.11.3.1 int TensorQOperator::getHdim (int *object*) const [inline]

Send the singular dimension of Hilbert space.

4.11.3.2 int TensorQOperator::getHtot () const [inline]

Send the total dimension of the Hilbert space.

4.11.3.3 int TensorQOperator::getNb () const [inline]

Send the number of QOperator inside the tensor product.

4.11.3.4 QOperator TensorQOperator::getPartialTrace (int *respectto*)

Send the partial trace in respect to another QOperator. Calculate the partial trace.

Parameters:

respectto Is the index to get the partial trace.

Returns:

Returns the the partial trace.

4.11.3.5 QOperator TensorQOperator::getQOperator (int *i*) const [inline]

Send the singular QOperator.

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

- QO.h
- QO.cpp

Chapter 5

File Documentation

5.1 Bra.cpp File Reference

Contains the Bra definitions. `#include "Bra.h"`

`#include "includes.h"`

Functions

- EXPORTED_FUNCTION Bra operator∗ (complex< double > a, const Bra &v)
- EXPORTED_FUNCTION Bra operator∗ (double a, const Bra &v)

5.1.1 Detailed Description

Contains the Bra definitions.

5.1.2 Function Documentation

5.1.2.1 EXPORTED_FUNCTION Bra operator∗ (double *a*, const Bra & *v*)

Allow the double - bra multiplication, i.e.: B = a∗B;.

5.1.2.2 EXPORTED_FUNCTION Bra operator∗ (complex< double > *a*, const Bra & *v*)

Allow the complex - bra multiplication, i.e.: B = a∗B;.

5.2 Bra.h File Reference

Contains the Bra Object class. `#include "Ket.h"`

Classes

- class Bra

  Bra, the bra class definition.

5.2.1 Detailed Description

Contains the Bra Object class.

5.3 CQEDSimulator.cpp File Reference

Contains all fonctions for matlab or win32 program. `#include "CQEDSimulator.h"`

`#include "QO.h"`

`#include "includes.h"`

Defines

- #define _USE_MATH_DEFINES

Functions

- void SendMatrixToMatlab (SMatrix A, const char *name)
  Send a matrix to Matlab workspace.
- void SendVariableToMatlab (complex< double > A, const char *name)
  Send a complex variable to Matlab workspace.
- void SendVariableToMatlab (double A, const char *name)
  Send a double to Matlab workspace.
- SMatrix GetMatrixFromMatlab (const char *variable)
  Get a matrix to Matlab workspace.
- EXPORTED_FUNCTION void WignerFunctionFock (int dim, int non0row, int non0col)
  Calculate the Wigner Function for fock states.
- EXPORTED_FUNCTION void TestMatrix (void)
  Test all matrix functions.
- EXPORTED_FUNCTION void Test2Matrix (void)
  Test matrix operations.
- EXPORTED_FUNCTION void TestMatrixFunctions (const char *variable)
  Get a matlab matrix and calculate all matrix operations.
- EXPORTED_FUNCTION void TestKet (void)
  Test Ket operations.
- EXPORTED_FUNCTION void TestBra (void)
  Test Bra operations.
- EXPORTED_FUNCTION void TestBraKetOperator (void)
  Test Bra Ket operations.
- EXPORTED_FUNCTION void TestQO (void)
  Test the Quantum Operators operations.

5.3.1 Detailed Description

Contains all fonctions for matlab or win32 program.

5.3.2 Define Documentation

5.3.2.1 #define _USE_MATH_DEFINES

5.3.3 Function Documentation

5.3.3.1 SMatrix GetMatrixFromMatlab (const char * *variable*)

Get a matrix to Matlab workspace.

Parameters:

variable Get the name of the Matlab variable.

Returns:

Returns a SMatrix with the Matlab variable.

5.3.3.2 void SendMatrixToMatlab (SMatrix *A*, const char * *name*)

Send a matrix to Matlab workspace.

Parameters:

A The CQEDSimulator matrix.

name The name of variable to be allocated in Matlab.

5.3.3.3 void SendVariableToMatlab (double *A*, const char * *name*)

Send a double to Matlab workspace.

Parameters:

A The CQEDSimulator double.

name The name of variable to be allocated in Matlab.

5.3.3.4 void SendVariableToMatlab (complex< double > *A*, const char * *name*)

Send a complex variable to Matlab workspace.

Parameters:

A The CQEDSimulator complex.

name The name of variable to be allocated in Matlab.

5.3.3.5 EXPORTED_FUNCTION void Test2Matrix (void)

Test matrix operations.

5.3.3.6 EXPORTED_FUNCTION void TestBra (void)

Test Bra operations.

5.3.3.7 EXPORTED_FUNCTION void TestBraKetOperator (void)

Test Bra Ket operations.

5.3.3.8 EXPORTED_FUNCTION void TestKet (void)

Test Ket operations.

5.3.3.9 EXPORTED_FUNCTION void TestMatrix (void)

Test all matrix functions.

5.3.3.10 EXPORTED_FUNCTION void TestMatrixFunctions (const char * *variable*)

Get a matlab matrix and calculate all matrix operations.

5.3.3.11 EXPORTED_FUNCTION void TestQO (void)

Test the Quantum Operators operations.

5.3.3.12 EXPORTED_FUNCTION void WignerFunctionFock (int *dim*, int *non0row*, int *non0col*)

Calculate the Wigner Function for fock states. Calculates the Wigner function for fock states.

Parameters:

dim The Hilbert space dimention.

non0row The non zero row for density matrix.

non0col The non zero column for density matrix.

Returns:

Creates 3 files with x,y,z points.

5.4 CQEDSimulator.h File Reference

The main file for C and Matlab application.

Defines

- #define EXPORTED_FUNCTION

Functions

- EXPORTED_FUNCTION void TestMatrix (void)

  Test all matrix functions.

- EXPORTED_FUNCTION void Test2Matrix (void)

  Test matrix operations.

- EXPORTED_FUNCTION void TestMatrixFunctions (const char *variable)

  Get a matlab matrix and calculate all matrix operations.

- EXPORTED_FUNCTION void TestKet (void)

  Test Ket operations.

- EXPORTED_FUNCTION void TestBra (void)

  Test Bra operations.

- EXPORTED_FUNCTION void TestBraKetOperator (void)

  Test Bra Ket operations.

- EXPORTED_FUNCTION void TestQO (void)

  Test the Quantum Operators operations.

- EXPORTED_FUNCTION void WignerFunctionFock (int dim, int non0row, int non0col)

  Calculates the Wigner function for fock states.

5.4.1 Detailed Description

The main file for C and Matlab application.

5.4.2 Define Documentation

5.4.2.1 #define EXPORTED_FUNCTION

CQEDSimulator.h is the main header file. To load this library type: loadlibrary('CQEDSimulator','CQEDSimulator.h') Use this header in C programs, create your own functions.

5.4.3 Function Documentation

5.4.3.1 EXPORTED_FUNCTION void Test2Matrix (void)

Test matrix operations.

5.4.3.2 EXPORTED_FUNCTION void TestBra (void)

Test Bra operations.

5.4.3.3 EXPORTED_FUNCTION void TestBraKetOperator (void)

Test Bra Ket operations.

5.4.3.4 EXPORTED_FUNCTION void TestKet (void)

Test Ket operations.

5.4.3.5 EXPORTED_FUNCTION void TestMatrix (void)

Test all matrix functions.

5.4.3.6 EXPORTED_FUNCTION void TestMatrixFunctions (const char * *variable*)

Get a matlab matrix and calculate all matrix operations.

5.4.3.7 EXPORTED_FUNCTION void TestQO (void)

Test the Quantum Operators operations.

5.4.3.8 EXPORTED_FUNCTION void WignerFunctionFock (int *dim*, int *non0row*, int *non0col*)

Calculates the Wigner function for fock states. Calculates the Wigner function for fock states.

Parameters:

dim The Hilbert space dimention.

non0row The non zero row for density matrix.

non0col The non zero column for density matrix.

Returns:

Creates 3 files with x,y,z points.

5.5 includes.h File Reference

Contains all the necessary includes for all classes. `#include <iostream>`

```
#include <stdio.h>
#include <vector>
#include <cmath>
#include <cassert>
#include <iomanip>
#include <fstream>
#include <ctime>
```

5.5.1 Detailed Description

Contains all the necessary includes for all classes.

5.6 Ket.cpp File Reference

Contains the Ket definitions. `#include "Ket.h"`

`#include "Bra.h"`

`#include "QOperators.h"`

`#include "includes.h"`

Functions

- EXPORTED_FUNCTION Ket operator∗ (complex< double > a, const Ket &v)
- EXPORTED_FUNCTION Ket operator∗ (double a, const Ket &v)
- EXPORTED_FUNCTION Ket operator∗ (const Ket &v, complex< double > a)
- EXPORTED_FUNCTION Ket operator∗ (const Ket &v, double a)
- EXPORTED_FUNCTION Ket operator/ (const Ket &v, complex< double > a)
- EXPORTED_FUNCTION Ket operator/ (const Ket &v, double a)
- EXPORTED_FUNCTION complex< double > operator∗ (const Bra &bra, const Ket &ket)
- EXPORTED_FUNCTION QOperator operator∗ (const Ket &ket, const Bra &bra)
- EXPORTED_FUNCTION std::ostream & operator<< (std::ostream &out, const Ket &v)

5.6.1 Detailed Description

Contains the Ket definitions.

5.6.2 Function Documentation

5.6.2.1 EXPORTED_FUNCTION QOperator operator∗ (const Ket & *ket*, const Bra & *bra*)

Allow the ket - bra multiplication.

5.6.2.2 EXPORTED_FUNCTION complex<double> operator∗ (const Bra & *bra*, const Ket & *ket*)

Allow the bra - ket multiplication.

5.6.2.3 EXPORTED_FUNCTION Ket operator∗ (const Ket & *v*, double *a*)

Allow the ket - double multiplication.

5.6.2.4 EXPORTED_FUNCTION Ket operator∗ (const Ket & *v*, complex< double > *a*)

Allow the ket - complex multiplication.

5.6.2.5 EXPORTED_FUNCTION Ket operator∗ (double *a*, const Ket & *v*)

Allow the double - ket multiplication.

5.6.2.6 EXPORTED_FUNCTION Ket operator∗ (complex< double > *a*, const Ket & *v*)

Allow the complex - ket multiplication.

5.6.2.7 EXPORTED_FUNCTION Ket operator/ (const Ket & *v*, double *a*)

Allow the ket - double multiplication.

5.6.2.8 EXPORTED_FUNCTION Ket operator/ (const Ket & *v*, complex< double > *a*)

Allow the ket - complex multiplication.

5.6.2.9 EXPORTED_FUNCTION std::ostream& operator<< (std::ostream & *out*, const Ket & *v*)

Allow the use of << in C++, i.e.: cout<<ket;.

5.7 Ket.h File Reference

Contains the Ket Object class. `#include <iostream>`

`#include <cassert>`

`#include <complex>`

Classes

- class Ket

  Ket, ket class definition.

Defines

- #define EXPORTED_FUNCTION

5.7.1 Detailed Description

Contains the Ket Object class.

5.7.2 Define Documentation

5.7.2.1 #define EXPORTED_FUNCTION

5.8 QO.cpp File Reference

Contains the Quantum Object definitions. `#include "QO.h"`

`#include "includes.h"`

Functions

- EXPORTED_FUNCTION Ket basis (int dim_i, int comp)

  Basis returns the ket basis element.

5.8.1 Detailed Description

Contains the Quantum Object definitions.

5.8.2 Function Documentation

5.8.2.1 EXPORTED_FUNCTION Ket basis (int *dim_i*, int *comp*)

Basis returns the ket basis element.

Parameters:

dim_i Ket dimension.

comp Ket non zero component.

Returns:

Returns a Ket with only 1 non zero element.

5.9 QO.h File Reference

Contains the Quantum Object class. `#include "QOperators.h"`

Classes

- class TensorKet

  TensorKet definition class.

- class TensorBra

  TensorBra definition class.

- class TensorQOperator

  TensorQOperator definition class.

Defines

- #define EXPORTED_FUNCTION

Functions

- EXPORTED_FUNCTION Ket basis (int dim_i, int comp)

  Basis returns the ket basis element.

5.9.1 Detailed Description

Contains the Quantum Object class.

5.9.2 Define Documentation

5.9.2.1 #define EXPORTED_FUNCTION

5.9.3 Function Documentation

5.9.3.1 EXPORTED_FUNCTION Ket basis (int *dim_i*, int *comp*)

Basis returns the ket basis element.

Parameters:

dim_i Ket dimension.

comp Ket non zero component.

Returns:

Returns a Ket with only 1 non zero element.

5.10 QOperators.cpp File Reference

Contains the Quantum Operators Object definitions. `#include "QOperators.h"`

`#include "includes.h"`

Functions

- EXPORTED_FUNCTION Ket operator∗ (const QOperator &m, const Ket &ket)
- EXPORTED_FUNCTION Bra operator∗ (const Bra &bra, const QOperator &m)

5.10.1 Detailed Description

Contains the Quantum Operators Object definitions.

5.10.2 Function Documentation

5.10.2.1 EXPORTED_FUNCTION Bra operator∗ (const Bra & *bra*, const QOperator & *m*)

Allows the bra - Operator multiplication, i.e.: m = bra∗m;.

5.10.2.2 EXPORTED_FUNCTION Ket operator∗ (const QOperator & *m*, const Ket & *ket*)

Allows the QOperator - ket multiplication, i.e.: m = m∗ket;.

5.11 QOperators.h File Reference

Contains the Quantum Operators class. `#include "SMatrix.h"`

`#include "Bra.h"`

Classes

- class QOperator

  QOperator, quantum operator class definition.

- class QCreation

  QCreation, quantum creation operator class definition.

- class QAnnihilation

  QAnnihilation, quantum annihilation operator class definition.

- class QDisplacement

  QDisplacement, quantum displacement operator class definition.

- class QParity

  QParity, quantum parity operator class definition. exp(i∗pi∗a'∗a).

5.11.1 Detailed Description

Contains the Quantum Operators class.

5.12 SMatrix.cpp File Reference

Contains SquareMatrix definitions. `#include "SMatrix.h"`

`#include "Ket.h"`

`#include "includes.h"`

Functions

- EXPORTED_FUNCTION SMatrix operator∗ (complex< double > a, const SMatrix &m)
- EXPORTED_FUNCTION SMatrix operator∗ (double a, const SMatrix &m)
- EXPORTED_FUNCTION SMatrix operator∗ (const SMatrix &m, complex< double > a)
- EXPORTED_FUNCTION SMatrix operator∗ (const SMatrix &m, double a)
- EXPORTED_FUNCTION SMatrix operator∗ (complex< double > a, const SMatrix ∗m)
- EXPORTED_FUNCTION SMatrix operator/ (const SMatrix &m, double a)
- EXPORTED_FUNCTION SMatrix operator/ (const SMatrix &m, complex< double > a)
- EXPORTED_FUNCTION std::ostream & operator<< (std::ostream &out, const SMatrix &m)

5.12.1 Detailed Description

Contains SquareMatrix definitions.

5.12.2 Function Documentation

5.12.2.1 EXPORTED_FUNCTION SMatrix operator∗ (complex< double > *a*, const SMatrix ∗ *m*)

Allow you to multiply a complex with a matrix pointer.

Returns:

Returns the matrix multiplication.

5.12.2.2 EXPORTED_FUNCTION SMatrix operator∗ (const SMatrix & *m*, double *a*)

Allow you to multiply a matrix with a double, i.e: A = B∗a;

Returns:

Returns the matrix multiplication.

5.12.2.3 EXPORTED_FUNCTION SMatrix operator∗ (const SMatrix & *m*, complex< double > *a*)

Allow you to multiply a matrix with a complex, i.e: A = B∗a;

Returns:

Returns the matrix multiplication.

5.12.2.4 EXPORTED_FUNCTION SMatrix operator∗ (double *a*, const SMatrix & *m*)

Allow you to multiply a double with a matrix, i.e: A = a∗B;

Returns:

Returns the matrix multiplication.

5.12.2.5 EXPORTED_FUNCTION SMatrix operator∗ (complex< double > *a*, const SMatrix & *m*)

Allow you to multiply a complex with a matrix, i.e: A = a∗B;

Returns:

Returns the matrix multiplication.

5.12.2.6 EXPORTED_FUNCTION SMatrix operator/ (const SMatrix & *m*, complex< double > *a*)

Allow you to divide a matrix with a complex, i.e: A = B/a;

Returns:

Returns the matrix division.

5.12.2.7 EXPORTED_FUNCTION SMatrix operator/ (const SMatrix & *m*, double *a*)

Allow you to divide a matrix with a double, i.e: A = B/a;

Returns:

Returns the matrix division.

5.12.2.8 EXPORTED_FUNCTION std::ostream& operator<< (std::ostream & *out*, const SMatrix & *m*)

Allow you to use the std::ostream operator << in C++, i.e.: cout<<A;

Returns:

Returns the matrix output.

5.13 SMatrix.h File Reference

Contains the SquareMatrix class. `#include <iostream>`

`#include <cassert>`

`#include <complex>`

Classes

- class SMatrix

  SMatrix, square matrix class definition.

Defines

- #define EXPORTED_FUNCTION

5.13.1 Detailed Description

Contains the SquareMatrix class.

5.13.2 Define Documentation

5.13.2.1 #define EXPORTED_FUNCTION

Index

~Ket
  Ket, 12
~QDisplacement
  QDisplacement, 18
~SMatrix
  SMatrix, 28
~TensorBra
  TensorBra, 36
~TensorKet
  TensorKet, 39
~TensorQOperator
  TensorQOperator, 41
_USE_MATH_DEFINES
  CQEDSimulator.cpp, 46

basis
  QO.cpp, 56
  QO.h, 57
Bra, 7
  Bra, 8
  getket, 8
  operator∗, 9
  print, 9
Bra.cpp, 43
  operator∗, 43
Bra.h, 44

CQEDSimulator.cpp, 45
  _USE_MATH_DEFINES, 46
  GetMatrixFromMatlab, 46
  SendMatrixToMatlab, 46
  SendVariableToMatlab, 46
  Test2Matrix, 47
  TestBra, 47
  TestBraKetOperator, 47
  TestKet, 47
  TestMatrix, 47
  TestMatrixFunctions, 47
  TestQO, 47
  WignerFunctionFock, 47
CQEDSimulator.h, 49
  EXPORTED_FUNCTION, 49
  Test2Matrix, 50
  TestBra, 50
  TestBraKetOperator, 50
  TestKet, 50
  TestMatrix, 50
  TestMatrixFunctions, 50
  TestQO, 50
  WignerFunctionFock, 50

decompositionCholesky
  SMatrix, 28
decompositionLU
  SMatrix, 28

EXPORTED_FUNCTION
  CQEDSimulator.h, 49
  Ket.h, 55
  QO.h, 57
  SMatrix.h, 64

getabsmatrix
  SMatrix, 28
getbra
  Ket, 12
getdet
  SMatrix, 29
getdetformal
  SMatrix, 29
getdim
  Ket, 13
  SMatrix, 29
getexpm
  SMatrix, 29
getHdim
  TensorKet, 39
  TensorQOperator, 41
gethermitiantransposed
  SMatrix, 29
getHtot
  TensorKet, 39
  TensorQOperator, 41
getinversed
  SMatrix, 29
getinversedformal
  SMatrix, 30
getKet
  TensorKet, 39
getket

Bra, 8
GetMatrixFromMatlab
CQEDSimulator.cpp, 46
getmax
SMatrix, 30
getNb
TensorKet, 39
TensorQOperator, 41
getnorm
SMatrix, 30
getnorme
Ket, 13
getPartialTrace
TensorKet, 39
TensorQOperator, 41
getQOperator
TensorQOperator, 41
getreducedmatrix
SMatrix, 30
getscalarproduct
Ket, 13
gettrace
SMatrix, 31
gettransposed
SMatrix, 31

includes.h, 52

Ket, 10
∼Ket, 12
getbra, 12
getdim, 13
getnorme, 13
getscalarproduct, 13
Ket, 12
operator<<, 15
operator∗, 14, 15
operator(), 13
operator+, 13
operator+=, 13
operator-, 13
operator/, 15
operator=, 14
print, 14
redim, 14
Ket.cpp, 53
operator<<, 54
operator∗, 53, 54
operator/, 54
Ket.h, 55
EXPORTED_FUNCTION, 55

operator<<
Ket, 15
Ket.cpp, 54
SMatrix, 35
SMatrix.cpp, 62
operator∗
Bra, 9
Bra.cpp, 43
Ket, 14, 15
Ket.cpp, 53, 54
QOperator, 21
QOperators.cpp, 59
SMatrix, 31, 34
SMatrix.cpp, 61, 62
operator()
Ket, 13
QDisplacement, 19
SMatrix, 31
operator+
Ket, 13
SMatrix, 31
operator+=
Ket, 13
operator-
Ket, 13
SMatrix, 31
operator/
Ket, 15
Ket.cpp, 54
SMatrix, 32, 34, 35
SMatrix.cpp, 62
operator=
Ket, 14
SMatrix, 32

print
Bra, 9
Ket, 14
SMatrix, 32

QAnnihilation, 16
QAnnihilation, 16
QCreation, 17
QCreation, 17
QDisplacement, 18
∼QDisplacement, 18
operator(), 19
QDisplacement, 18
QO.cpp, 56
basis, 56
QO.h, 57
basis, 57
EXPORTED_FUNCTION, 57
QOperator, 20
operator∗, 21
QOperator, 21

QOperators.cpp, 59
operator∗, 59
QOperators.h, 60
QParity, 23
QParity, 23

redim
Ket, 14
SMatrix, 32

SendMatrixToMatlab
CQEDSimulator.cpp, 46
SendVariableToMatlab
CQEDSimulator.cpp, 46
setdiag
SMatrix, 32, 33
SMatrix, 24
∼SMatrix, 28
decompositionCholesky, 28
decompositionLU, 28
getabsmatrix, 28
getdet, 29
getdetformal, 29
getdim, 29
getexpm, 29
gethermitiantransposed, 29
getinversed, 29
getinversedformal, 30
getmax, 30
getnorm, 30
getreducedmatrix, 30
gettrace, 31
gettransposed, 31
operator<<, 35
operator∗, 31, 34
operator(), 31
operator+, 31
operator-, 31
operator/, 32, 34, 35
operator=, 32
print, 32
redim, 32
setdiag, 32, 33
SMatrix, 27
SolveCrout, 33
SolveGaussSeidel, 33
SolveSOR, 34
SMatrix.cpp, 61
operator<<, 62
operator∗, 61, 62
operator/, 62
SMatrix.h, 64
EXPORTED_FUNCTION, 64
SolveCrout
SMatrix, 33
SolveGaussSeidel
SMatrix, 33
SolveSOR
SMatrix, 34

TensorBra, 36
∼TensorBra, 36
TensorBra, 36
TensorKet, 38
∼TensorKet, 39
getHdim, 39
getHtot, 39
getKet, 39
getNb, 39
getPartialTrace, 39
TensorKet, 38, 39
TensorQOperator, 40
∼TensorQOperator, 41
getHdim, 41
getHtot, 41
getNb, 41
getPartialTrace, 41
getQOperator, 41
TensorQOperator, 41
Test2Matrix
CQEDSimulator.cpp, 47
CQEDSimulator.h, 50
TestBra
CQEDSimulator.cpp, 47
CQEDSimulator.h, 50
TestBraKetOperator
CQEDSimulator.cpp, 47
CQEDSimulator.h, 50
TestKet
CQEDSimulator.cpp, 47
CQEDSimulator.h, 50
TestMatrix
CQEDSimulator.cpp, 47
CQEDSimulator.h, 50
TestMatrixFunctions
CQEDSimulator.cpp, 47
CQEDSimulator.h, 50
TestQO
CQEDSimulator.cpp, 47
CQEDSimulator.h, 50

WignerFunctionFock
CQEDSimulator.cpp, 47
CQEDSimulator.h, 50