Package ‘qsimulatR’

September 9, 2022

Version 1.1

Date 2022-09-09

Title A Quantum Computer Simulator

Description A quantum computer simulator framework with up to 24 qubits. It allows to define general single qubit gates and general controlled single qubit gates. For convenience, it currently provides the most common gates (X, Y, Z, H, Z, S, T, Rx, Ry, Rz, CNOT, SWAP, Toffoli or CCNOT, Fredkin or CSWAP). 'qsimulatR' also implements noise models. 'qsimulatR' supports plotting of circuits and is able to export circuits to 'Qiskit' <https://qiskit.org/>, a python package which can be used to run on IBM's hardware <https://quantum-computing.ibm.com/>.

Imports methods, stats

Suggests knitr, markdown, rmarkdown

License GPL-3

RoxygenNote 7.2.1

Encoding UTF-8

VignetteBuilder knitr

URL https://github.com/HISKP-LQCD/qsimulatR

BugReports https://github.com/HISKP-LQCD/qsimulatR/issues

Collate 'state.R' 'sqgate.R' 'ccqgate.R' 'cnotgate.R' 'cnqgate.R' 'cqgate.R' 'export2qiskit.R' 'measure.R' 'phase_estimation.R' 'plot-qstate.R' 'qft.R' 'qsimulatR-package.R' 'swapgate.R'

NeedsCompilation no

Author Johann Ostmeyer [aut], Carsten Urbach [aut, cre]

Maintainer Carsten Urbach <urbach@hiskp.uni-bonn.de>

Repository CRAN

Date/Publication 2022-09-09 16:10:02 UTC

R topics documented:

*,ccnotgate,qstate-method . . . 3
*,ccqgate,qstate-method . . . 3
*,cnotgate,qstate-method . . . 4
*,cnqgate,qstate-method . . . 4
*,complex,qstate-method . . . 5
*,cqgate,qstate-method . . . 5
*,cswapgate,qstate-method . . . 6
*,matrix,qstate-method . . . 6
*,sqgate,qstate-method . . . 7
*,swapgate,qstate-method . . . 7
CCNOT . . . 8
ccnotgate . . . 8
ccqgate . . . 9
CNOT . . . 9
cnotgate . . . 10
cnqgate . . . 10
cqft . . . 11
cqgate . . . 12
CSWAP . . . 12
cswapgate . . . 13
export2qiskit . . . 13
genComputationalBasis . . . 14
genNoise . . . 15
genStateNumber . . . 15
genStateString . . . 16
H . . . 17
hist.measurement . . . 17
Id . . . 18
is.bitset . . . 18
measure . . . 19
noise . . . 20
normalise . . . 21
phase_estimation . . . 21
plot,qstate,missing-method . . . 22
qft . . . 23
qsimulatR . . . 24
qstate . . . 24
Ri . . . 25
Rx . . . 26
Ry . . . 26
Rz . . . 27
S . . . 28
sqgate . . . 28
summary.measurement . . . 29
SWAP . . . 29
swapgate . . . 30

Tgate . . . 30
truth.table . . . 31
X . . . 32
Y . . . 32
Z . . . 33

Index **34**

`*,ccnotgate,qstate-method`
times-ccnotgate-qstate

Description

Applies a CCNOT (or toffoli) gate to a quantum state.

Usage

```
## S4 method for signature 'ccnotgate,qstate'
e1 * e2
```

Arguments

`e1`	object of S4 class 'ccnotgate'
`e2`	object of S4 class 'qstate'

Value

An object of S4 class 'qstate'

`*,ccqgate,qstate-method`
times-ccqgate-qstate

Description

Applies a twice controlled single qubit gate to a quantum state.

Usage

```
## S4 method for signature 'ccqgate,qstate'
e1 * e2
```

Arguments

`e1`	object of S4 class 'ccqgate'
`e2`	object of S4 class 'qstate'

Value

An object of S4 class 'qstate'

`*,cnotgate,qstate-method`
times-cnotgate-qstate

Description

Applies a CNOT gate to a quantum state.

Usage

```
## S4 method for signature 'cnotgate,qstate'
e1 * e2
```

Arguments

e1	object of S4 class 'cnotgate'
e2	object of S4 class 'qstate'

Value

An object of S4 class 'qstate'

`*,cnqgate,qstate-method`
times-cnqgate-qstate

Description

Applies n-fold controlled single qubit gate to a quantum state.

Usage

```
## S4 method for signature 'cnqgate,qstate'
e1 * e2
```

Arguments

e1	object of S4 class 'cnqgate'
e2	object of S4 class 'qstate'

Value

An object of S4 class 'qstate'

*,complex,qstate-method
times-number-qstate

Description

Multiplies a quantum gate by a global (phase) factor.

Usage

```
## S4 method for signature 'complex,qstate'
e1 * e2
```

Arguments

e1	object of S4 class 'complex'
e2	object of S4 class 'qstate'

Value

An object of S4 class 'qstate'

*,cqgate,qstate-method
times-cqgate-qstate

Description

Applies a controlled single qubit gate to a quantum state.

Usage

```
## S4 method for signature 'cqgate,qstate'
e1 * e2
```

Arguments

e1	object of S4 class 'cqgate'
e2	object of S4 class 'qstate'

Value

An object of S4 class 'qstate'

*,cswapgate,qstate-method
times-cswapgate-qstate

Description

Applies a CSWAP gate to a quantum state.

Usage

```
## S4 method for signature 'cswapgate,qstate'
e1 * e2
```

Arguments

e1	object of S4 class 'cswapgate'
e2	object of S4 class 'qstate'

Value

An object of S4 class 'qstate'

*,matrix,qstate-method
times-matrix-qstate

Description

Applies a single qubit gate to a quantum state.

Usage

```
## S4 method for signature 'matrix,qstate'
e1 * e2
```

Arguments

e1	object of S4 class 'matrix'
e2	object of S4 class 'qstate'

Value

An object of S4 class 'qstate'

*,sqgate,qstate-method
times-sqgate-qstate

Description

Applies a single qubit gate to a quantum state.

Usage

```
## S4 method for signature 'sqgate,qstate'
e1 * e2
```

Arguments

e1	object of S4 class 'sqgate'
e2	object of S4 class 'qstate'

Value

An object of S4 class 'qstate'

*,swapgate,qstate-method
times-swapgate-qstate

Description

Applies a SWAP gate to a quantum state.

Usage

```
## S4 method for signature 'swapgate,qstate'
e1 * e2
```

Arguments

e1	object of S4 class 'swapgate'
e2	object of S4 class 'qstate'

Value

An object of S4 class 'qstate'

CCNOT *The CCNOT or toffoli gate*

Description

The CCNOT or toffoli gate

Usage

```
CCNOT(bits = c(1, 2, 3))

toffoli(bits = c(1, 2, 3))
```

Arguments

bits	integer vector of length two, the first bit being the control and the second the target bit.

Value

An S4 class 'ccnotgate' object is returned

ccnotgate *The CCNOT gate*

Description

This class represents a generic CNOT gate

Slots

bits Integer vector of length 2. First two bits are the control bits, third the target bit.

Examples

```
x <- qstate(nbits=3)
z <- CCNOT(c(1,2,3)) * (H(1) * x)
```

ccqgate *A twice controlled single qubit gate*

Description

This class represents a generic controlled gate

Details

The qubits are counted from 1 to `nbits` starting with the least significant bit.

Slots

`bits` Integer. Integer vector of bits. The first two are the control bits, the third the target bit.

`gate` sqgate. The single qubit gate.

Examples

```
x <- H(1) * qstate(nbits=3)
## application of the CCX (CCNOT) gate to bit 1,2,3
z <- ccqgate(bits=c(1L, 2L, 3L), gate=X(3L)) * x
z
## the same, but differently implemented
z <- CCNOT(c(1,2,3)) * x
z
```

CNOT *The CNOT gate*

Description

The CNOT gate

Usage

```
CNOT(bits = c(1, 2))
```

Arguments

`bits`	integer vector of length two, the first bit being the control and the second the target bit.

Value

An S4 class 'cnotgate' object is returned

cnotgate *The CNOT gate*

Description

This class represents a generic CNOT gate

Slots

`bits` Integer vector of length 2. First bit is the control bit, second the target bit.

Examples

```
x <- qstate(nbits=2)
## A Bell state
z <- CNOT(c(1,2)) * (H(1) * x)
```

cnqgate *n-fold controlled single qubit gate*

Description

This class represents a generic n-fold controlled gate

Details

The qubits are counted from 1 to `nbits` starting with the least significant bit.

Slots

`cbits` Integer. Integer vector of control bits.

`tbit` Integer. Target bit.

`gate` sqgate. The single qubit gate.

`inverse` Logical. Boolean vector of same length as `cbits`. If `TRUE`, the corresponding control bit is negated.

Examples

```
x <- H(1) * qstate(nbits=3)
## application of the CCX (CCNOT) gate to bits 1,2 and 3
z <- cnqgate(cbits=c(1L, 2L), tbit=3L, gate=X(3L)) * x
z
## the same, but differently implemented
z <- CCNOT(c(1,2,3)) * x
z
```

cqft *cqft*

Description

performs the controlled quantum Fourier Trafo on the qstate x and the specified list of qubits.

Usage

```
cqft(c, x, inverse = FALSE, bits)
```

Arguments

c	integer. a single control qubit.
x	qstate. state the qft will applied to
inverse	boolean. If 'TRUE', perform inverse transform
bits	integer. list of qubits to include in the trafo. if missing, `bits=c(1:n)[-c]` is assumed, with `n` the number of qubits in `x`.

Details

Controlled Quantum Fourier Trafo

The Fourier Trafo is defined as

$$|j>->1/sqrt(N)sum_k=0_1^Nexp(2piijk/N)|k>$$

the inverse with the oposite sign in the exponential.

Value

a `qstate` object with the quantum Fourier trafo of input `x`.

Examples

```
x <- qstate(3)
y <- cqft(1, x)
z <- cqft(1, y, inverse=TRUE)
```

cqgate — *A controlled single qubit gate*

Description

This class represents a generic controlled gate

Details

The qubits are counted from 1 to `nbits` starting with the least significant bit.

Slots

`bits` Integer. Integer vector of bits. The first is the control bit, the second the target bit.

`gate` sqgate. The single qubit gate.

Examples

```
x <- H(1) * qstate(nbits=2)
## application of the CX (CNOT) gate to bit 1,2
z <- cqgate(bits=c(1L, 2L), gate=X(2L)) * x
z
## the same as, but differently implemented
z <- CNOT(c(1,2)) * x
z
```

CSWAP — *The CSWAP or Fredkin gate*

Description

The CSWAP or Fredkin gate

Usage

```
CSWAP(bits = c(1, 2, 3))

fredkin(bits = c(1, 2, 3))
```

Arguments

`bits`	integer vector of length two, the first bit being the control and the second the target bit.

Value

An S4 class 'cswapgate' object is returned

cswapgate — *The CSWAP gate*

Description

This class represents a generic SWAP gate, also called Fredkin gate

Slots

`bits` Integer vector of length 2. First two bits are the control bits, third the target bit.

Examples

```
x <- qstate(nbits=3)
z <- CSWAP(c(1,2,3)) * (H(1) * x)
```

export2qiskit — *export2qiskit*

Description

export a circuit to IBM's qiskit python format. Note that only gates can be exported where the correspondence in qiskit is known and well defined. Qiskit can then be used for IBM's QASM to run on real hardware.

Usage

```
export2qiskit(object, varname = "qc", filename = "circuit.py",
  append = FALSE, import = FALSE)
```

Arguments

`object`	a qstate object
`varname`	character. The name of the circuit variable
`filename`	character. The filename of the textfile where to store the circuit
`append`	boolean. Whether or not to append to the file. For this the file has to exist.
`import`	boolean. Shall numpy and qiskit be loaded explicitly?

Details

Export to IBM's Qiskit

Currently the following gates can be exported: H, X, Y, Z, S, Tgate, Rz, Rx, Ry, CNOT, SWAP, CCNOT, CSWAP, measure.

note that only standard gates can be exported, not self defined ones. The function will draw a warning in case a gate cannot be exported and indicate it in the output file.

Value

nothing is returned, but a file is created.

References

https://qiskit.org/documentation/

Examples

```
x <- qstate(2)
x <- H(1) * x
x <- X(2) * x
x <- CNOT(c(1,2)) * x
export2qiskit(measure(x,1)$psi)
cat(readLines("circuit.py"), sep = '\n')
file.remove("circuit.py")
```

---

genComputationalBasis *genComputationalBasis*

Description

function to generate the basis strings for given number of bits

Usage

```
genComputationalBasis(nbits, collapse = "")
```

Arguments

nbits	integer. The number of qubits
collapse	character. String to fill in between separate bits

Value

a character vector of length 2^nbits

Examples

```
genComputationalBasis(4)
genComputationalBasis(2, collapse=">|")
```

genNoise *genNoise*

Description

function to generate the noise list

Usage

```
genNoise(nbits, p = 0, bits = 1:nbits, error = "any", ...)
```

Arguments

nbits	integer. The number of qubits
p	probability with which noise is applied after every gate
bits	integer or integer array. The bit to which to apply the gate.
error	String containing the error model.
...	Additional arguments to be stored in args.

Details

See function noise for details.

Value

a list containing p, bits, error and args

Examples

```
genNoise(4)
genNoise(2, p=1, error="small", sigma=0.1)
```

genStateNumber *genStateNumber*

Description

function to generate the bit representation for a specific basis state

Usage

```
genStateNumber(int, nbits)
```

Arguments

`int`	integer number representing the basis state
`nbits`	integer. The number of qubits

Value

a integer vector of length `nbits`

Examples

```
genStateNumber(5, 4)
genStateNumber(2, 2)
```

---

genStateString *genStateString*

Description

function to generate the string for a specific basis state

Usage

```
genStateString(int, nbits, collapse = "")
```

Arguments

`int`	integer number representing the basis state
`nbits`	integer. The number of qubits
`collapse`	character. String to fill in between separate bits

Value

a character

Examples

```
genStateString(5, 4)
genStateString(2, 2, collapse=">|")
```

H — *The Hadarmard gate*

Description

The Hadarmard gate

Usage

```
H(bit)
```

Arguments

`bit`	integer. The bit to which to apply the gate

Value

An S4 class 'sqgate' object is returned

Examples

```
x <- qstate(nbits=2)
z <- H(1) * x
z
```

hist.measurement — *Plot the histogram of a quantum measurement*

Description

Plot the histogram of a quantum measurement

Usage

```
## S3 method for class 'measurement'
hist(x, only.nonzero = TRUE, by.name = only.nonzero,
  freq = TRUE, ...)
```

Arguments

`x`	object as returned by `measure`
`only.nonzero`	are the states with zero measurements to be plotted?
`by.name`	shall the xlabel contain the basis names? If `FALSE`, the index number is used.
`freq`	shall the total counts be plotted? If not, the values are normalised to 1.
`...`	Generic parameters to pass on to `barplot()`

Value

No return value.

`Id` *The identity gate*

Description

The identity gate

Usage

```
Id(bit)
```

Arguments

`bit`	integer. The bit to which to apply the gate

Value

An S4 class 'sqgate' object is returned

Examples

```
x <- qstate(nbits=2)
z <- Id(1) * x
z
```

`is.bitset` *is.bitset*

Description

checks whether or not a bit is set in target

Usage

```
is.bitset(x, bit)
```

Arguments

`x`	target vector
`bit`	integer. The bit to check

Value

a boolean vector

`measure` *Method measure*

Description

performs a masurement on a `qstate` object.

Usage

```
measure(e1, bit = NA, repetitions = NA)

## S4 method for signature 'qstate'
measure(e1, bit = NA, repetitions = 1)
```

Arguments

`e1`	object to measure
`bit`	bit to project on
`repetitions`	number of measurements

Details

`measure(e1, bit, repetitions)` performs `repetitions` many projections/measurements of the qubit `bit`. If `bit` is not given explicitly, all qubits are projected.

Value

`measure(e1, bit, repetitions)` returns a list with the measured `bit`, the number of `repetitions`, the probability distribution of all states `prob` and the results vector `value`. If all bits are measured, the basis is added to the list as `basis`. The collapsed state is stored as `psi` if exactly one measurement is performed. In the case of a single qubit measurement `value` is of length `repetitions` and contains all the results of this projection. Otherwise `value` is of length 2^nbits and it contains the counts how often each state has been obtained.

Examples

```
## measure the separate bits
x <- H(1) * (H(2) * qstate(nbits=2))
summary(measure(x, bit=1))
hist(measure(x, rep=100))
```

noise — *A noise gate*

Description

A noise gate

Usage

```
noise(bit, p = 1, error = "any", type = "ERR", args = list())
```

Arguments

`bit`	integer or integer array. The bit to which to apply the gate. If an array is provided, the gate will be applied randomly to one of the bits only.
`p`	probability with which noise is applied
`error`	one of "X", "Y", "Z", "small" or "any". The model which the noise follows. Can be one of the Pauli matrices (X,Y,Z), a random SU(2)-matrix with a small deviation `sigma` from the identity ("small") or an arbitrary, uniformly sampled, SU(2)-matrix ("any").
`type`	a character vector representing the type of gate
`args`	a list of further arguments passed to specific error models. For `error="small"` the standard deviation `sigma` has to be provided here (default=1).

Value

An S4 class 'sqgate' object is returned

Examples

```
x <- noise(1, error="X") * qstate(nbits=2)
x
y <- noise(2, p=0.5) * x
y
z <- noise(2, error="small", args=list(sigma=0.1)) * x
z
```

`normalise` *normalise*

Description

Normalises a complex vector to 1

Usage

```
normalise(x)
```

Arguments

`x`	complex valued vector

Value

Returns the normalised complex valued vector

`phase_estimation` *phase_estimation*

Description

phase estimation algorithm

Usage

```
phase_estimation(bitmask, FUN, x, ...)
```

Arguments

`bitmask`	integer. Vector of qubits for the t qubit wide register needed for the phase estimation
`FUN`	a function implementing the controlled application of a unitary operator U to the power 2^(j-1) to the state x. It's first argument must be the control qubit 'c', the second the integer 'j' and the third the state 'x'. Additional parameters can be passed via '...'.
`x`	a 'qstate' object
`...`	additional parameter to be passed on to 'FUN'

Examples

```
## NOT^k = Id if k even
cnotwrapper <- function(c, j, x, t) {
  if(j == 1) return(CNOT(c(c, t)) * x)
  return(Id(t) * x)
}
x <- X(1) * qstate(3)
## X has eigenvalues lambda=1 and lambda=-1
## thus phases 0 and 1/2
x <- phase_estimation(bitmas=c(2:3), FUN=cnotwrapper, x=x, t=1)
x
```

---

plot,qstate,missing-method

plot-qstate

Description

Plots a circuit corresponding to a qstate object

Usage

```
## S4 method for signature 'qstate,missing'
plot(x, y, ...)
```

Arguments

x	qstate object
y	not used here
...	additional parameters to be passed on

Value

nothing is returned, but a plot created

Examples

```
x <- qstate(2)
y <- H(1) * x
z <- CNOT(c(1,2)) * y
plot(z)
```

qft *qft*

Description

performs the quantum Fourier Trafo on the qstate x and the specified list of qubits.

Usage

```
qft(x, inverse = FALSE, bits)
```

Arguments

x	qstate
inverse	boolean. If 'TRUE', perform inverse transform
bits	integer. list of qubits to include in the trafo. if missing, bits=c(1:n) is assumed, with n the number of qubits in x.

Details

Quantum Fourier Trafo

The Fourier Trafo is defined as

$$|j>->1/sqrt(N)sum_k=0_1^N exp(2piijk/N)|k>$$

the inverse with the oposite sign in the exponential.

Value

a qstate object with the quantum Fourier trafo of input x.

Examples

```
x <- qstate(3)
y <- qft(x)
z <- qft(y, inverse=TRUE)
```

qsimulatR *The qsimulatR Package*

Description

A simulator for a quantum computer

Details

A quantum computer simulator framework. General single qubit gates and general controlled single qubit gates can be easily defined. For convenience, it currently directly provides most common gates (X, Y, Z, H, Z, S, T, Rx, Ry, Rz, CNOT, SWAP, toffoli or CCNOT, CSWAP). 'qsimulatR' supports plotting of circuits and is able to export circuits into IBM's 'Qiskit' python package, which can be run on IBM's real quantum hardware. 'qsimulatR' currently works for up to 24 qubits (a virtual restriction, which can be lifted).

Author(s)

Johann Ostemeyer, Carsten Urbach, `<urbach@hiskp.uni-bonn.de>`

qstate *The qstate class*

Description

This class represents a quantum state

Details

The qubits are counted from 1 to `nbits` starting with the least significant bit.

Slots

`nbits` The number of qubits

`coefs` The 2^nbits complex valued vector of coefficients

`basis` String or vector of strings. A single string will be interpreted as the `collapse`-parameter in `genComputationalBasis`. A vector of length 2^nbits yields the basis directly.

`noise` List containing the probability `p` some noise is applied to one of the `bits` after a gate application, the model `error` of this noise and further arguments `args` to be passed to the function `noise`. See function `noise` for details. The list `noise` can be generated with `genNoise`.

`circuit` List containing the number of non-quantum bits `ncbits` and a list of gates `gatelist` applied to the original state. Filled automatically as gates are applied, required for plotting.

Examples

```
x <- qstate(nbits=2)
x

x <- qstate(nbits=2, coefs=as.complex(sqrt(rep(0.25, 4))), basis=",")
x

x <- qstate(nbits=1, coefs=as.complex(sqrt(rep(0.5, 2))), basis=c("|dead>", "|alive>"))
x

x <- qstate(nbits=2, noise=genNoise(nbits=2, p=1))
Id(2) * x

x <- qstate(nbits=3, noise=genNoise(p=1, bits=1:2, error="small", sigma=0.1))
Id(2) * x
```

`Ri` *The Ri gate*

Description

The Ri gate

Usage

```
Ri(bit, i, sign = +1)
```

Arguments

`bit`	integer. The bit to which to apply the gate
`i`	integer
`sign`	integer

Details

Implements the gate (1 0) (0 exp(+-2*pi*1i/2^i))

If 'sign < 0', the inverse of the exponential is used. This gate is up to global phase identical with the 'Rz' gate with specific values of the angle.

Value

An S4 class 'sqgate' object is returned

Examples

```
x <- X(1) * qstate(nbits=2)
z <- Ri(1, i=2) * x
z
```

Rx *The Rx gate*

Description

The Rx gate

Usage

```
Rx(bit, theta = 0)
```

Arguments

bit	integer. The bit to which to apply the gate
theta	numeric. angle

Value

An S4 class 'sqgate' object is returned

Examples

```
x <- qstate(nbits=2)
z <- Rx(1, pi/4) * x
z
```

Ry *The Ry gate*

Description

The Ry gate

Usage

```
Ry(bit, theta = 0)
```

Arguments

`bit`	integer. The bit to which to apply the gate
`theta`	numeric. angle

Value

An S4 class 'sqgate' object is returned

Examples

```
x <- qstate(nbits=2)
z <- Ry(1, pi/4) * x
z
```

`Rz` *The Rz gate*

Description

The Rz gate

Usage

```
Rz(bit, theta = 0)
```

Arguments

`bit`	integer. The bit to which to apply the gate
`theta`	numeric. angle

Value

An S4 class 'sqgate' object is returned

Examples

```
x <- qstate(nbits=2)
z <- Rz(1, pi/4) * x
z
```

S *The S gate*

Description

The S gate

Usage

```
S(bit)
```

Arguments

bit	integer. The bit to which to apply the gate

Value

An S4 class 'sqgate' object is returned

Examples

```
x <- X(1) * qstate(nbits=2)
z <- S(1) * x
z
```

sqgate *A single qubit gate*

Description

This class represents a generic single qubit gate

Details

The qubits are counted from 1 to `nbits` starting with the least significant bit.

Slots

`bit` Integer. The single bit to act on.

`M` complex valued array. The 2x2 matrix representing the gate

`type` a character vector representing the type of gate

Examples

```
x <- qstate(nbits=2)
## application of the X (NOT) gate to bit 1
z <- sqgate(bit=1L, M=array(as.complex(c(0,1,1,0)), dim=c(2,2))) * x
z
```

---

summary.measurement *Summarize a quantum measurement*

Description

Summarize a quantum measurement

Usage

```
## S3 method for class 'measurement'
summary(object, ...)
```

Arguments

`object`	as returned by `measure`
`...`	Generic parameters to pass on, not used here.

Value

No return value.

SWAP *The SWAP gate*

Description

The SWAP gate

Usage

```
SWAP(bits = c(1, 2))
```

Arguments

`bits`	integer vector of length two, containing the bits to swap.

Value

An S4 class 'swapgate' object is returned swapgate *The SWAP gate*

Description

This class represents a generic SWAP gate

Slots

`bits` Integer vector of length 2. The two bits to swap.

Examples

```
x <- H(1) * qstate(nbits=2)
z <- SWAP(c(1,2)) * (H(1) * x)
```

Tgate *The Tgate gate*

Description

The Tgate gate

Usage

```
Tgate(bit)
```

Arguments

`bit` integer. The bit to which to apply the gate

Value

An S4 class 'sqgate' object is returned

Examples

```
x <- X(1)*qstate(nbits=2)
z <- Tgate(1) * x
z
```

truth.table *Method truth.table*

Description

Method truth.table

Usage

```
truth.table(e1, nbits, bits, ...)
```

Arguments

e1	gate to measure.
nbits	number of bits the gate acts on.
bits	optional vector of length `nbits` containing the qubit order in the gate.
...	additional parameters to passed be on to 'e1'

Details

calculates the quantum truth table of the gate `e1`. If a basis state is transformed to a superposition of basis states by the gate, the result is 'NA'.

Value

returns a data.frame containing the truth table. Each row corresponds to one input-output combination. Each column to one specific bit.

Examples

```
## truth table for a single bit gate
truth.table(X, 1)
## for a 2-bit gate
truth.table(CNOT, 2)
## for a 2-bit gate with reversed controll and target bits
truth.table(CNOT, bits=2:1)
## for a general controlled gate
truth.table(cqgate, 2, gate=H(2))
## for an arbitrary circuit (here a swap implementation using only CNOTs)
myswap <- function(bits){ function(x){ CNOT(bits) * (CNOT(rev(bits)) * (CNOT(bits) * x))}}
truth.table(myswap, 2)
```

X *The X gate*

Description

The X gate

Usage

```
X(bit)
```

Arguments

`bit`	integer. The bit to which to apply the gate

Value

An S4 class 'sqgate' object is returned

Examples

```
x <- qstate(nbits=2)
z <- X(1) * x
z
```

Y *The Y gate*

Description

The Y gate

Usage

```
Y(bit)
```

Arguments

`bit`	integer. The bit to which to apply the gate

Value

An S4 class 'sqgate' object is returned

Examples

```
x <- qstate(nbits=2)
z <- Y(1) * x
z
```

`Z` *The Z gate*

Description

The Z gate

Usage

```
Z(bit)
```

Arguments

`bit`	integer. The bit to which to apply the gate

Value

An S4 class 'sqgate' object is returned

Examples

```
x <- X(1) * qstate(nbits=2)
z <- Z(1) * x
z
```

Index

∗ **package**
qsimulatR, 24
*,ccnotgate,qstate-method, 3
*,ccqgate,qstate-method, 3
*,cnotgate,qstate-method, 4
*,cnqgate,qstate-method, 4
*,complex,qstate-method, 5
*,cqgate,qstate-method, 5
*,cswapgate,qstate-method, 6
*,matrix,qstate-method, 6
*,sqgate,qstate-method, 7
*,swapgate,qstate-method, 7

CCNOT, 8
ccnotgate, 8
ccnotgate-class (ccnotgate), 8
ccqgate, 9
ccqgate-class (ccqgate), 9
CNOT, 9
cnotgate, 10
cnotgate-class (cnotgate), 10
cnqgate, 10
cnqgate-class (cnqgate), 10
cqft, 11
cqgate, 12
cqgate-class (cqgate), 12
CSWAP, 12
cswapgate, 13
cswapgate-class (cswapgate), 13

export2qiskit, 13

fredkin (CSWAP), 12

genComputationalBasis, 14
genNoise, 15
genStateNumber, 15
genStateString, 16

H, 17
hist.measurement, 17

Id, 18
is.bitset, 18

measure, 19
measure,qstate-method (measure), 19

noise, 20
normalise, 21

phase_estimation, 21
plot (plot,qstate,missing-method), 22
plot,qstate,missing-method, 22

qft, 23
qsimulatR, 24
qstate, 24
qstate-class (qstate), 24

Ri, 25
Rx, 26
Ry, 26
Rz, 27

S, 28
sqgate, 28
sqgate-class (sqgate), 28
summary.measurement, 29
SWAP, 29
swapgate, 30
swapgate-class (swapgate), 30

Tgate, 30
toffoli (CCNOT), 8
truth.table, 31

X, 32

Y, 32

Z, 33