Package ‘QGameTheory’

June 12, 2020

Title Quantum Game Theory Simulator

Version 0.1.2

Author Indranil Ghosh

Maintainer Indranil Ghosh <indranilg49@gmail.com>

Imports dplyr, RColorBrewer, R.utils

Depends R(>= 3.4)

Description General purpose toolbox for simulating quantum versions of game theoretic models (Flitney and Abbott 2002) <arXiv:quant-ph/0208069>. Quantum (Nielsen and Chuang 2010, ISBN:978-1-107-00217-3) versions of models that have been handled are: Penny Flip Game (David A. Meyer 1998) <arXiv:quant-ph/9804010>, Prisoner's Dilemma (J. Orlin Grabbe 2005) <arXiv:quant-ph/0506219>, Two Person Duel (Flitney and Abbott 2004) <arXiv:quant-ph/0305058>, Battle of the Sexes (Nawaz and Toor 2004) <arXiv:quant-ph/0110096>, Hawk and Dove Game (Nawaz and Toor 2010) <arXiv:quant-ph/0108075>, Newcomb's Paradox (Piotrowski and Sladkowski 2002) <arXiv:quant-ph/0202074> and Monty Hall Problem (Flitney and Abbott 2002) <arXiv:quant-ph/0109035>.

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/indrag49/QGameTheory

BugReports https://github.com/indrag49/QGameTheory/issues

LazyData true

RoxygenNote 7.1.0

NeedsCompilation no

Repository CRAN

Date/Publication 2020-06-12 08:20:03 UTC

R topics documented:

Bell *Bell States*

Description

The function builds one of the four Bell states, according to the input qubits

Usage

```
Bell(qubit1, qubit2)
```

Arguments

qubit1	1st input qubit
qubit2	2nd input qubit

Value

One of the Bell states as a vector depending on the input qubits.

References

https://en.wikipedia.org/wiki/Bell_state
https://books.google.co.in/books?id=66TgFp2YqrAC&pg=PA25&redir_esc=y

Examples

```
init()
Bell(Q$Q0, Q$Q0)
Bell(Q$Q0, Q$Q1)
Bell(Q$Q1, Q$Q0)
Bell(Q$Q1, Q$Q1)
```

CNOT *CNOT gate*

Description

This function operates the CNOT gate on a conformable input matrix/vector

Usage

```
CNOT(n)
```

Arguments

n   A vector/matrix

Value

A matrix or a vector after performing the CNOT gate operation on a conformable input matrix or a vector.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf

Examples

```
init()
CNOT(Q$I4)
CNOT(Q$Q11)
```

---

col_count   *Number of columns of a vector/matrix*

Description

This function counts the number of columns of a vector or a matrix

Usage

```
col_count(M)
```

Arguments

M   A vector/matrix

Value

An integer that gives the number of columns in a vector or a matrix.

Examples

```
init()
col_count(Q$Q11)
col_count(Q$lambda4)
col_count(Q$I2)
```

`Fredkin` *Fredkin Gate*

Description

This function operates the Fredkin gate on a conformable input matrix/vector

Usage

```
Fredkin(n)
```

Arguments

n	A vector/matrix

Value

A matrix or a vector after performing the Fredkin gate operation on a conformable input matrix or a vector.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf

Examples

```
init()
Fredkin(Q$I8)
Fredkin(Q$Q110)
```

`Hadamard` *Hadamard Gate*

Description

This function operates the Hadamard gate on a conformable input matrix/vector

Usage

```
Hadamard(n)
```

Arguments

n	A vector/matrix

Value

A matrix or a vector after performing the Hadamard operation on a conformable input matrix or a vector.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf

Examples

```
init()
Hadamard(Q$Q0)
Hadamard(Q$I2)
Hadamard(Hadamard(Q$Q1))
```

IDSDS *Iterated Deletion of Strictly Dominated Strategies algorithm*

Description

This function applies the IDSDS algorithm to result in the equilibrium strategies based on the rationaility of the players. The input parameters are equal dimensional payoff matrices for the first and the second players.

Usage

```
IDSDS(P1, P2)
```

Arguments

P1	Payoff matrix to Alice
P2	Payoff matrix to Bob

Value

A list consisting of the equilibrium strategies based on the rationality of the players by application of the IDSDS algorithm on `P1` and `P2`.

References

https://arxiv.org/abs/1512.06808
https://en.wikipedia.org/wiki/Strategic_dominance

Examples

```
init()
Alice <- matrix(c(8, 0, 3, 3, 2, 4, 2, 1, 3), ncol=3, byrow=TRUE)
Bob <- matrix(c(6, 9, 8, 2, 1, 3, 8, 5, 1), ncol=3, byrow=TRUE)
IDSDS(Alice, Bob)
```

---

init *Initialization*

Description

Builds the parameters in the required environment after initialization

Usage

```
init()
```

Value

No return value, generates the required variables/parameters.

References

https://arxiv.org/pdf/Quant-ph/0512125.pdf
https://arxiv.org/pdf/0910.4222.pdf
https://arxiv.org/pdf/Quant-ph/9703032.pdf
https://en.wikipedia.org/wiki/Quantum_computing
https://en.wikipedia.org/wiki/Qubit
https://en.wikipedia.org/wiki/Qutrit
https://en.wikipedia.org/wiki/Clebsch%E2%80%93Gordan_coefficients_for_SU(3)

Examples

```
init()
Q$Q110
Q$Qt12
Q$Q_minus
Q$lambda4
```

`levi_civita` *Levi-Civita symbol*

Description

This function computes the Levi-Civita symbol depending on the permutations of the three inputs, lying in 0 to 2

Usage

```
levi_civita(i, j, k)
```

Arguments

`i`	an integer 0, 1 or 2
`j`	an integer 0, 1 or 2
`k`	an integer 0, 1 or 2

Value

0, 1 or -1 after computing the Levi-Civita symbol depending on the permutations of the three inputs 0, 1 and 2

References

https://en.wikipedia.org/wiki/Levi-Civita_symbol

Examples

```
init()
levi_civita(0, 2, 1)
levi_civita(1, 2, 0)
levi_civita(1, 2, 1)
```

`NASH` *Nash Equilibrium*

Description

This function finds out the Nash equilibria of the 2-D payoff matrix for the players. The input parameters are equal dimensional payoff matrices for the first and the second players.

Usage

```
NASH(P1, P2)
```

Arguments

P1	Payoff matrix to Alice
P2	Payoff matrix to Bob

Value

The cell positons of the Nash equilibrium/equilibria as a dataframe from the payoff matrices of the players.

References

https://arxiv.org/abs/1512.06808
https://en.wikipedia.org/wiki/Nash_equilibrium

Examples

```
init()
Alice <- matrix(c(4, 3, 2, 4, 4, 2, 1, 0, 3, 5, 3, 5, 2, 3, 1, 3), ncol=4, byrow=TRUE)
Bob <- matrix(c(0, 2, 3, 8, 2, 1, 2, 2, 6, 5, 1, 0, 3, 2, 2, 3), ncol=4, byrow=TRUE)
NASH(Alice, Bob)
```

PayoffMatrix_QBOS *Quantum Battle of the Sexes game: Payoff Matrix*

Description

This function generates the payoff matrix for the Quantum Battle of Sexes game for all the four combinations of p and q. moves is a list of two possible strategies for each of the players and alpha,beta,gamma are the payoffs for the players corresponding to the choices available to them with the chain of inequalities, alpha>beta>gamma.

Usage

```
PayoffMatrix_QBOS(moves, alpha, beta, gamma)
```

Arguments

moves	a list of matrices
alpha	a number
beta	a number
gamma	a number

Value

The payoff matrices for the two players as two elements of a list.

References

https://arxiv.org/pdf/quant-ph/0506219.pdf
https://arxiv.org/pdf/quant-ph/0208069.pdf
https://arxiv.org/abs/quant-ph/0110096

Examples

```
init()
moves <- list(Q$I2, sigmaX(Q$I2))
PayoffMatrix_QBOS(moves, 5, 3, 1)
```

PayoffMatrix_QHawkDove *Quantum Hawk and Dove game: Payoff Matrix*

Description

This function generates the payoff matrix for the Quantum Hawk and Dove game for all the four combinations of `p` and `q`. `moves` is a list of two possible strategies for each of the players and `v,j,D` are the value of resource, cost of injury and cost of displaying respectively.

Usage

```
PayoffMatrix_QHawkDove(moves, v, j, D)
```

Arguments

moves	a list of matrices
v	a number
j	a number
D	a number

Value

The payoff matrices for the two players as two elements of a list.

References

https://arxiv.org/pdf/quant-ph/0506219.pdf
https://arxiv.org/pdf/quant-ph/0208069.pdf
https://arxiv.org/pdf/quant-ph/0108075.pdf

Examples

```
init()
moves <- list(Q$I2, sigmaX(Q$I2))
PayoffMatrix_QHawkDove(moves, 50, -100, -10)
```

PayoffMatrix_QPD *Quantum Prisoner's Dilemma game: Payoff Matrix*

Description

This function generates the payoff matrix for the Quantum Prisoner's Dilemma game . `moves` is a list of the possible strategies for each of the players and `w,x,y,z` are the payoffs for the players corresponding to the choices available to them with the chain of inequalities, `z>w>x>y`. This function also plots the probability distribution plots of the qubits for all the possible combinations of the strategies of the players.

Usage

```
PayoffMatrix_QPD(moves, w, x, y, z)
```

Arguments

`moves`	a list of matrices
`w`	a number
`x`	a number
`y`	a number
`z`	a number

Value

The payoff matrices for the two players as two elements of a list.

References

https://arxiv.org/pdf/quant-ph/0506219.pdf
https://arxiv.org/pdf/quant-ph/0208069.pdf
https://arxiv.org/pdf/quant-ph/0004076.pdf

Examples

```
init()
moves <- list(Q$I2, sigmaX(Q$I2), Hadamard(Q$I2), sigmaZ(Q$I2))
PayoffMatrix_QPD(moves, 3, 1, 0, 5)
```

Phase *Phase Gate*

Description

This function operates the Phase gate on a conformable input matrix/vector

Usage

```
Phase(n)
```

Arguments

n	a vector/matrix

Value

A matrix or a vector after performing the Phase gate operation on a conformable input matrix or a vector.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf

Examples

```
init()
Phase(Q$I2)
Phase(Q$Q_plus)
```

PhaseDagger *Hermitian Transpose of the Phase Gate*

Description

This function operates the hermitian transpose of the Phase gate on a conformable input matrix/vector

Usage

```
PhaseDagger(n)
```

Arguments

n	a vector/matrix

Value

A matrix or a vector after performing the operation of the hermitian transpose of the Phase gate on a conformable input matrix or a vector.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf

Examples

```
init()
Conj(t(Phase(Q$I2)))==PhaseDagger(Q$I2)
PhaseDagger(Q$Q_plus)
```

QBOS *Quantum Battle of the Sexes game*

Description

This function returns the expected payoffs to Alice and Bob with respect to the probabilities `p` and `q`. `p+q` should equal 1 and `moves` is a list of two possible strategies for each of the players and `alpha,beta,gamma` are the payoffs for the players corresponding to the choices available to them with the chain of inequalities, `alpha>beta>gamma`.

Usage

```
QBOS(p, q, moves, alpha, beta, gamma)
```

Arguments

p	a real number between 0 and 1 including the end points
q	a real number between 0 and 1 including the end points
moves	alist of matrices
alpha	a number
beta	a number
gamma	a number

Value

A vector consisting of the Payoffs to Alice and Bob as its two elements depending on the inputs.

References

https://arxiv.org/pdf/quant-ph/0506219.pdf
https://arxiv.org/pdf/quant-ph/0208069.pdf
https://arxiv.org/abs/quant-ph/0110096

Examples

```
init()
moves <- list(Q$I2, sigmaX(Q$I2))
QBOS(0, 1, moves, 5, 3, 1)
QBOS(1, 1, moves, 5, 3, 1)
QBOS(0.5, 0.5, moves, 5, 3, 1)
```

QDuelsPlot1 *Quantum Two Person Duel game*

Description

This function helps us to plot Alice's and Bob's expected payoffs as functions of alpha1 and alpha2. Psi is the initial state of the quantum game, n is the number of rounds, a is the probability of Alice missing the target, b is the probability of Bob missing the target, and alpha1,alpha2,beta1,beta2 are arbitrary phase factors that lie in -pi to pi that control the outcome of a poorly performing player.

Usage

```
QDuelsPlot1(Psi, n, a, b, beta1, beta2)
```

Arguments

Psi	a vector representing the initial quantum state
n	an integer
a	a number
b	a number
beta1	a number
beta2	a number

Value

No return value, plots Alice's and Bob's expected payoffs as functions of alpha1 and alpha2.

References

https://arxiv.org/pdf/quant-ph/0506219.pdf
https://arxiv.org/pdf/quant-ph/0208069.pdf
https://arxiv.org/pdf/quant-ph/0305058.pdf

Examples

```
init()
QDuelsPlot1(Q$Q10, 2, 0.66666, 0.5, 0.2, 0.8)
```

`QDuelsPlot2` *Quantum Two Person Duel game*

Description

This function helps us to plot Alice's and Bob's expected payoffs as functions of the number of rounds `n` played in a repeated quantum duel. `Psi` is the initial state of the quantum game, `n` is the number of rounds, `a` is the probability of Alice missing the target, `b` is the probability of Bob missing the target, and `alpha1,alpha2,beta1,beta2` are arbitrary phase factors that lie in -pi to pi that control the outcome of a poorly performing player.

Usage

```
QDuelsPlot2(Psi, n, a, b, alpha1, alpha2, beta1, beta2)
```

Arguments

`Psi`	a vector representing the initial quantum state
`n`	an integer
`a`	a number
`b`	a number
`alpha1`	a number
`alpha2`	a number
`beta1`	a number
`beta2`	a number

Value

No return value, plots Alice's and Bob's expected payoffs as functions of the number of rounds `n` played in a repeated quantum duel.

References

https://arxiv.org/pdf/quant-ph/0506219.pdf
https://arxiv.org/pdf/quant-ph/0208069.pdf
https://arxiv.org/pdf/quant-ph/0305058.pdf

Examples

```
init()
QDuelsPlot2(Q$Q01, 10, 0.66666, 0.5, -pi/2, pi/4, 0.6, 0.4)
```

QDuelsPlot3 *Quantum Two Person Duel game*

Description

This function helps us to plot the improvement in Alice's expected payoff as a function of `a` and `b`, if Alice chooses to fire at the air in her second shot, in a two round game. `Psi` is the initial state of the quantum game, `n` is the number of rounds, `a` is the probability of Alice missing the target, `b` is the probability of Bob missing the target, and `alpha1,alpha2,beta1,beta2` are arbitrary phase factors that lie in -pi to pi that control the outcome of a poorly performing player.

Usage

```
QDuelsPlot3(Psi, alpha1, alpha2)
```

Arguments

`Psi`	a vector representing the initial quantum state
`alpha1`	a number
`alpha2`	a number

Value

No return value, plots the improvement in Alice's expected payoff as a function of `a` and `b`, if Alice chooses to fire at the air in her second shot, in a two round game.

References

https://arxiv.org/pdf/quant-ph/0506219.pdf
https://arxiv.org/pdf/quant-ph/0208069.pdf
https://arxiv.org/pdf/quant-ph/0305058.pdf

Examples

```
init()
Qs <- (Q$Q0+Q$Q1)/sqrt(2)
Psi <- kronecker(Q$Q1, Qs)
QDuelsPlot3(Psi, pi/3, pi/6)
```

QDuelsPlot4 *Quantum Two Person Duel game*

Description

This function helps us to plot the improvement in Bob's expected payoff as a function of a and b, if Bob chooses to fire at the air in her second shot, in a two round game. Psi is the initial state of the quantum game, n is the number of rounds, a is the probability of Alice missing the target, b is the probability of Bob missing the target, and alpha1,alpha2,beta1,beta2 are arbitrary phase factors that lie in -pi to pi that control the outcome of a poorly performing player.

Usage

```
QDuelsPlot4(Psi, alpha1, alpha2)
```

Arguments

Psi	a vector representing the initial quantum state
alpha1	a number
alpha2	a number

Value

No return value, plots the improvement in Bob's expected payoff as a function of a and b, if Bob chooses to fire at the air in her second shot, in a two round game.

References

https://arxiv.org/pdf/quant-ph/0506219.pdf
https://arxiv.org/pdf/quant-ph/0208069.pdf
https://arxiv.org/pdf/quant-ph/0305058.pdf

Examples

```
init()
Qs <- (Q$Q0+Q$Q1)/sqrt(2)
Psi <- kronecker(Q$Q1, Qs)
QDuelsPlot4(Psi, pi/3, pi/6)
```

QDuels_Alice_payoffs *Quantum Two Person Duel game*

Description

This function returns the expected payoff to Alice for three possible cases for the Quantum Duel game:

1. The game is continued for n rounds and none of the players shoots at the air.
2. The game is continued for 2 rounds and Alice shoots at the air in her second round.
3. The game is continued for 2 rounds and Bob shoots at the air in her second round.

`Psi` is the initial state of the quantum game, `n` is the number of rounds, `a` is the probability of Alice missing the target, `b` is the probability of Bob missing the target, and `alpha1,alpha2,beta1,beta2` are arbitrary phase factors that lie in -pi to pi that control the outcome of a poorly performing player.

Usage

```
QDuels_Alice_payoffs(Psi, n, a, b, alpha1, alpha2, beta1, beta2)
```

Arguments

Psi	a vector representing the initial quantum state
n	an integer
a	a number
b	a number
alpha1	a number
alpha2	a number
beta1	a number
beta2	a number

Value

A list consisting of the payoff value to Alice depending on three situations of the quantum duel game: 1) The game is continued for n rounds and none of the players shoots at the air, 2) The game is continued for 2 rounds and Alice shoots at the air in her second round and 3) The game is continued for 2 rounds and Bob shoots at the air in her second round.

References

https://arxiv.org/pdf/quant-ph/0506219.pdf
https://arxiv.org/pdf/quant-ph/0208069.pdf
https://arxiv.org/pdf/quant-ph/0305058.pdf

Examples

```
init()
QDuels_Alice_payoffs(Q$Q11, 5, 0.666666, 0.5, 0, 0, 0.2, 0.7)
Qs <- (Q$Q0+Q$Q1)/sqrt(2)
Psi <- kronecker(Qs, Qs)
QDuels_Alice_payoffs(Psi, 5, 0.666666, 0.5, 0, 0, 0.2, 0.7)
```

QDuels_Bob_payoffs *Quantum Two Person Duel game*

Description

This function returns the expected payoff to Bob for three possible cases for the Quantum Duel game:

1. The game is continued for n rounds and none of the players shoots at the air.
2. The game is continued for 2 rounds and Alice shoots at the air in her second round.
3. The game is continued for 2 rounds and Bob shoots at the air in her second round.

Psi is the initial state of the quantum game, n is the number of rounds, a is the probability of Alice missing the target, b is the probability of Bob missing the target, and alpha1,alpha2,beta1,beta2 are arbitrary phase factors that lie in -pi to pi that control the outcome of a poorly performing player.

Usage

```
QDuels_Bob_payoffs(Psi, n, a, b, alpha1, alpha2, beta1, beta2)
```

Arguments

Psi	a vector representing the initial quantum state
n	an integer
a	a number
b	a number
alpha1	a number
alpha2	a number
beta1	a number
beta2	a number

Value

A list consisting of the payoff value to Bob depending on three situations of the quantum duel game: 1) The game is continued for n rounds and none of the players shoots at the air, 2) The game is continued for 2 rounds and Alice shoots at the air in her second round and 3) The game is continued for 2 rounds and Bob shoots at the air in her second round.

References

https://arxiv.org/pdf/quant-ph/0506219.pdf
https://arxiv.org/pdf/quant-ph/0208069.pdf
https://arxiv.org/pdf/quant-ph/0305058.pdf

Examples

```
init()
QDuels_Bob_payoffs(Q$Q11, 5, 0.666666, 0.5, 0, 0, 0.2, 0.7)
Qs <- (Q$Q0+Q$Q1)/sqrt(2)
Psi <- kronecker(Qs, Qs)
QDuels_Bob_payoffs(Psi, 5, 0.666666, 0.5, 0, 0, 0.2, 0.7)
```

QFT *Quantum Fourier Transform*

Description

This function performs Quantum Fourier Transform for a given state `|y>` from the computational basis to the Fourier basis.

Usage

```
QFT(y)
```

Arguments

`y`	an integer

Value

A vector representing the Quantum Fourier transformation of the state `|y>` from the computational basis to the Fourier basis.

References

https://arxiv.org/pdf/quant-ph/0506219.pdf
https://books.google.co.in/books?id=66TgFp2YqrAC&pg=PA25&redir_esc=y
https://en.wikipedia.org/wiki/Quantum_Fourier_transform

Examples

```
init()
QFT(5)
```

QHawkDove *Quantum Hawk and Dove game*

Description

This function returns the expected payoffs to Alice and Bob with respect to the probabilities p and q. p+q should equal 1 and moves is a list of two possible strategies for each of the players and v,j,D are the value of resource, cost of injury and cost of displaying respectively.

Usage

```
QHawkDove(p, q, moves, v, j, D)
```

Arguments

p	a real number between 0 and 1 including the end points
q	a real number between 0 and 1 including the end points
moves	a list of matrics
v	a number
j	a number
D	a number

Value

A vector consisting of the expected payoffs to Alice and Bob as its elements calculated according to the probabilities p and q provided as inputs.

References

https://arxiv.org/pdf/quant-ph/0506219.pdf
https://arxiv.org/pdf/quant-ph/0208069.pdf
https://arxiv.org/pdf/quant-ph/0108075.pdf

Examples

```
init()
moves <- list(Q$I2, sigmaX(Q$I2))
QHawkDove(0, 1, moves, 50, -100, -10)
QHawkDove(0, 0, moves, 50, -100, -10)
```

`QMeasure` *Measurement*

Description

This function performs a projective measurement of a quantum state `n`, in the computational basis and plots the corresponding probability distributions of the qubits.

Usage

```
QMeasure(n)
```

Arguments

`n`	a vector representing a quantum state

Value

No return value, plots the probability distributions of the qubits after performing a projective measurement of a quantum state `n`.

References

https://books.google.co.in/books?id=66TgFp2YqrAC&pg=PA25&redir_esc=y
https://en.wikipedia.org/wiki/Measurement_in_quantum_mechanics

Examples

```
init()
QMeasure(Q$Q10110)
```

`QMontyHall` *Quantum Monty Hall Problem*

Description

This function simulates the quantum version of the Monty Hall problem, by taking in `Psi_in` as the initial quantum state of the game, `gamma` lying in 0 to pi/2, `Ahat` and `Bhat` as the choice operators in SU(3) for Alice and Bob respectively as the inputs. It returns the expected payoffs to Alice and Bob after the end of the game.

Usage

```
QMontyHall(Psi_in, gamma, Ahat, Bhat)
```

Arguments

`Psi_in`	a vector representing the initial quantum state
`gamma`	a number between 0 and pi/2 including the end points
`Ahat`	a matrix lying in SU(3)
`Bhat`	a matrix lying in SU(3)

Value

A vector consisting of the expected payoffs to Alice and Bob as its elements depending on the input parameters.

References

https://arxiv.org/pdf/quant-ph/0506219.pdf
https://arxiv.org/pdf/quant-ph/0208069.pdf
https://arxiv.org/pdf/quant-ph/0109035.pdf

Examples

```
init()
Psi_in <- kronecker(Q$Qt0, (Q$Qt00+Q$Qt11+Q$Qt22)/sqrt(3))
QMontyHall(Psi_in, pi/4, Q$Identity3, Q$Hhat)
```

`QNewcomb` *Quantum Newcomb's Paradox*

Description

This function simulates the quantum version of the Newcomb's Paradox by taking in the choice of the qubit |0> or |1> by the supercomputer `Omega` and the probability `'probability'` with which Alice plays the spin flip operator. It returns the final state of the quantum game along with plotting the probability densities of the qubits of the final state after measurement.

Usage

```
QNewcomb(Omega, probability)
```

Arguments

`Omega`	|0> or |1>
`probability`	a real number between 0 and 1 including the end points

Value

The final state of the quantum game as a vector along with plotting the probability densities of the qubits of the final state after measurement.

References

https://arxiv.org/pdf/quant-ph/0506219.pdf
https://arxiv.org/pdf/quant-ph/0208069.pdf
https://arxiv.org/pdf/quant-ph/0202074.pdf

Examples

```
init()
QNewcomb(Q$Q0, 0)
QNewcomb(Q$Q1, 0)
QNewcomb(Q$Q1, 0.7)
```

QPD *Quantum Prisoner's Dilemma game*

Description

This function returns the expected payoffs to Alice and Bob, with the strategy moves by Alice and Bob as two of the inputs. `w,x,y,z` are the payoffs to the players corresponding to the choices available to them with the chain of inequalities, `z>w>x>y`. This function also plots the probability distribution plots of the qubits for one of all the combinations of the strategies of the players.

Usage

```
QPD(U_Alice, U_Bob, w, x, y, z)
```

Arguments

U_Alice	a matrix lying in SU(2)
U_Bob	a matrix lying in SU(2)
w	a number
x	a number
y	a number
z	a number

Value

A vector consisting of the expected payoffs to Alice and Bob as its elements according to the strategies played by Alice and Bob and also the payoff values.

References

https://arxiv.org/pdf/quant-ph/0506219.pdf
https://arxiv.org/pdf/quant-ph/0208069.pdf
https://arxiv.org/pdf/quant-ph/0004076.pdf

Examples

```
init()
QPD(Hadamard(Q$I2), sigmaZ(Q$I2), 3, 1, 0, 5)
```

QPennyFlip *Quantum Penny Flip game*

Description

This function simulates the Quantum Penny Flip game by taking in the initial state of the game that is set by Alice and the strategies available to Alice and Bob. It returns the final state of the game along with the plot of the probability distribution of the qubits after measurement of the final state.

Usage

```
QPennyFlip(initial_state, strategies_Alice, strategies_Bob)
```

Arguments

`initial_state`	a vector representing the initial quantum state
`strategies_Alice`	a matrix lying in SU(2)
`strategies_Bob`	a matrix lying in SU(2)

Value

The final state of the game along with the plot of the probability distribution of the qubits after measurement of the final state by taking in the initial state of the game that is set by Alice and the strategies available to Alice and Bob as the inputs.

References

https://arxiv.org/pdf/quant-ph/0506219.pdf
https://arxiv.org/pdf/quant-ph/0208069.pdf
https://arxiv.org/pdf/quant-ph/9804010.pdf

Examples

```
init()
psi <- (u+d)/sqrt(2)
S1 <- sigmaX(Q$I2)
S2 <- Q$I2
H <- Hadamard(Q$I2)
SA <- list(S1, S2)
SB <- list(H)
QPennyFlip(psi, SA,SB)
```

`row_count` *Number of rows of a vector/matrix*

Description

This function counts the number of rows of a vector or a matrix

Usage

```
row_count(M)
```

Arguments

`M`	A vector/matrix

Value

An integer that gives the number of rows in a vector or a matrix.

Examples

```
init()
row_count(Q$Q01)
row_count(Q$lambda5)
row_count(Q$Qt12)
```

`Rx` *Rotation operation about x-axis of the Bloch sphere*

Description

This function operates the Rotation gate about the x-axis of the Bloch sphere by an angle `theta` on a conformable input matrix `n`.

Usage

```
Rx(n, theta)
```

Arguments

`n`	a vector/matrix
`theta`	an angle

Value

A vector or a matrix after operating the Rotation gate about the x-axis of the Bloch sphere, by an angle `theta`, on a conformable input matrix or a vector `n`

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf
http://www.physics.udel.edu/~msafrono/650/Lecture%204%20-%205.pdf

Examples

```
init()
Rx(Q$Q0, pi/6)
```

Ry — *Rotation operation about y-axis of the Bloch sphere*

Description

This function operates the Rotation gate about the y-axis of the Bloch sphere by an angle `theta` on a conformable input matrix `n`.

Usage

```
Ry(n, theta)
```

Arguments

`n`	a vector/matrix
`theta`	an angle

Value

A vector or a matrix after operating the Rotation gate about the y-axis of the Bloch sphere, by an angle `theta`, on a conformable input matrix or a vector `n`.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf
http://www.physics.udel.edu/~msafrono/650/Lecture%204%20-%205.pdf

Examples

```
init()
Ry(Q$Q1, pi/3)
```

Rz *Rotation operation about z-axis of the Bloch sphere*

Description

This function operates the Rotation gate about the z-axis of the Bloch sphere by an angle `theta` on a conformable input matrix `n`.

Usage

```
Rz(n, theta)
```

Arguments

n	a vector/matrix
theta	an angle

Value

A vector or a matrix after operating the Rotation gate about the z-axis of the Bloch sphere, by an angle `theta`, on a conformable input matrix or a vector `n`.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf
http://www.physics.udel.edu/~msafrono/650/Lecture%204%20-%205.pdf

Examples

```
init()
Rz(Q$Q1, pi)
```

sigmaX *Pauli-X gate*

Description

This function operates the Pauli-X gate on a conformable input matrix or a vector.

Usage

```
sigmaX(n)
```

Arguments

n	a vector/matrix

Value

A matrix or a vector after performing the Pauli-X gate operation on a conformable input matrix or a vector.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf
http://www.physics.udel.edu/~msafrono/650/Lecture%204%20-%205.pdf

Examples

```
init()
sigmaX(Q$I2)
sigmaX(Hadamard(Q$I2))
sigmaX(Q$Q1)
```

sigmaY *Pauli-Y gate*

Description

This function operates the Pauli-Y gate on a conformable input matrix or a vector.

Usage

```
sigmaY(n)
```

Arguments

n	a vector/matrix

Value

A matrix or a vector after performing the Pauli-Y gate operation on a conformable input matrix or a vector.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf
http://www.physics.udel.edu/~msafrono/650/Lecture%204%20-%205.pdf

Examples

```
init()
sigmaY(Q$I2)
sigmaY(Hadamard(Q$I2))
sigmaY(Q$Q0)
```

---

sigmaZ *Pauli-Z gate*

Description

This function operates the Pauli-Z gate on a conformable input matrix or a vector.

Usage

```
sigmaZ(n)
```

Arguments

n	a vector/matrix

Value

A matrix or a vector after performing the Pauli-Z gate operation on a conformable input matrix or a vector.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf
http://www.physics.udel.edu/~msafrono/650/Lecture%204%20-%205.pdf

Examples

```
init()
sigmaZ(Q$I2)
sigmaZ(Hadamard(Q$I2))
sigmaZ(Q$Q0)
```

SWAP *SWAP gate*

Description

This function operates the SWAP gate on a conformable input matrix or a vector.

Usage

```
SWAP(n)
```

Arguments

n	a vector/matrix

Value

A matrix or a vector after performing the SWAP gate operation on a conformable input matrix or a vector.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf

Examples

```
init()
SWAP(Q$I4)
SWAP(Q$Q10)
```

T *T gate*

Description

This function operates the T gate on a conformable input matrix or a vector.

Usage

```
T(n)
```

Arguments

n	a vector/matrix

Value

A matrix or a vector after performing the T gate operation on a conformable input matrix or a vector.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf

Examples

```
init()
T(Q$I2)
T(Q$Q_minus)
```

TDagger *Hermitian Transpose of the T gate*

Description

This function operates the hermitian transpose of the T gate on a conformable input matrix or a vector.

Usage

```
TDagger(n)
```

Arguments

`n` a vector/matrix

Value

A matrix or a vector after performing the operation of the hermitian transpose of the T gate on a conformable input matrix or a vector.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf

Examples

```
init()
TDagger(Q$I2)
TDagger(Q$Q_plus)
```

Toffoli *Toffoli gate*

Description

This function operates the Toffoli gate on a conformable input matrix or a vector.

Usage

```
Toffoli(n)
```

Arguments

`n` a vector/matrix

Value

A matrix or a vector after performing the Toffoli gate operation on a conformable input matrix or a vector.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf

Examples

```
init()
Toffoli(Q$I8)
Toffoli(Q$Q010)
```

`Walsh` *Walsh-Hadamard gate*

Description

This function operates the Walsh-Hadamard gate on a conformable input matrix or a vector.

Usage

```
Walsh(n)
```

Arguments

n	a vector/matrix

Value

A matrix or a vector after performing the Walsh-Hadamard gate operation on a conformable input matrix or a vector.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf
https://arxiv.org/pdf/quant-ph/0506219.pdf
https://en.wikipedia.org/wiki/Hadamard_transform

Examples

```
init()
Walsh(Q$I2)
Walsh(Q$Q0)
```

`Walsh16` *Walsh-Hadamard gate*

Description

This function operates the Walsh-16 gate on a conformable input matrix or a vector.

Usage

```
Walsh16(n)
```

Arguments

`n`	a vector/matrix

Value

A matrix or a vector after performing the Walsh-16 gate operation on a conformable input matrix or a vector.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf
https://arxiv.org/pdf/quant-ph/0506219.pdf
https://en.wikipedia.org/wiki/Hadamard_transform

Examples

```
init()
Walsh16(Q$I16)
Walsh16(Q$Q1001)
```

`Walsh32` *Walsh-Hadamard gate*

Description

This function operates the Walsh-32 gate on a conformable input matrix or a vector.

Usage

```
Walsh32(n)
```

Arguments

`n`	a vector/matrix

Value

A matrix or a vector after performing the Walsh-32 gate operation on a conformable input matrix or a vector.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf
https://arxiv.org/pdf/quant-ph/0506219.pdf
https://en.wikipedia.org/wiki/Hadamard_transform

Examples

```
init()
Walsh32(Q$I32)
Walsh32(Q$Q10011)
```

Walsh4 *Walsh-Hadamard gate*

Description

This function operates the Walsh-4 gate on a conformable input matrix or a vector.

Usage

```
Walsh4(n)
```

Arguments

`n`	a vector/matrix

Value

A matrix or a vector after performing the Walsh-4 gate operation on a conformable input matrix or a vector.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf
https://arxiv.org/pdf/quant-ph/0506219.pdf
https://en.wikipedia.org/wiki/Hadamard_transform

Examples

```
init()
Walsh4(Q$I4)
Walsh4(Q$Q10)
```

Walsh8 *Walsh-Hadamard gate*

Description

This function operates the Walsh-8 gate on a conformable input matrix or vector.

Usage

```
Walsh8(n)
```

Arguments

n	a vector/matrix

Value

A matrix or a vector after performing the Walsh-8 gate operation on a conformable input matrix or a vector.

References

https://en.wikipedia.org/wiki/Quantum_logic_gate
http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGates.pdf
https://arxiv.org/pdf/quant-ph/0506219.pdf
https://en.wikipedia.org/wiki/Hadamard_transform

Examples

```
init()
Walsh8(Q$I8)
Walsh8(Q$Q000)
```

Index