

Quack! - A quantum computer simulator for MATLAB

Peter P. Rohde

`rohde@physics.uq.edu.au`

`http://www.physics.uq.edu.au/people/rohde/`

Centre for Quantum Computer Technology, Department of Physics
University of Queensland, Brisbane, QLD 4072, Australia

November 8, 2005

1 Introduction

Quack! is a MATLAB package for simulating simple quantum circuits. All of the commands operate from within the MATLAB environment. Features include:

- Single qubit and Bell state preparation
- Single qubit unitary gates, with built in Pauli, rotation, Hadamard, phase and $\pi/8$ gates
- Controlled-NOT, controlled-sign, swap, Toffoli and general controlled-unitary gates
- Quantum Fourier Transforms
- Arbitrary depolarizing, bit-flip and phase-flip channels
- Single qubit and Bell measurements
- Classically controlled operations
- Calculate reduced density matrices for arbitrary sub-systems

Quack! comes with several simple example M-files which demonstrate usage and functionality.

2 What does *Quack!* stand for?

QUAntum <something beginning with 'CK'. Maybe the fragrance. I haven't decided yet.>!

3 Disclaimer

This software may be freely used, distributed and modified, provided that the original author is acknowledged.

4 Command reference

4.1 Initialization

COMMAND: `quack`

DESCRIPTION: Initialize *Quack!*. Must be called before any other commands.

COMMAND: `init_state(n)`

DESCRIPTION: Initializes an n qubit system. By default the system is initialized into the $|00\dots 0\rangle$ state.

4.2 State preparation

The following commands are used for preparing the initial state of the system. They must follow a call to `init_state`. Note that the state preparation commands can only be used to prepare the *initial* state of the system. They cannot be used midway during a simulation.

COMMANDS: `prepare_zero(n)`, `prepare_one(n)`, `prepare_plus(n)`,
`prepare_minus(n)`

DESCRIPTION: Prepare qubit n into the $|0\rangle$, $|1\rangle$, $|+\rangle$ or $|-\rangle$ state respectively.

COMMAND: `prepare_state(v,n)`

DESCRIPTION: Prepare qubit n into the arbitrary pure state specified by the 2-element row-vector v . The state will be denoted $|p\rangle$ (p for *pure*) in the circuit history.

COMMAND: `prepare_mixed(n)`

DESCRIPTION: Prepare qubit n into the completely mixed state ($\hat{I}/2$). The state will be denoted $|m\rangle$ (m for *mixed*) in the circuit history.

COMMANDS: `prepare_bell_00(n,m)`, `prepare_bell_01(n,m)`,
`prepare_bell_10(n,m)`, `prepare_bell_11(n,m)`

DESCRIPTION: Prepare qubits n and m into the 00, 01, 10 or 11 Bell states respectively, where $|\beta_{00}\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$, $|\beta_{01}\rangle = (|01\rangle + |10\rangle)/\sqrt{2}$, $|\beta_{10}\rangle = (|00\rangle - |11\rangle)/\sqrt{2}$ and $|\beta_{11}\rangle = (|01\rangle - |10\rangle)/\sqrt{2}$.

4.3 Gates

COMMANDS: `X(n)`, `Y(n)`, `Z(n)`, `H(n)`, `S(n)`, `T(n)`

DESCRIPTION: Performs the Pauli-X, Pauli-Y, Pauli-Z, Hadamard, phase or $\pi/8$ gate on qubit n respectively.

COMMANDS: `Rx(theta,n)`, `Ry(theta,n)`, `Rz(theta,n)`

DESCRIPTION: Applies the respective X, Y or Z rotation operation by angle $theta$ to qubit n .

COMMAND: `U(M,n)`

DESCRIPTION: Applies the general single-qubit unitary M to qubit n . M is a 2×2 matrix.

COMMANDS: `cnot(c,t)`, `cz(c,t)`, `swap(c,t)`

DESCRIPTION: Performs the CNOT, controlled-Z or swap gate between qubits c (control) and t (target) respectively.

COMMAND: `cu(U,c,t)`

DESCRIPTION: Performs a general controlled-unitary (U) operation between qubits c (control) and t (target), where U is a single-qubit unitary gate.

COMMAND: `toffoli(c1,c2,t)`

DESCRIPTION: Perform a Toffoli gate between qubits $c1$ (first control), $c2$ (second control) and t (target).

COMMAND: `qft(m,n)`

DESCRIPTION: Perform a Quantum Fourier Transform from bits m to n .

4.4 Probabilistic processes

COMMANDS: `depol_ch(p,n)`, `X_ch(p,n)`, `Z_ch(p,n)`

DESCRIPTION: Applies a p -probability depolarizing, bit-flip or phase-flip channel to qubit n respectively.

4.5 Measurement

COMMAND: `m = Z_measure(n)`

DESCRIPTION: Perform a Z-basis measurement on qubit n . Returns the measurement result (+1 or -1).

COMMAND: `Z_measure_forced(n,m)`

DESCRIPTION: Performs a Z-basis measurement on qubit n , forcing outcome m (+1 or -1).

COMMAND: `m = bell_measure(n,m)`
DESCRIPTION: Perform a Bell measurement on qubits n and m . Returns the measurement result ('00', '01', '10' or '11').

COMMAND: `bell_measure_forced(n,m,o)`
DESCRIPTION: Perform a Bell measurement on qubits n and m , forcing outcome o ('00', '01', '10' or '11').

4.6 Feedback

COMMAND: `m = get_dm`
DESCRIPTION: Returns the density matrix of the entire system.

COMMAND: `print_dm`
DESCRIPTION: Display the density matrix of the entire system.

COMMAND: `m = get_rdm(v)`
DESCRIPTION: Returns the reduced density matrix for the qubits specified by the vector v .

COMMAND: `print_rdm(v)`
DESCRIPTION: Display the reduced density matrix for the qubits specified by the vector v .

COMMAND: `v = get_bloch_vect(n)`
DESCRIPTION: Returns the Bloch vector for the reduced density matrix of qubit n . v is a 3×1 vector.

COMMAND: `print_bloch_vect(n)`
DESCRIPTION: Display the Bloch vector for the reduced density matrix of qubit n .

COMMAND: `purity(v)`
DESCRIPTION: Display the purity of the sub-system of qubits specified by the vector v .

COMMAND: `entropy(v)`
DESCRIPTION: Display the von Neuman entropy of the sub-system of qubits specified by the vector v .

COMMAND: `print_hist`
DESCRIPTION: Displays an ASCII based circuit model diagram for the circuit applied since `init_state` was last called.