Computer Physics Communications 182 (2011) 1693–1707

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

QCWAVE – A Mathematica quantum computer simulation update ☆

Frank Tabakin [a], Bruno Juliá-Díaz [b,c,*]

[a] Department of Physics and Astronomy, University of Pittsburgh, Pittsburgh, PA 15260, United States
[b] Departament de Estructura i Constituents de la Materia, Universitat de Barcelona, 08028 Barcelona, Spain
[c] ICFO-Institut de Ciències Fotòniques, Parc Mediterrani de la Tecnologia, 08860 Barcelona, Spain

ARTICLE INFO

Article history:
Received 11 January 2011
Received in revised form 27 February 2011
Accepted 12 April 2011
Available online 15 April 2011

Keywords:
Quantum computation
Mathematica
Quantum simulation
Quantum algorithms

ABSTRACT

This Mathematica 7.0/8.0 package upgrades and extends the quantum computer simulation code called QDENSITY. Use of the density matrix was emphasized in QDENSITY, although that code was also applicable to a quantum state description. In the present version, the quantum state version is stressed and made amenable to future extensions to parallel computer simulations. The add-on QCWAVE extends QDENSITY in several ways. The first way is to describe the action of one, two and three-qubit quantum gates as a set of small (2×2, 4×4 or 8×8) matrices acting on the 2^{n_q} amplitudes for a system of n_q qubits. This procedure was described in our parallel computer simulation QCMPI and is reviewed here. The advantage is that smaller storage demands are made, without loss of speed, and that the procedure can take advantage of message passing interface (MPI) techniques, which will hopefully be generally available in future Mathematica versions.

Another extension of QDENSITY provided here is a multiverse approach, as described in our QCMPI paper. This multiverse approach involves using the present slave–master parallel processing capabilities of Mathematica 7.0/8.0 to simulate errors and error correction. The basic idea is that parallel versions of QCWAVE run simultaneously with random errors introduced on some of the processors, with an ensemble average used to represent the real world situation. Within this approach, error correction steps can be simulated and their efficacy tested. This capability allows one to examine the detrimental effects of errors and the benefits of error correction on particular quantum algorithms.

Other upgrades provided in this version include circuit-diagram drawing commands, better Dirac form and amplitude display features. These are included in the add-ons **QCWave.m** and **Circuits.m**, and are illustrated in tutorial notebooks.

In separate notebooks, QCWAVE is applied to sample algorithms in which the parallel multiverse setup is illustrated and error correction is simulated. These extensions and upgrades will hopefully help in both instruction and in application to QC dynamics and error correction studies.

Program summary

Program title: QCWAVE
Catalogue identifier: ADXH_v3_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/ADXH_v3_0.html
Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland
Licensing provisions: Standard CPC licence, http://cpc.cs.qub.ac.uk/licence/licence.html
No. of lines in distributed program, including test data, etc.: 94 159
No. of bytes in distributed program, including test data, etc.: 734 158
Distribution format: tar.gz
Programming language: Mathematica 7.0 and 8.0.
Computer: Any supporting Mathematica.
Operating system: Any operating system that supports Mathematica; tested under Microsoft Windows XP, Macintosh OSX, and Linux FC4.
Has the code been vectorised or parallelized?: Utilises Mathematica's (7.0 and 8.0) parallel computing features.
Classification: 4.15.

☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail address: bjulia@gmail.com (B. Juliá-Díaz).

0010-4655/$ – see front matter
doi:10.1016/j.cpc.2011.04.010

Catalogue identifier of previous version: ADXH_v2_0
Journal reference of previous version: Comput. Phys. Comm. 180 (2009) 474.
Does the new version supersede the previous version?: Yes. This version supersedes all prior versions of QDENSITY.
Nature of problem: Simulation of quantum circuits, quantum algorithms, noise and quantum error correction.
Solution method: A Mathematica package containing commands to create and analyze quantum circuits is upgraded and extended, with emphasis on state amplitudes. Several Mathematica notebooks containing relevant examples are explained in detail. The parallel computing feature of Mathematica is used to develop a multiverse approach for including noise and forming suitable ensemble averaged density matrix evolution. Error correction is simulated.
Reasons for new version: The new version updates QDENSITY to run on Mathematica 7.0 and 8.0 and makes it compatible with our extension QCWAVE. QCWAVE emphasizes wavefunctions with efficient gate operations and also extends the code to use the parallel computing features of Mathematica 7.0–8.0. Circuit diagram and amplitude display are new features. Dirac display of states is also provided.
Summary of revisions: The revisions include working with state vectors and the implementation of efficient Op1, Op2 and Op3, one, two, three gate operators. Parallel processing is used to form a multiverse approach for simulating noise effects and error corrections in quantum operations. Drawing circuit diagrams and displaying amplitude evolution has been added. A simple Dirac display feature DForm has also been provided.
Running time: The notebooks provided in the distribution package take only a matter of minutes to execute.

1. Introduction

In this paper, QDENSITY [1] (a Mathematica [2] package that provides a flexible simulation of a quantum computer) is extended and upgraded by an add-on called QCWAVE.[1] The earlier flexibility in QDENSITY is enhanced by adopting a simple state vector approach to initializations, operators, gates, and measurements. Although the present version stresses a state vector approach the density matrix can always be constructed and examined. Indeed, a parallel universe (or multiverse) approach is also included, using the present Mathematica 7.0/8.0 slave–master feature. This multiverse approach, which was published [5] in our QCMPI paper,[2] allows separate dynamical evolutions on several processors with some evolutions subject to random errors. Then an ensemble average is performed over the various processors to produce a density matrix that describes a QC system with realistic errors. Error correction methods can also be invoked on the set of processors to test the efficacy of such methods on selected QC algorithms.

In Section 2, we introduce qubit state vectors and associated amplitudes for one-, two- and multi-qubit states. In Section 3, a method for handling one-, two- and three-qubit operators acting on state vectors with commands from QCWAVE are presented.

In Section 4, illustrations of how to apply gates to states are shown. In Section 5, the multiverse approach is described and the parallel method for introduction of errors and error correction is given. The ensemble averaged density matrix is then constructed.

Additional upgrades, such as Dirac and amplitude displays and circuit drawing are presented in Section 6. Suggested applications are presented in the conclusion Section 7.

1.1. Comparison to other simulators

This paper describes an environment for quantum simulation using the symbolic package Mathematica. In this subsection, we briefly describe the capabilities of our package in contrast to others on the market. A deeper comparison to other quantum simulators is beyond the scope of this work, and, if needed, is left to the potential user.

An appreciable number of quantum simulators are already available as can be seen in Refs. [5–7]. In addition to invoking various computer languages (most of them are use C/C++), the goals of these simulators often vary. For example, some focus on simulating particular algorithms, notably Shor's and Grover's; whereas, others emphasize the scalability of their codes by, in some cases, implementing parallel computing features, which no doubt will be employed extensively in the near future. An example of a quite versatile package coded in C++ is libquantum [8], which provides a modular approach to simulate a broad range of quantum problems. That example, and other packages coded in high-performing languages, are advantageous mostly for large scale simulators in presently available computers.

There is, however, a rather limited number of simulators employing high level symbolic packages, such as Mathematica, Maxima or Maple. These languages, and especially Mathematica, provide a powerful framework where analytical calculations, drawing capabilities, and good numerical tools are integrated. In this way, quantum simulators using these programming languages can be useful for a broad range of problems. For instance, our earlier version QDENSITY contained pedagogical simulations of important quantum algorithms, built from a number of macros and functions that performed the basic quantum computing algorithms. In this way, one was able to simulate a large number of user-defined quantum algorithms. The emphasis in QDENSITY was on using the density matrix as the primary tool to describe quantum systems. A particularly interesting package using Mathematica, which appeared soon after our QDENSITY, was the package "Quantum" [3], which was aimed initially at improving the notational aspects, providing powerful Dirac notation features, and also circuit drawing features. Other groups have invested some time in building similar tools in non-proprietary computer languages, as is the case of Qinf [4], which is coded using Maxima.

In QCWAVE, we have substantially improved QDENSITY by incorporating computing tools that were not available in 2005. The prime point is that we invoke a multiverse approach, using the parallel slave–master capabilities, that are now available in current Mathematica distributions. That allows simulation of noise and of error correction, and as that capability develops will allow for larger scale computation. Although current highly-performing languages are faster in numerical computations than symbolic ori-

[1] Other authors have also developed Mathematica/Maxima QDENSITY based quantum computing simulations [3,4]. Hopefully, they will incorporate the ideas we provide herein in their future efforts.

[2] QCMPI is a quantum computer (QC) simulation package written in Fortran 90 with parallel processing capabilities.

ented languages, there is an ongoing, important effort to improve the numerical capabilities of symbolic oriented languages. Clearly, integrating parallel computing tools is an important step forward in that effort.

2. Multi-qubit states

2.1. One-qubit states

The basic idea of a quantum state, its representation in Hilbert space and the concepts of quantum computing have been discussed in many texts [9–11]. A brief review was given in our earlier papers in this series [1,5]. Here we proceed directly from one-, two- and multi-qubit states and their amplitudes to how various operators alter those amplitudes.

To start, recall that when one focuses on just two states of a quantum system, such as the spin part of a spin-1/2 particle, the two states are represented as either $|0\rangle$ or $|1\rangle$. A one qubit state is a superposition of the two states associated with the above 0 and 1 bits:

$$|\Psi_1\rangle = C_0|0\rangle + C_1|1\rangle, \tag{1}$$

where $C_0 \equiv \langle 0|\Psi_1\rangle$ and $C_1 \equiv \langle 1|\Psi_1\rangle$ are complex probability amplitudes for finding the qubit in the state $|0\rangle$ or $|1\rangle$, respectively. The normalization of the state $\langle\Psi_1|\Psi_1\rangle = 1$, yields $|C_0|^2 + |C_1|^2 = 1$. Note that the spatial aspects of the wave function are being suppressed; which corresponds to the particle being in a fixed location. The kets $|0\rangle$ and $|1\rangle$ can be represented as $|0\rangle \to \binom{1}{0}$ and $|1\rangle \to \binom{0}{1}$. Hence a 2×1 matrix representation of this one-qubit state is: $|\Psi_1\rangle \to \binom{C_0}{C_1}$.

An essential point is that a quantum mechanical (QM) system can exist in a superposition of these two bits; hence, the state is called a quantum-bit or "qubit". Although our discussion uses the notation of a system with spin 1/2, it should be noted that the same discussion applies to any two distinct quantum states that can be associated with $|0\rangle$ and $|1\rangle$.

2.2. Two-qubit states

The single-qubit case can now be generalized to multiple qubits. Consider the product space of two qubits both in the "up" $|0\rangle$ state and denote that product state as $|0\ 0\rangle = |0\rangle|0\rangle$, which clearly generalizes to

$$|q_1\ q_2\rangle = |q_1\rangle|q_2\rangle, \tag{2}$$

where q_1, q_2 in general take on the values 0 and 1. This product is called a tensor product and is symbolized as

$$|q_1\ q_2\rangle = |q_1\rangle \otimes |q_2\rangle. \tag{3}$$

In QDENSITY, the kets $|0\rangle, |1\rangle$ are invoked by the commands **Ket**[0] and **Ket**[1], and the product state by for example $|00\rangle =$ **Ket**[0] $\otimes$ **Ket**[0].

The kets $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$ can be represented as 4×1 matrices

$$|00\rangle \to \begin{pmatrix}1\\0\\0\\0\end{pmatrix}; \quad |01\rangle \to \begin{pmatrix}0\\1\\0\\0\end{pmatrix}; \quad |10\rangle \to \begin{pmatrix}0\\0\\1\\0\end{pmatrix}; \quad |11\rangle \to \begin{pmatrix}0\\0\\0\\1\end{pmatrix}. \tag{4}$$

Hence, a 4×1 matrix representation of the two-qubit state

$$|\Psi_2\rangle = C_0|00\rangle + C_1|01\rangle + C_2|10\rangle + C_3|11\rangle, \tag{5}$$

is

$$|\Psi_2\rangle \to \begin{pmatrix}C_0\\C_1\\C_2\\C_3\end{pmatrix}. \tag{6}$$

Again $C_0 \equiv \langle 00|\Psi_2\rangle$, $C_1 \equiv \langle 01|\Psi_2\rangle$, $C_2 \equiv \langle 10|\Psi_2\rangle$, and $C_3 \equiv \langle 11|\Psi_2\rangle$, are complex probability amplitudes for finding the two-qubit system in the states $|q_1\ q_2\rangle$. The normalization of the state $\langle\Psi_2|\Psi_2\rangle = 1$, yields

$$|C_0|^2 + |C_1|^2 + |C_2|^2 + |C_3|^2 = 1. \tag{7}$$

Note that we label the amplitudes using the decimal equivalent of the bit product q_1q_2, so that for example a binary label on the amplitude C_{10} is equivalent to the decimal label C_2.

2.3. Multi-qubit states

For n_q qubits the computational basis of states generalizes to:

$$|n\rangle_{n_q} \equiv |q_1\rangle \cdots |q_{n_q}\rangle \equiv |q_1\ q_2\ \cdots\ q_{n_q}\rangle \equiv |\mathbf{Q}\rangle. \tag{8}$$

We use the convention that the most significant qubit is labeled as q_1 and the least significant qubit by q_{n_q}. Note we use q_i to indicate the quantum number of the ith qubit. The values assumed by any qubit is limited to either $q_i = 0$ or 1. The state label $\mathbf{Q}$ denotes the qubit array $\mathbf{Q} = (q_1, q_2, \ldots, q_{n_q})$, which is a binary number label for the state with equivalent decimal label n. This decimal multi-qubit state label is related to the equivalent binary label by

$$n \equiv q_1 \cdot 2^{n_q-1} + q_2 \cdot 2^{n_q-2} + \cdots + q_{n_q} \cdot 2^0 = \sum_{i=1}^{n_q} q_i \cdot 2^{n_q-i}. \tag{9}$$

Note that the ith qubit contributes a value of $q_i \cdot 2^{n_q-i}$ to the decimal number n. Later we will consider "partner states" $(|n_0\rangle, |n_1\rangle)$ associated with a given n, where a particular qubit i_s has a value of $q_{i_s} = 0$,

$$n_0 = n - q_{i_s} \cdot 2^{n_q-i_s}, \tag{10}$$

or a value of $q_{i_s} = 1$,

$$n_1 = n - (q_{i_s} - 1) \cdot 2^{n_q-i_s}. \tag{11}$$

These partner states are involved in the action of a single operator acting on qubit i_s, as described in the next section.

A general state with n_q qubits can be expanded in terms of the above computational basis states as follows

$$|\Psi\rangle_{n_q} = \sum_{\mathbf{Q}} C_{\mathbf{Q}}|\mathbf{Q}\rangle \equiv \sum_{n=0}^{2^{n_q}-1} C_n|n\rangle, \tag{12}$$

where the sum over $\mathbf{Q}$ is really a product of n_q summations of the form $\sum_{q_i=0,1}$. The above Hilbert space expression maps over to an array, or column vector, of length 2^{n_q}

$$|\Psi\rangle_{n_q} \equiv \begin{pmatrix}C_0\\C_1\\\vdots\\\vdots\\C_{2^{n_q}-1}\end{pmatrix} \quad \text{or with binary labels} \longrightarrow \begin{pmatrix}C_{0\cdots00}\\C_{0\cdots01}\\\vdots\\\vdots\\C_{1\cdots11}\end{pmatrix}. \tag{13}$$

The expansion coefficients C_n (or $C_{\mathbf{Q}}$) are complex numbers with the physical meaning that $C_n = \langle n|\Psi\rangle_{n_q}$ is the probability amplitude for finding the system in the computational basis state $|n\rangle$,

which corresponds to having the qubits pointing in the directions specified by the binary array **Q**. Switching between decimal n and equivalent binary **Q** labels is accomplished by the Mathematica command **IntegerDigits**.

In general, the complex amplitudes C_n vary with time and are changed by the action of operators or gates, as outlined next.

3. Multi-qubit operators

Operators that act in the multi-qubit space described above can be generated from a set of separate Pauli operators,[3] acting in each qubit space. These separate Pauli operators refer to distinct quantum systems and hence they commute. Note, Pauli operators acting on the same qubit do not commute; indeed, they have the property $\sigma_i\sigma_j - \sigma_j\sigma_i = 2i\epsilon_{ijk}\sigma_k$. The Pauli operator σ_0 is just the unit 2×2 matrix. We denote a Pauli operator acting on qubit i_s as $\sigma_k^{(i_s)}$, where $k = (x, y, z) = (1, 2, 3)$ is the component of the Pauli operator. For example, the tensor product of two-qubit operators has the following structure

$$\begin{aligned}\langle a_1|\sigma_i|b_1\rangle\langle a_2|\sigma_j|b_2\rangle &= \langle a_1a_2|\sigma_i^{(1)}\sigma_j^{(2)}|b_1b_2\rangle \\ &= \langle a_1a_2|\sigma_i^{(1)}\otimes\sigma_j^{(2)}|b_1b_2\rangle,\end{aligned} \tag{14}$$

which defines what we mean by the tensor product of two qubit operators $\sigma_i^{(1)}\otimes\sigma_j^{(2)}$. The generalization to more qubits is immediate

$$\left(\sigma_i^{(1)}\otimes\sigma_j^{(2)}\right)\otimes\left(\sigma_k^{(3)}\otimes\sigma_l^{(4)}\right)\cdots. \tag{15}$$

3.1. One-qubit operators

One-qubit operators change the amplitude coefficients of the quantum state. The NOT and Hadamard $\mathcal{H}$ are examples of one-qubit operators of particular interest: $\mathbf{NOT}\equiv\sigma_x = \left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)$, $\mathcal{H}\equiv\frac{\sigma_x+\sigma_z}{\sqrt{2}} = \frac{1}{\sqrt{2}}\left(\begin{smallmatrix}1&1\\1&-1\end{smallmatrix}\right)$. These have the following effect on the basis states $\mathbf{NOT}|0\rangle = |1\rangle$, $\mathbf{NOT}|1\rangle = |0\rangle$, and $\mathcal{H}|0\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$, and $\mathcal{H}|1\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$.

General one-qubit operators can also be constructed from the Pauli operators; we denote the general one-qubit operator acting on qubit s as Ω_s. Consider the action of such an operator on the multi-qubit state $|\Psi\rangle_{n_q}$:

$$\Omega_s|\Psi\rangle_{n_q} = \sum_{\mathbf{Q}} C_{\mathbf{Q}}\Omega_s|\mathbf{Q}\rangle \tag{16}$$

$$= \sum_{q_1=0,1}\cdots\sum_{q_s=0,1}\cdots\sum_{q_{nq}=0,1} C_{\mathbf{Q}}|q_1\rangle\cdots\left(\Omega_s|q_s\rangle\right)\cdots|q_{n_q}\rangle. \tag{17}$$

Here Ω_s is assumed to act only on the qubit i_s of value q_s. The $(\Omega_s|q_s\rangle)$ term can be expressed as

$$\Omega_s|q_s\rangle = \sum_{q_s'=0,1}|q_s'\rangle\langle q_s'|\Omega_s|q_s\rangle, \tag{18}$$

using the closure property of the one-qubit states. Thus Eq. (17) becomes

$$\begin{aligned}\Omega_s|\Psi\rangle_{n_q} &= \sum_{\mathbf{Q}} C_{\mathbf{Q}}\Omega_s|\mathbf{Q}\rangle \\ &= \sum_{q_1=0,1}\cdots\sum_{q_s=0,1}\cdots\sum_{q_{nq}=0,1} \\ &\quad\times\sum_{q_s'=0,1} C_{\mathbf{Q}}\langle q_s'|\Omega_s|q_s\rangle|q_1\rangle\cdots|q_s'\rangle\cdots|q_{n_q}\rangle.\end{aligned} \tag{19}$$

Now we can interchange the labels $q_s\leftrightarrow q_s'$, and use the label **Q** to obtain the algebraic result for the action of a one-qubit operator on a multi-qubit state

$$\Omega_s|\Psi\rangle_{n_q} = \sum_{\mathbf{Q}}\tilde{C}_{\mathbf{Q}}|\mathbf{Q}\rangle = \sum_{n=0}^{2^{n_q}-1}\tilde{C}_n\,|n\rangle, \tag{20}$$

where

$$\tilde{C}_{\mathbf{Q}} = \tilde{C}_n = \sum_{q_s'=0,1}\langle q_s|\Omega_s|q_s'\rangle C_{\mathbf{Q}'}, \tag{21}$$

where $\mathbf{Q} = (q_1, q_2, \ldots, q_{n_q})$, and $\mathbf{Q}' = (q_1, \ldots, q_s', \ldots, q_{n_q})$. That is **Q** and $\mathbf{Q}'$ are equal except for the qubit acted upon by the one-body operator Ω_s.

A better way to state the above result is to consider Eq. (21) for the case that n has $q_s = 0$ and thus $n\to n_0$ and to write out the sum over q_s' to get

$$\tilde{C}_{n_0} = \langle 0|\Omega_s|0\rangle C_{n_0} + \langle 0|\Omega_s|1\rangle C_{n_1}, \tag{22}$$

where we introduced the partner to n_0 namely n_1. For the case that n has $q_s = 1$ and thus $n\to n_1$ Eq. (21), with expansion of the sum over q_s' yields

$$\tilde{C}_{n_1} = \langle 1|\Omega_s|0\rangle C_{n_0} + \langle 1|\Omega_s|1\rangle C_{n_1} \tag{23}$$

or written as a matrix equation we have for each n_0, n_1 partner pair

$$\begin{pmatrix}\tilde{C}_{n_0}\\ \tilde{C}_{n_1}\end{pmatrix} = \begin{pmatrix}\langle 0|\Omega_s|0\rangle & \langle 0|\Omega_s|1\rangle\\ \langle 1|\Omega_s|0\rangle & \langle 1|\Omega_s|1\rangle\end{pmatrix}\begin{pmatrix}C_{n_0}\\ C_{n_1}\end{pmatrix}. \tag{24}$$

This is not an unexpected result.

Eq. (24) above shows how a 2×2 one-qubit operator Ω_s acting on qubit i_s changes the state amplitude for each value of n_0. Here, n_0 denotes a decimal number for a computational basis state with qubit i_s having the q_s value zero and n_1 denotes its partner decimal number for a computational basis state with qubit i_s having the q_s value one. They are related by

$$n_1 = n_0 + 2^{n_q - i_s}. \tag{25}$$

At times, we shall call $2^{n_q-i_s}$ the "stride" of the i_s qubit; it is the step in n needed to get to a partner. There are $2^{n_q}/2$ values of n_0 and hence $2^{n_q}/2$ pairs n_0, n_1. Eq. (24) is applied to each of these pairs. In QCWAVE that process is included in the command **Op1**.[4]

Note that we have replaced the full $2^{n_q}\times 2^{n_q}$ one-qubit operator by a series of $2^{n_q}/2$ sparse matrices. Thus we do not have to store the full $2^{n_q}\times 2^{n_q}$ but simply provide a 2×2 matrix for repeated use. Each application of the 2×2 matrix involves distinct amplitude partners and therefore the set of 2×2 operations can occur simultaneously and hence in parallel. That parallel advantage was employed in our QCMPI Fortran version, using the MPI [12] protocol for inter-processor communication. The 7.0 and 8.0 versions of Mathematica include only master–slave communication and therefore this advantage is not generally available. It is possible to use

[3] The Pauli operators act in the qubit Hilbert space, and have the matrix representation: $\sigma_x = \left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)$; $\sigma_y = \left(\begin{smallmatrix}0&-I\\I&0\end{smallmatrix}\right)$; $\sigma_z = \left(\begin{smallmatrix}1&0\\0&-1\end{smallmatrix}\right)$. Here $I\equiv\sqrt{-1}$.

[4] **Op1** yields result of a one-body operator Ω acting on qubit "is'" in state ψ_0; the result is the final state ψ_f. Called as: $\psi_f = \mathbf{Op1}[\Omega, is, \psi_0]$.

MPI with Mathematica [13], but only at considerable cost. Another promising idea is to use the "CLOJURATICA" [14] package, but that entails an additional language. So the full MPI advantage will have to wait until MPI becomes available hopefully on future Mathematica versions.

In the next section, this procedure is generalized to two- and three-qubit operators, using the same concepts.

3.2. Two-qubit operators

The case of a two-qubit operator is a generalization of the steps discussed for a one-qubit operator. Nevertheless, it is worthwhile to present those details, as a guide to those who plan to use and perhaps extend QCWAVE.

We now consider a general two-qubit operator that we assume acts on qubits i_{s_1} and i_{s_2}, each of which ranges over the full $1,\ldots,n_q$ possible qubits. General two-qubit operators can be constructed from tensor products of two Pauli operators; we denote the general two-qubit operator as $\mathcal{V}$. Consider the action of such an operator on the multi-qubit state $|\Psi\rangle_{n_q}$:

$$\begin{aligned}\mathcal{V}|\Psi\rangle_{n_q} &= \sum_{\mathbf{Q}} C_{\mathbf{Q}}\mathcal{V}|\mathbf{Q}\rangle \\ &= \sum_{q_1=0}^{1}\cdots\sum_{q_{s1},q_{s2}=0}^{1}\cdots\sum_{q_{nq}=0}^{1} C_{\mathbf{Q}}|q_1\rangle\cdots\big(\mathcal{V}|q_{s1}q_{s2}\rangle\big)\cdots|q_{n_q}\rangle.\end{aligned} \tag{26}$$

Here $\mathcal{V}$ is assumed to act only on the two q_{s1}, q_{s2} qubits. The $(\mathcal{V}|q_{s1}q_{s2}\rangle)$ term can be expressed as

$$\mathcal{V}|q_{s1}q_{s2}\rangle = \sum_{q'_{s1},q'_{s2}=0}^{1} |q'_{s1}q'_{s2}\rangle\langle q'_{s1}q'_{s2}|\mathcal{V}|q_{s1}q_{s2}\rangle \tag{27}$$

using the closure property of the two-qubit product states. Thus Eq. (26) becomes

$$\begin{aligned}\mathcal{V}|\Psi\rangle_{n_q} &= \sum_{\mathbf{Q}} C_{\mathbf{Q}}\mathcal{V}|\mathbf{Q}\rangle \\ &= \sum_{q_1=0}^{1}\cdots\sum_{q_{s1}=0}^{1}\cdots\sum_{q_{s2}=0}^{1}\cdots\sum_{q_{nq}=0}^{1} \\ &\quad\times\sum_{q'_{s1},q'_{s2}=0}^{1} C_{\mathbf{Q}}\langle q'_{s1}q'_{s2}|\mathcal{V}|q_{s1}q_{s1}\rangle|q_1\rangle\cdots|q'_{s1}q'_{s2}\rangle\cdots|q_{n_q}\rangle.\end{aligned} \tag{28}$$

Now we can interchange the labels $q_{s1}\leftrightarrow q'_{s1}$, $q_{s2}\leftrightarrow q'_{s2}$ and use the label $\mathbf{Q}$ to obtain the algebraic result for the action of a two-qubit operator on a multi-qubit state,

$$\mathcal{V}|\Psi\rangle_{n_q} = \sum_{\mathbf{Q}}\tilde{C}_{\mathbf{Q}}|\mathbf{Q}\rangle = \sum_{n=0}^{2^{n_q}-1}\tilde{C}_n|n\rangle, \tag{29}$$

where

$$\tilde{C}_{\mathbf{Q}} = \tilde{C}_n = \sum_{q'_{s1},q'_{s2}=0}^{1}\langle q_{s1}q_{s2}|\Omega_s|q'_{s1}q'_{s2}\rangle C'_{\mathbf{Q}}, \tag{30}$$

where $\mathbf{Q}=(q_1,q_2,\ldots,q_{n_q})$, and $\mathbf{Q}'=(q_1,\ldots,q'_{s1},\ldots,q'_{s2},\ldots,q_{n_q})$. That is $\mathbf{Q}$ and $\mathbf{Q}'$ are equal except for the qubits acted upon by the two-body operator $\mathcal{V}$.

```
In[98]:= psi

Out[98]= {1, 0, 0, 0, 0, 0, 0, 0}

In[99]:= Op1[ℋ, 1, psi]

Out[99]= {1/√2, 0, 0, 0, 1/√2, 0, 0, 0}

In[100]:= DForm[%]

Out[100]= + 1/√2 |000 > + 1/√2 |100 >
```

Fig. 1. One Hadamard example. Here psi $=|000\rangle$.

A better way to state the above result is to consider Eq. (30) for the following four choices

$$\begin{aligned} n_{00} &\to (q_1,\ldots,q_{s1}=0,\ldots,q_{s2}=0,\ldots,q_{n_q}),\\ n_{01} &\to (q_1,\ldots,q_{s1}=0,\ldots,q_{s2}=1,\ldots,q_{n_q}),\\ n_{10} &\to (q_1,\ldots,q_{s1}=1,\ldots,q_{s2}=0,\ldots,q_{n_q}),\\ n_{11} &\to (q_1,\ldots,q_{s1}=1,\ldots,q_{s2}=1,\ldots,q_{n_q}),\end{aligned} \tag{31}$$

where the computational basis state label $n_{q_{s1},q_{s2}}$ denotes the four decimal numbers corresponding to $\mathbf{Q}=(q_1,\ldots,q_{s1},\ldots,q_{s2},\ldots,q_{n_q})$.

Evaluating Eq. (30) for the four choices Eq. (31) and completing the sums over q'_{s1}, q'_{s2}, the effect of a general two-qubit operator on a multi-qubit state amplitudes is given by a 4×4 matrix

$$\begin{pmatrix}\tilde{C}_{n_{00}}\\ \tilde{C}_{n_{01}}\\ \tilde{C}_{n_{10}}\\ \tilde{C}_{n_{11}}\end{pmatrix} = \begin{pmatrix}\mathcal{V}_{00;00} & \mathcal{V}_{00;01} & \mathcal{V}_{00;10} & \mathcal{V}_{00;11}\\ \mathcal{V}_{01;00} & \mathcal{V}_{01;01} & \mathcal{V}_{01;10} & \mathcal{V}_{01;11}\\ \mathcal{V}_{10;00} & \mathcal{V}_{10;01} & \mathcal{V}_{10;10} & \mathcal{V}_{10;11}\\ \mathcal{V}_{11;00} & \mathcal{V}_{11;01} & \mathcal{V}_{11;10} & \mathcal{V}_{11;11}\end{pmatrix}\begin{pmatrix}C_{n_{00}}\\ C_{n_{01}}\\ C_{n_{10}}\\ C_{n_{11}}\end{pmatrix}, \tag{32}$$

where $\mathcal{V}_{ij;kl}\equiv\langle i,j|\mathcal{V}|k,l\rangle$. Eq. (32) shows how a 4×4 two-qubit operator $\mathcal{V}$ acting on qubits i_{s1}, i_{s2} changes the state amplitude for each value of n_{00}. Here, n_{00} denotes a decimal number for a computational basis state with qubits i_{s1}, i_{s2} both having the values zero and its three partner decimal numbers for a computational basis state with qubits i_{s1}, i_{s2} having the values $(0,1)$, $(1,0)$ and $(1,1)$, respectively. The four partners n_{00}, n_{01}, n_{10}, n_{11}, or "ampli-

```
In[107]:= Op1[ℋ, 3, Op1[ℋ, 2, Op1[ℋ, 1, psi]]]

Out[107]= {1/(2√2), 1/(2√2), 1/(2√2), 1/(2√2), 1/(2√2), 1/(2√2), 1/(2√2), 1/(2√2)}

In[108]:= DForm[%]

Out[108]= + 1/(2√2) |000 > + 1/(2√2) |001 > + 1/(2√2) |010 > + 1/(2√2) |
           011 > + 1/(2√2) |100 > + 1/(2√2) |101 > + 1/(2√2) |110 > + 1/(2√2) |111 >
```

Fig. 2. Hadamards on all three-qubits example. Here psi = |000⟩.

```
In[111]:= ΩALL[ℋ, psi]

Out[111]= {1/(2√2), 1/(2√2), 1/(2√2), 1/(2√2), 1/(2√2), 1/(2√2), 1/(2√2), 1/(2√2)}

In[112]:= DForm[%]

Out[112]= + 1/(2√2) |000 > + 1/(2√2) |001 > + 1/(2√2) |010 > + 1/(2√2) |
           011 > + 1/(2√2) |100 > + 1/(2√2) |101 > + 1/(2√2) |110 > + 1/(2√2) |111 >
```

Fig. 3. Hadamards on all three-qubits using the ΩALL command. Here psi = |000⟩.

tude quartet", coupled by the two-qubit operator are related by:

$$n_{01} = n_{00} + 2^{n_q - i_{s2}}, \qquad n_{10} = n_{00} + 2^{n_q - i_{s1}},$$
$$n_{11} = n_{00} + 2^{n_q - i_{s1}} + 2^{n_q - i_{s2}}, \tag{33}$$

where i_{s2}, i_{s2} label the quarks that are acted on by the two-qubit operator.

There are $2^{n_q}/4$ values of n_{00} and hence $2^{n_q}/4$ amplitude quartets n_{00}, n_{01}, n_{10}, n_{11}. Eq. (32) is applied to each of these quartets

```
In[113]:= psi

Out[113]= {1, 0, 0, 0, 0, 0, 0, 0}

In[114]:= X1 =  ℋ (1) | psi > ;
          X2 = ℋ (2) | X1  >;
          X3 = ℋ (3) | X2  >;

In[117]:= DForm[X1  ]
          DForm[X2  ]
          DForm[X3  ]

Out[117]=  + 1/√2 |000 > + 1/√2 |100 >

Out[118]=  + 1/2 |000 > + 1/2 |010 > + 1/2 |100 > + 1/2 |110 >

Out[119]=  + 1/(2 √2) |000 > + 1/(2 √2) |001 > + 1/(2 √2) |010 > + 1/(2 √2) |
           011 > + 1/(2 √2) |100 > + 1/(2 √2) |101 > + 1/(2 √2) |110 > + 1/(2 √2) |111 >

In[120]:= Op1[ℋ, 3, Op1[ℋ, 2, Op1[ℋ, 1, psi]]] == X3

Out[120]= True
```

Fig. 4. Application of Hadamards in Dirac form. Here psi = |000⟩.

```
(*  One-body operator Ω acts on qubit "is"  in state "state";
  Op1[ is,Ω,state] ;Using QCMPI method*)

   Op1[ Ω_?MatrixQ, is_,state_?VectorQ] :=
 Module[{wf,temp,i1,j1,icas,nqu},nqx=Log[2,Length[state]];nqu=nqx;
wf=state;temp=wf;Do[i1=Pick1[is,icas,1];j1=Pick1[is,icas,2];
temp[[i1]]= Ω[[1]][[1]]*wf[[i1]]+Ω[[1]][[2]]*wf[[j1]];
temp[[j1]]= Ω[[2]][[1]]*wf[[i1]]+Ω[[2]][[2]]*wf[[j1]],
{icas,1,2^nqu/2}];temp];
```

Fig. 5. The Op1 command as stipulated in **QCWave.m**.

for a given pair of struck qubits. In QCWAVE that process is included in the command **Op2**.[5]

In this treatment, we are essentially replacing a large sparse matrix, by a set of $2^{n_q}/4$ 4×4 matrix actions, thereby saving the storage of that large matrix.

3.3. Three-qubit operators

The above procedure can be extended to the case of three-qubit operators. Instead of pairs or quartets of states that are modified, we now have an octet of states modified by the three-qubit operator and $2^{n_q}/8$ repeats to cover the full change induced in the amplitude coefficients. For brevity we omit the derivation. In

[5] **Op2** yields result of a two-body operator Ω acting on qubits "is" and "$is2$" in state ψ_0; the result is the final state ψ_f. Called as: $\psi_f = \mathbf{Op2}[\Omega, is1, is2, \psi_0]$.

QCWAVE that process has been implemented by the command **Op3**.[6]

4. Implementation of gates on states

We now present some sample cases in which the above gates are applied to state vectors.

4.1. One-qubit operators

Consider a state vector for $n_q = 3$ qubits defined by $|\Psi\rangle_3 = |000\rangle$, which in vector form is

$$|\Psi_3\rangle = \begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{34}$$

Now have a Hadamard act on qubit 1 by use of the command $Op1[\mathcal{H}, 1, \Psi_3]$. The result is displayed in vector form and then in Dirac form by use of the DForm command in Fig. 1.

One can act with Hadamards on every qubit, by either repeated use of Op1 Fig. 2, or by the command ΩALL[$\mathcal{H}$, psi], which is illustrated in Fig. 3. A Dirac type notation is also available as illustrated in Fig. 4.

In **QCWave.m** the command Op1 is given as a Module see Fig. 5, which makes use of the command Pick1, Pick1 selects the pairs of decimal labels which differ only in the "is" qubits value of 1 and 0. Then all such pairs are swept through. Examples of Pick1 are presented in the Tutorial.

4.2. Two-qubit operators

The typical two-qubit operators are the CNOT, and controlled phase operators. General two-qubit operators can be constructed from tensor products of two Pauli operators, as discussed earlier. An example from QCWave of application of a CNOT gate is presented in Fig. 6.

A Dirac type notation is also available as illustrated in Fig. 7.

In **QCWave.m** the command Op2 is given as a Module see Fig. 8, which makes use of the command Pick2, Pick2 selects the quartet of decimal labels which differ only in the "is1" and "is2" qubit's values of 1 and 0. Then all such quartets are swept through. Examples of Pick2 are presented in the Tutorial.

4.3. Three-qubit operators

The Op3 command is also provided in **QCWave.m** as a Module, which makes use of the command Pick3, Pick3 selects the octet of decimal labels which differ only in the "is1," "is2," and "is3" qubit's values of 1 and 0. Then all such octets are swept through. Application to the Tofolli gate is provided in the Tutorial.

5. The multiverse approach

5.1. General remarks

Mathematica 7.0 & 8.0 provide a master–slave parallel processing facility. This is not a full implementation of a parallel processing setup that allows communication between the "slave" processors, such as used by the MPI [12] protocol. If MPI were readily available in Mathematica then one could invoke the full capabilities discussed in QCMPI. That capability allows for the state vector to be distributed over several processors which increases the number of qubits that could be simulated. It is indeed possible to have Mathematica upgraded to include MPI slave to slave communication; as is available in the "POOCH" [13] package. However, since that is an expensive route and most Mathematica users do not have access to MPI, although one can hope for such a capability in the future, we have not invoked the full state distribution aspect.

Nevertheless, the master–slave Mathematica 7.0 capability does provide for concurrent versions of a QCWave based algorithm to be run with different random error scenarios. Then an ensemble averaged density matrix can be formed which describes a real error prone QC setup. That opens the possibility of examining the role of errors and the efficacy of error correction methods using Mathematica.

Therefore, we provide a sample of a parallel setup using some simple basic algorithms, where the parallel setup is described and

```
In[132]:= Clear[chi]
          nqx = 3;
          chi = Flatten[ Ket[1] ⊗ (Ket[0] ⊗ Ket[0] )];
          DForm[ chi]

Out[135]=   + 1 |100 >

In[136]:= CN = CNOT[2, 1, 2]

Out[136]= {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 0, 1}, {0, 0, 1, 0}}

          Apply CNOT to qubits 1 & 2 of state chi.
          View in Dirac Forms

In[137]:= Op2[CN, 1, 2, chi] ;
          DForm[%]

Out[138]=   + 1 |110 >

          Apply CNOT to qubits 1 & 3 of state chi.

In[139]:= Op2[CN, 1, 3, chi] ;
          DForm[%]

Out[140]=   + 1 |101 >
```

Fig. 6. Use of Op2 to apply CNOT gates.

```
          Apply CNOT to qubits 1 & 2, then 1& 3 of state chi.

In[153]:= Y1 = CN (1, 2) | chi >;
          Y2 = CN (1, 3) | Y1 >;

In[155]:= DForm[chi]
          DForm[Y1]
          DForm[Y2]

Out[155]=   + 1 |100 >

Out[156]=   + 1 |110 >

Out[157]=   + 1 |111 >
```

Fig. 7. Application of CNOT gates in Dirac form.

[6] **Op3** yields result of a three-body operator Ω acting on qubits "$is1$", "$is2$" and "$is3$" in state ψ_0; the result is the final state ψ_f. Called as: $\psi_f =$ **Op3**$[\Omega, is1, is2, is3, \psi_0]$.

```
(*  Two-body operator Ω acts on qubits
 "is1" and "is2"  in state "state";
Op2[ is1,is2,Ω,state] ;Using QCMPI method*)
   Op2[Ω_?MatrixQ, is1_,is2_,state_?VectorQ] :=
 Module[{wf,temp,i2,j2,k2,m2,icase,nqu},wf=state;temp=wf;
nqx=Log[2,Length[state]];nqu=nqx;
Do[
i2=Pick2[is1,is2,icase,1];
j2=Pick2[is1,is2,icase,2];
k2=Pick2[is1,is2,icase,3];
m2=Pick2[is1,is2,icase,4];
temp[[i2]]= Ω[[1]][[1]]*wf[[i2]]
+Ω[[1]][[2]]*wf[[j2]]+Ω[[1]][[3]]*wf[[k2]]+Ω[[1]][[4]]*wf[[m2]];
temp[[j2]]= Ω[[2]][[1]]*wf[[i2]]
+Ω[[2]][[2]]*wf[[j2]]+Ω[[2]][[3]]*wf[[k2]]+Ω[[2]][[4]]*wf[[m2]];
temp[[k2]]= Ω[[3]][[1]]*wf[[i2]]
+Ω[[3]][[2]]*wf[[j2]]+Ω[[3]][[3]]*wf[[k2]]+Ω[[3]][[4]]*wf[[m2]];
temp[[m2]]= Ω[[4]][[1]]*wf[[i2]]
+Ω[[4]][[2]]*wf[[j2]]+Ω[[4]][[3]]*wf[[k2]]+Ω[[4]][[4]]*wf[[m2]],
{icase,1,2^nqu/4}];temp];
```

Fig. 8. The Op2 command as stipulated in **QCWave.m**.

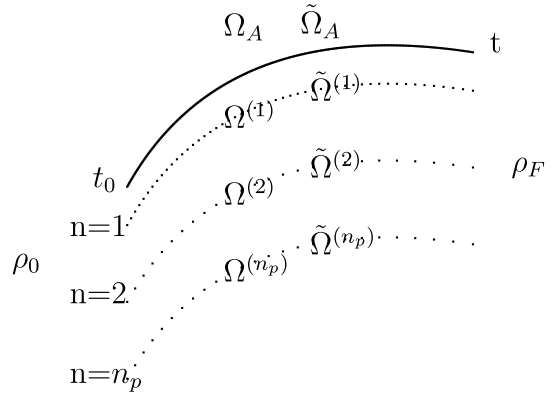

Fig. 9. The multiverse approach. Each curve represents a possible evolution of a quantum system of n_q qubits. The solid curve represents the main evolution path, with probability p. The dotted curves represent the nth evolution path, with probability $\frac{\epsilon}{n_p}$. The initial density matrix is ρ_0 and the final ρ_F. The unitary noise operators $\Omega^{(i)}$ act on each path i. Here $\tilde{\Omega}^{(i)}$ denote subsequent noise operators. The operators Ω_A and $\tilde{\Omega}_A$ denote the n_q algorithm operators which act on all paths.

explained in detail. The following steps are needed: (1) set up your Mathematica code to access several processors, see Appendix A for some help; (2) identify the processor number; (3) introduce random errors depending on the processor number; (4) assign a probability distribution for the various processors; (5) form an ensemble average over the processors and store that information as a density matrix on the master processor; (6) repeat these steps including, the algorithm, noise and finally error correction (EC) steps on all processors; (7) examine the resultant density matrix and its evolution to test the EC efficacy. This is an important program that we start by providing simple examples.

A quantum system can evolve in many ways. Different dynamical evolutions are called paths [15] or histories [16]. We refer to these alternate evolutions as separate "universes" and a collection of such possibilities as a multiverse or ensemble of paths. Parallel processing provides a convenient method for describing such alternate paths.

5.2. The ideal and the noisy channels

In our application, we assume that the main path follows an ideal algorithm exactly and the alternate paths incorporate the algorithm with possible noise. That noise is described by random one-qubit operators acting once, or with less likelihood twice. To describe this idea, which is realized in the notebooks **MV1-Noise.nb**, **MV2-Noise.nb** and **MVn-Noise.nb**, consider an initial density matrix ρ_0. For a pure state, $\rho_0 = |\psi_0\rangle\langle\psi_0|$, but it can be a general initial density matrix subject only to the conditions $\rho^\dagger = \rho$ and $\mathrm{Tr}[\rho] = 1$. The density matrix has $2^{2n_q} - 1$ parameters and 2^{n_q} real eigenvalues $\lambda_n \leqslant 1$ with $\sum_n \lambda_n = 1$. The simplest one-qubit case has the form $\rho = \frac{1}{2}(1 + \vec{P}\cdot\vec{\sigma})$, where the real polarization vector $\vec{P} = \mathrm{Tr}[\vec{\sigma}\,\rho]$, is within the "Bloch sphere", $(\vec{P}\cdot\vec{P}) \leqslant 1$.[7]

5.2.1. Storage case

Consider a simple case where the ideal algorithm is simply leaving the state, as described by ρ_0, alone. This is a memory storage case. Ideally, ρ remains fixed in time. Assume however that this ideal case occurs with a probability $\mathrm{p} \lesssim 1$ and that alternate evolutions occur with a probability $\epsilon = 1 - \mathrm{p}$. For example, we take $\mathrm{p} = 0.8$ and $\epsilon = 0.2$, corresponding to an 80% perfect storage and 20% possibility of noise. We also assume for more than one-qubit cases that 95% of the 20% noise ($0.2 \times 0.95 \to 19\%$) involves a single one-qubit hit, while 5% of the 20% noise ($0.2 \times 0.05 \to 1\%$) involves two one-qubit hits.

The ensemble average over all paths then yields a density matrix

$$\rho_f = \mathrm{p}\rho_0 + \frac{\epsilon}{n_p}\big(\Omega^{(1)}\rho_0\Omega^{(1)\dagger} + \Omega^{(2)}\rho_0\Omega^{(2)\dagger}\cdots\Omega^{(n_p)}\rho_0\Omega^{(n_p)\dagger}\big), \tag{35}$$

where the operators $\Omega^{(1)}, \Omega^{(2)}, \ldots, \Omega^{(n_p)}$ act in each of the n_p paths with a probability $\frac{\epsilon}{n_p}$. Each of these n_p terms is evaluated on a separate processor, so that n_p equals the total number of processors invoked. The above ensemble average preserves the trace:

$$\mathrm{Tr}[\rho_f] = \mathrm{p}\,\mathrm{Tr}[\rho_0] + \frac{\epsilon}{n_p}\sum_{n=1}^{n=n_p}\mathrm{Tr}\big[\Omega^{(n)}\rho_0\Omega^{(n)\dagger}\big] = \mathrm{p} + \frac{\epsilon}{n_p}n_p = 1. \tag{36}$$

[7] The two-qubit case is of the form $\rho = \frac{1}{4}(1 + \vec{P}_1\cdot(\vec{\sigma}\otimes\mathbf{1}) + \vec{P}_2\cdot(\mathbf{1}\otimes\vec{\sigma}) + \overleftrightarrow{C}\cdot(\vec{\sigma}\otimes\vec{\sigma}))$, where $\vec{P}_1 = \mathrm{Tr}[(\vec{\sigma}\otimes\mathbf{1})\cdot\rho]$ and $\vec{P}_2 = \mathrm{Tr}[(\mathbf{1}\otimes\vec{\sigma})\cdot\rho]$ are the polarization vectors for qubits 1 and 2 and $\overleftrightarrow{C} = \mathrm{Tr}[(\vec{\sigma}\otimes\vec{\sigma})\cdot\rho]$ is the 3×3 spin correlation tensor. Note the number of polarization plus correlations are $2^{2n_q} - 1 = 3$ (for one qubit) and 15 (for two qubits).

```
 denE[0] = ρ[initstate];
denE[nv_] : denE[nv] =
 (OPA[nv]).(prob * denE[nv - 1] + eps * WaitAll[Sum[ParallelSubmit[{i}, probx[i] *
          (OP[i, nv] .denE[nv - 1]). (Adj[OP[i, nv]]) ], {i, 1, 2 * nprocs}]]).(Adj[OPA[nv]] )
```

Fig. 10. Here denE[0[] is the initial density matrix ρ_0 and denE[nv] denotes the ensemble averaged density matrix after nv steps. The algorithm operators "OPA[nv]" are setup within the code, and the noise operators OP[i, nv] act on the ith processor at the nvth step. These noise operators are held fixed after they are randomly generated.

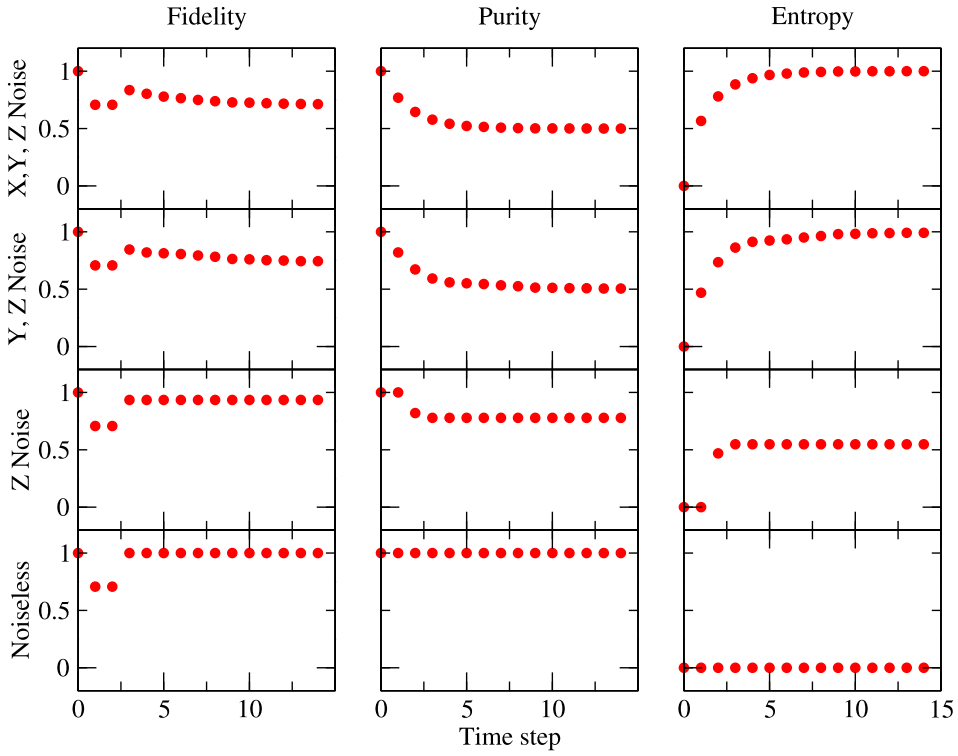

Fig. 11. The fidelity (left column), purity (central column) and entropy (right column) evolution is displayed for one qubit initially in a state $|\psi_0\rangle = |0\rangle$, that is subject to a Hadamard at step 1 and another Hadamard at step 3, for various noise scenarios. The top plot is for X, Y and Z noise, the next plot down has Y and Z noise, and the next Z-noise only. The bottom plot is the noiseless channel, for which the fidelity returns to one, after dropping to $1/\sqrt{2}$, clearly reflects the action of 2 sequential Hadamards.

Here we assume that each $\Omega^{(n)\dagger}\Omega^{(n)} = 1$, and hence that $\mathrm{Tr}[\Omega^{(n)}\rho_o\Omega^{(n)\dagger}] = \mathrm{Tr}[\rho_o] = 1$. In addition, $\rho_f^\dagger = \rho_f$.

This multiuniverse approach is illustrated in Fig. 9. For the pure storage case the algorithm operators $\Omega_A, \tilde{\Omega}_A, \ldots$ are all set equal to unit operators. See later for a simple non-trivial case.

5.2.2. Multiverse and POVM

The above representation can also be cast in the POVM (Positive Operator Valued Measure) and Kraus operator form. The evolution can be expressed as

$$\rho_f = \sum_{n=0}^{n=n_p} \tilde{\Omega}^n \rho_0 \tilde{\Omega}^{n\dagger}, \tag{37}$$

where we define $\tilde{\Omega}^0 = \sqrt{\mathfrak{p}}\,\mathbf{1}$, and $\tilde{\Omega}^{n>0} = \sqrt{\frac{\epsilon}{n_p}}\Omega^n$. This is the form known as POVM, which can be deduced [17] from embedding a quantum system in an environment, which is then projected out. This evolution form can also be used to deduce the Lindblad [18] equation for the evolution of a density matrix subject to environmental interactions. Here we arrive at these forms from a simple multiuniverse approach.

5.2.3. Multiverse and classical limit

Our task is to set up this multiuniverse approach using the parallel, multi-processor features of Mathematica. Eq. (35) describes the evolution of a density matrix after one set of operators act in the various possible paths. A subsequent set of operators is described by

$$\rho_F = \mathfrak{p}\rho_f + \frac{\epsilon}{n_p}\sum_{n=1}^{n=n_p}\left(\Omega^{(n)}\rho_f\Omega^{(n)\dagger}\right). \tag{38}$$

As this evolution process continues to be subject to additional noise operators, the density matrix evolves into a diagonal or classical form. In this way the noise yields a final classical density matrix with zero off-diagonal terms; this is the decoherence caused by a quantum system interacting with an environment. If the qubit states are not degenerate and the noise is of thermal distribution, the density matrix in the classical limit will evolve towards the thermodynamic form $\exp(-\frac{H}{kT})$. At every stage, one can track the von Neumann entropy ($S(\rho) = -\mathrm{Tr}[\rho\ln[\rho]]$), the Purity ($\mathrm{Tr}[\rho\cdot\rho]$), and the Fidelity ($F[\rho,\rho_0] = \mathrm{Tr}[\sqrt{\sqrt{\rho_0}\cdot\rho\cdot\sqrt{\rho_0}}]$),[8] of the system. In addition, the eigenvalues of the system can be monitored where in the classical limit the eigenvalues all approach $\frac{1}{2^{n_q}}$, and the entropy goes to $S[\rho] \to n_q$. Subsystem entropy and eigenvalues can also be examined.

[8] To evaluate this complicated expression, we find the eigenvalues of ρ. ρ_0 and then form the sum $\sum_n \sqrt{|\tilde{\lambda}_i|}$, to obtain a good approximate value.

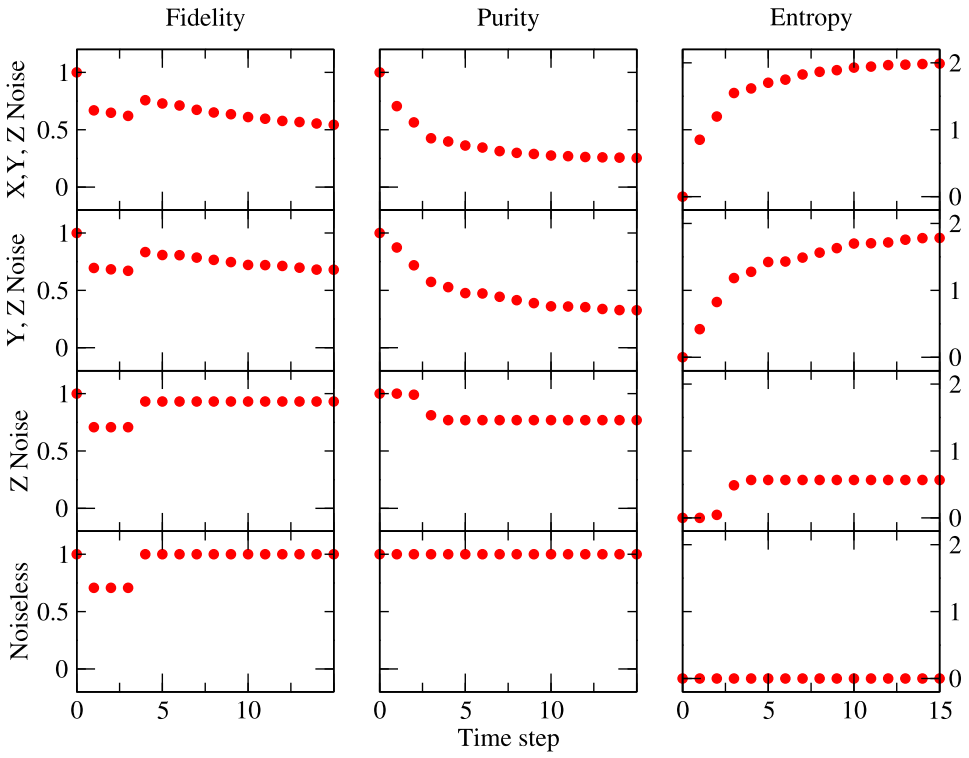

Fig. 12. The fidelity (left column), purity (central column) and entropy (right column) evolution is displayed for two qubits initially in a state $|\psi_0\rangle = |00\rangle$, that is subject to a $\mathrm{CNOT}_{12}\mathcal{H}_1$ and later an $\mathcal{H}_1\mathrm{CNOT}_{12}$, for various noise scenarios. The top plot is for X, Y and Z noise, the next plot down has Y and Z noise, and the next Z-noise only. The bottom plot is the noiseless channel, for which the fidelity returns to one, after dropping to $1/\sqrt{2}$, which clearly reflects the action of the 2 sequential operators.

```
(*  Wigner Rotation Function--used to represent a   rotation  with Euler angles a,b,c  *)

UE1[a_, b_, c_] := MatrixExp[-I a s[3] / 2].( Cos[b / 2]  -Sin[b / 2] ).MatrixExp[-I c s[3] / 2];
                                             ( Sin[b / 2]   Cos[b / 2] )

(*  Set ss[4] as the Unitary rotation matrix *)
i4select = 1
If[i4select == 0,
 ss[4]  = UE1[RandomReal[{0, 2 Pi}], RandomReal[{0, Pi}], RandomReal[{0, 2 Pi}]],
 ss[4] := UE1[RandomReal[{0, 2 Pi}], RandomReal[{0, Pi}], RandomReal[{0, 2 Pi}]]]

ss[4]
ss[4]
ss[4] == ss[4]
If[i4select == 0, Print[" Fixed random unitary rotation matrix case"],
 Print[ " Variable random unitary rotation matrix case" ] ]
```

Fig. 13. This is an option to use a general rotation as the noise operator as an alternative to the Pauli operators. The s[4] location can be used to include this option.

5.3. Multiverse algorithms and errors

The evolution of the density matrix, with noise included via the multiverse approach on the available processors, can also be implemented when an algorithm is included. The procedure is to act with the algorithm gate operators after each ensemble averaged density matrix is formed. The explicit expression is given in Eq. (35) which for the nth step is

$$\rho_{n+1} = \Omega_A \left[\rho_n + \frac{\epsilon}{n_p} \sum_{k=1}^{k=n_p} \sum_{s=1,2} p_s \Omega_{ks} \rho_n \Omega_{ks}^\dagger \right] \Omega_A^\dagger, \tag{39}$$

where Ω_A are the gates for the specific algorithm and Ω_{ks} are the noise operators on the kth processor for two cases, $s = 1$ denotes a one-qubit noise operator hitting one-qubit and $s = 2$ denotes one qubit operators hitting two separate qubits. The factors p_k are assumed to be $p_1 = 0.95$ and $p_2 = 0.05$, so that the one-qubit hits have a net probability $(\epsilon * p_k)$ of 19% and the double hit case a net probability of 1%. The algorithm operators Ω_A are applied to all processors, but implemented by evaluation on the master processor.

In QCWAVE, the above steps are implemented in the notebooks **MV1-Noise**, **MV2-Noise** and **MVn-Noise**, for systems consisting of 1, 2 or n qubits. The key step is shown in Fig. 10, where denE[n] denotes the ensemble averaged density matrix at stage n, and the parallel part of the command distributes the evaluation of the noise over the "nprocs" processors, which is doubled to account for the "s" label in Eq. (39).

A simple algorithm is illustrated in **MV1-Noise**; namely, one starts with the state $|0\rangle$ which is then hit by a Hadamard $\mathcal{H}$, and after an interlude of noise, another Hadamard hits, followed by a long sequence of noise. Without noise this process correspond to a rotation to the x-axis and then a rotation back to the z-axis. One also sees the polarization vector rotated, rotated back and then, af-

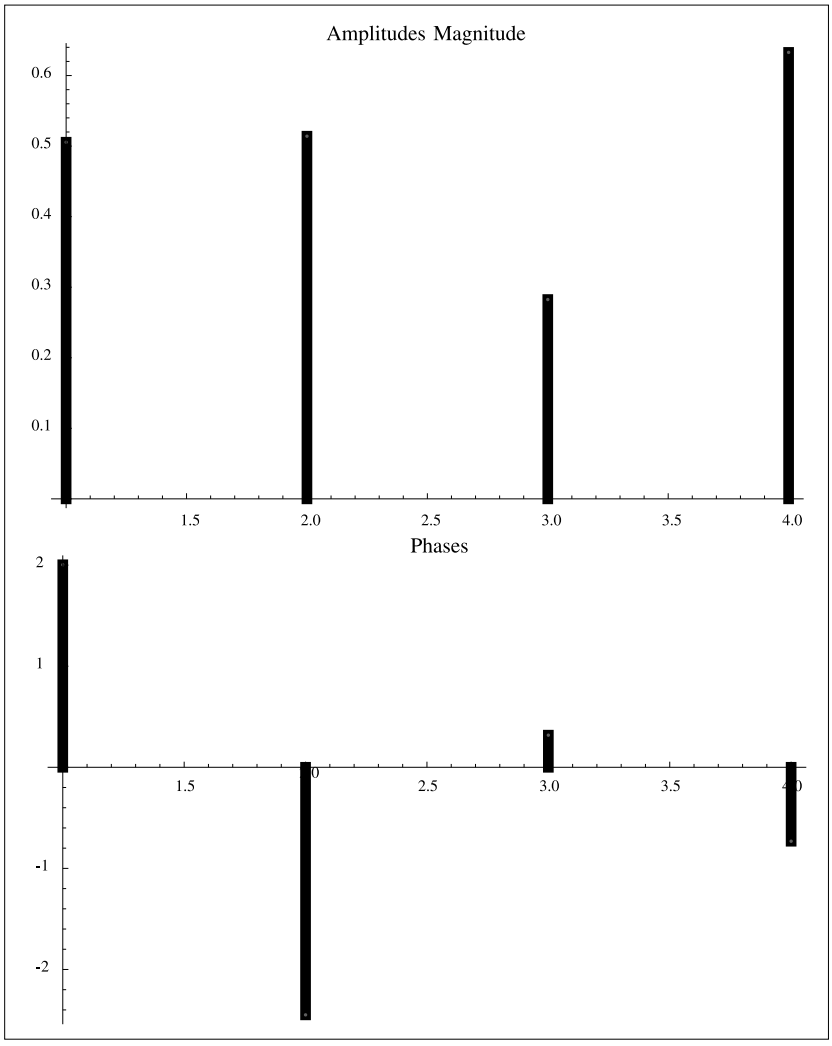

Fig. 14. Amplitudes command displays amplitudes as magnitude and phase bar graphs – see **QCWave.m**.

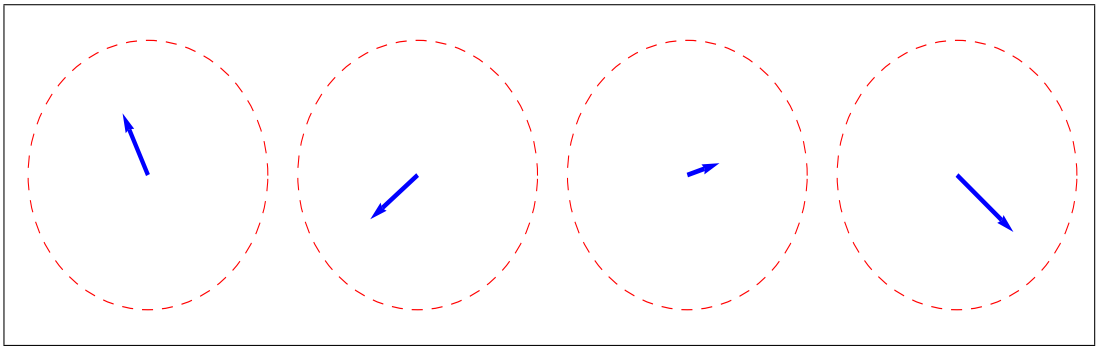

Fig. 15. MeterGraph displays amplitudes as argand plots – see **QCWave.m**.

ter a sequence of noise hits, decay to zero and density matrix then evolves into a diagonal form, with both eigenvalues equal to 1/2. How is this simple process affected by noise during these steps? To answer that question the entropy, purity and fidelity evolution are tracked. The results from **MV1-Noise** are illustrated in Fig. 11.

Another simple algorithm is illustrated **in MV2-Noise**; namely, one starts with the state $|00\rangle$, qubit one is then hit by a Hadamard $\mathcal{H}$, and after an interlude of noise, a CNOT gate acts on both qubits. This is the algorithm for producing a Bell state $\mathrm{CNOT}_{1,2}\mathcal{H}_1|00\rangle = \frac{1}{2}(|00\rangle + |11\rangle)$. This is followed by an inverse Bell operator $\mathcal{H}_1\mathrm{CNOT}_{1,2}$, and then a long sequence of noise. The density matrix then evolves into a diagonal form, with all 4 eigenvalues equal to 1/4. The two polarizations, and the spin correlations are displayed along with the evolution of the eigenvalues, the entropy, purity and fidelity. The results from **MV2-Noise** are illustrated in Fig. 12.

Clearly, more sophisticated algorithms can be invoked. We next consider how to monitor and correct for the noise.

```
matTel2 = ⎛ 1  1   1   C   H   1  M  DB  ⎞
          ⎜ 1  1   1   1   C   H  M  DB  ⎟
          ⎜ H  C   1  CX   1   1  M  DB  ⎟
          ⎜ H  1   C   1  CX   1  M  DB  ⎟
          ⎜ 1 CX   1   1   1   1  1 BML  ⎟
          ⎝ 1  1  CX   1   1   1  1  BM  ⎠ ;

nqset = 2;
initialKets[matr_] := initialKetsA[matr]
Circuit[matTel2, {}];
Show[%, PlotLabel → Style[" Two Qubit Teleportation ", 18],
 ImageSize → 450]
initialKets[matr_] := initialKetsq[matr]
```

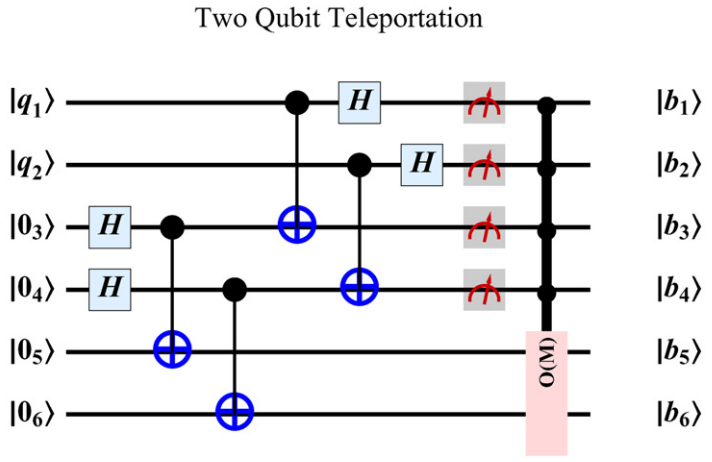

Fig. 16. A sample circuit diagram as produced in notebook **TeleportationW** is shown here. The initialKetsA sets the initial array sequence as: $|q_1\rangle, |q_2\rangle, |0_3\rangle, \ldots, |0_6\rangle$.

5.4. Multiverse and error correction

5.4.1. Simulation of error correction

Error correction (EC) typically involves encoding the qubits using extra qubits, then entangling those encoded qubits with auxiliary qubits. Measurements are made on the auxiliary qubits, so as not to disturb the original encoded qubits. Those measurements provide information as to whether an error has occurred, its nature and where it acted. Hence a remedial gate can be applied to undo the error. If desired, the encoded qubits can then be decoded and the original error-free qubit restored. That process is illustrated for simple X and Y errors on one-qubit EC and for Shor's nine-qubit EC code in notebooks **EC3x**, **EC3z and Shor9Tutorial**. More sophisticated EC codes are available in the literature, along with a general theoretical framework [19]. This kind of EC has to be constantly invoked as an algorithm evolves, which is a rather awkward and qubit-costly process. Error in the gates themselves is an additional concern, usually one assumes perfect gates, with errors (noise) occurring only in-between application of the gates.

For our purpose, instead of applying the procedures outlined in the above EC notebooks, we simulate EC by a rather simple procedure. In the notebooks, the operators ss[i] are set equal to the Pauli operators s[i] to generate noise. By replacing ss[1] by s[0] (the 2×2 unit operator), all X-noise is turned off "by hand." Then a rerun is generated which has the same structure as with the noise, except the X-noise has been removed. Similar steps can be used to remove the Y- and the Z-noise operators. In that way a set of results can be generated ranging from a full noise, to partial noise to no noise cases. Examples from **MV1-noise** and **MV2-noise** are presented in Figs. 11 and 12.

If the user wishes to invoke other noise operators, that can be accommodated as well. For example, a general unitary random rotation can be used as a noise operator by invoking the form shown in Fig. 13. Thus "ss[4] = UE1" is used to turn on such rotations. Replacing s[4] by s[0] again provides a way to turn this operator off to remove that noise element.

With this simple scheme, one can study many more noise and EC scenarios. For example, in the notebook **MVn-Noise** the case of an algorithm for five-qubits is presented. The algorithm consists of a Hadamard followed by a CNOT chain, i.e. $\mathrm{CNOT}_{15}\mathrm{CNOT}_{14}\mathrm{CNOT}_{13}\mathrm{CNOT}_{12}\mathcal{H}_1|00\rangle$, including noise in-between and after the 9th step. Detailed examination of the entropy, fidelity, purity and eigenvalues and noise is presented within that notebook. Of particular interest is the EC simulation results when the noise operators are turned of sequentially.

6. Additional features

6.1. Amplitude displays

The amplitude coefficients C_n can be displayed in various ways using the commands **Amplitudes** and **MeterGraph** as illustrated in Figs. 14 and 15.

6.2. Dirac form

The command DForm has already been demonstrated in Figs. 1–4 and 6. Another Dirac form has be invoked in QCWAVE as shown in Fig. 4. A more extensive Dirac notation scheme has been provided by José Luis Gómez-Muñoz et al. in Ref. [3].

6.3. Circuit diagrams

Illustrations of circuit drawing are included throughout the notebooks, with the **CircuitTutorial** notebook providing an over-

view. The commands are all defined in **Circuits.m**. One example is given in Fig. 16.

6.4. Upgraded applications

Upgraded versions of Grover [20], Teleportation [21] and Shor [22] algorithms are included in the present version.

7. Conclusion and future applications

This package will hopefully be instructive and useful for applications to error correction studies. Hopefully users will contribute to improvements and extensions and for that purpose we are developing an interactive web page. When MPI becomes available on Mathematica, there will be another opportunity to upgrade QCWAVE to a full research tool.

Application to explicit quantum computing problems, such as study of non-degenerate states and the associated phase factors, errors in gates themselves (where the gates are produced by explicit pulses), and the direct application of EC schemes are among the possible future applications. Novel EC schemes, such as stabilizing pulses or EC stable spaces could be additional fruitful applications.

Acknowledgements

This project was supported earlier in part by the U.S. National Science Foundation and in part under Grants PHY070002P and PHY070018N from the Pittsburgh Supercomputing Center, which is supported by several federal agencies, the Commonwealth of Pennsylvania and private industry. B.J.-D. is supported by a CPAN CSD 2007-0042 contract. This work is also supported by Grants No. FIS2008-01661 (Spain), and No. 2009SGR1289 from Generalitat de Catalunya.

Appendix A. Setting up processors with Mathematica

In order to use parallel processing with Mathematica, one needs to first gain access to several processors. There are other ways to do this, but, the commands we used are given in Fig. A.1. Note that the user needs to be sure that the ssh (secure shell) access is working and accesses the Mathematica command on the other machines (which could be Macs or PCs or a combination of them).

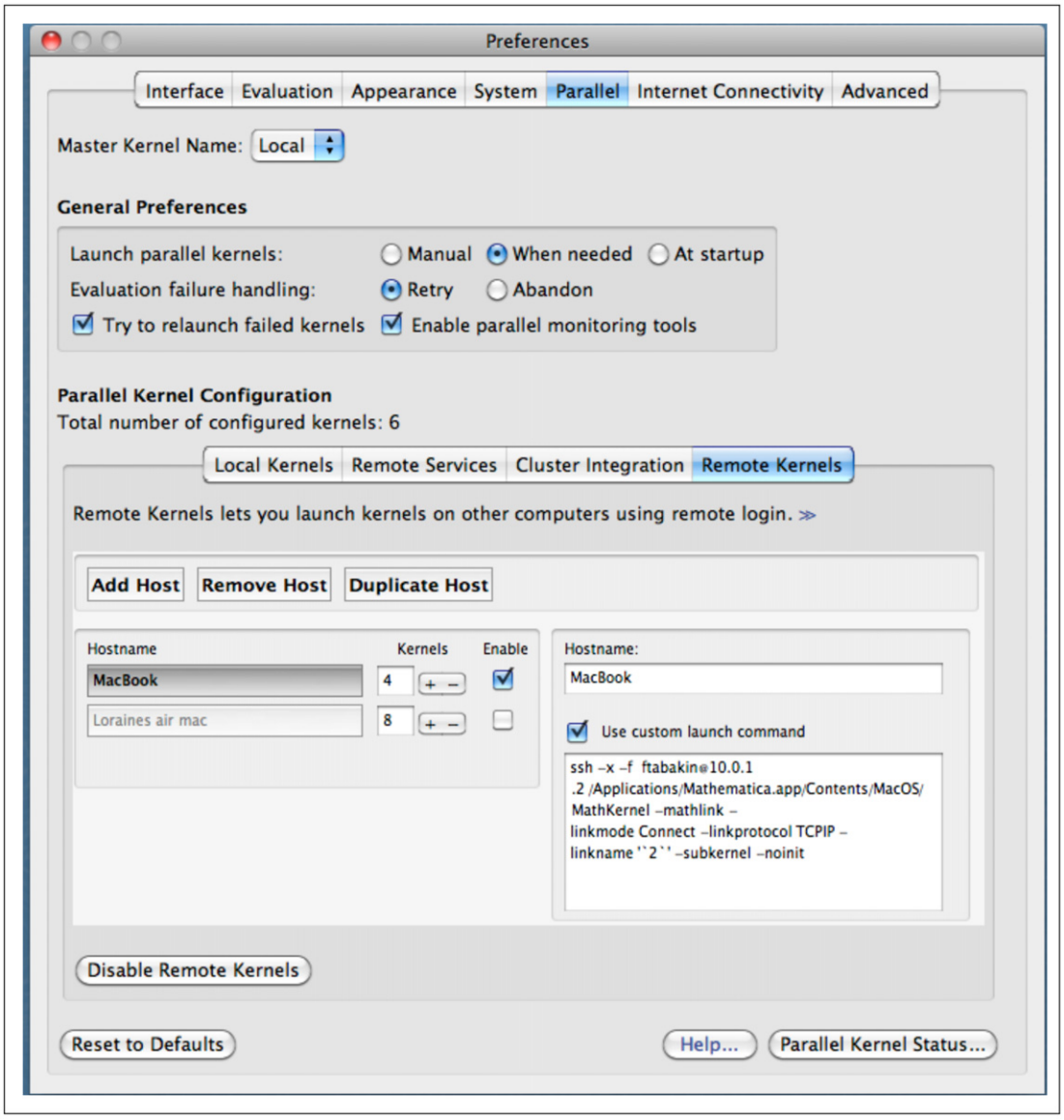

Fig. A.1. The preferences setup to access other processors – use "Evaluation/Parallel Kernel Configuration/Remote Kernels" to get to this page. You also need to set the Local Kernels entry. The user also needs to establish an ssh link to the other machines and be sure it properly accesses the Mathematica installed on all processors.

```
Needs["QDENSITY`Qdensity`"];
ParallelNeeds["QDENSITY`Qdensity`"];
Needs["QDENSITY`QCWave`"];
ParallelNeeds["QDENSITY`QCWave`"];
Needs["QDENSITY`Circuits`"];
```

Fig. A.2. The commands needed to invoke the packages on parallel processors. **QCWave.m and Circuits.m** are add-ons to the original **QDensity.m** package.

In addition, one needs to setup the basic programs and requisite packages on all the processors used. For that the initializations shown in Fig. A.2 are needed.

References

[1] Bruno Juliá-Díaz, Joseph M. Burdis, Frank Tabakin, QDENSITY — A Mathematica quantum computer simulation, Comput. Phys. Comm. 174 (2006) 914–934; also see: Comput. Phys. Comm. 1 (80) (2009) 474, http://www.pitt.edu/~tabakin/QW/, and for our QCWAVE webpage.

[2] See http://www.wolfram.com/mathematica/.

[3] José Luis Gómez-Muñoz, A free Mathematica add-on for Dirac Bra–Ket Notation, Quantum Algebra and Quantum Computing, http://homepage.cem.itesm.mx/lgomez/index.htm.

[4] J. Lapeyre, Qinf quantum information and entanglement package for the Maxima computer algebra system, http://www.johnlapeyre.com/qinf/index.html.

[5] Frank Tabakin, Bruno Juliá-Díaz, QCMPI: A parallel environment for quantum computing, Comput. Phys. Comm. 180 (2009) 948–964.

[6] A list of current quantum computer simulators is kept at http://www.quantiki.org/wiki/List_of_QC_simulators.

[7] H. De Raedt, K. Michielsen, Computational Methods for Simulating Quantum Computers, Handbook of Theoretical and Computational Nanotechnology, American Scientific Publishers, 2006.

[8] Liquantum, http://www.libquantum.de.

[9] P.A.M. Dirac, The Principles of Quantum Mechanics, 4th ed., Oxford University Press, USA, ISBN 0198520115, 1958.

[10] Albert Messiah, Quantum Mechanics, Dover Publications, ISBN 0486409244, 1999.

[11] Michael A. Nielsen, Isaac I. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, 2000.

[12] See: http://www.open-mpi.org/.

[13] A Mathematica MPI code is available at http://daugerresearch.com/index.shtml.

[14] CLOJURATICA is available at http://clojuratica.weebly.com/index.html. It combines the advantages of Mathematica with the Clojure language and "includes a concurrency framework that lets multiple Clojure threads execute Mathematica expressions without blocking others…".

[15] R.P. Feynman, A.R. Hibbs, Quantum Mechanics and Path Integrals, McGraw–Hill, New York, 1965.

[16] R.B. Griffiths, Consistent Quantum Theory, Cambridge University Press, 2003.

[17] John Preskill, Lecture Notes on Quantum Information and Computation, available at http://www.theory.caltech.edu/people/preskill/ph229/, see: Chapters 3.2–3.5.

[18] G. Lindblad, Comm. Math. Phys. 4 (1976) 119.

[19] D. Gottesman, A theory of fault-tolerant quantum computation, Phys. Rev. A 57 (1998) 127–137, quant-ph/9702029.

[20] L.K. Grover, Phys. Rev. Lett. 79 (1997) 325–328.

[21] C.H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, W.K. Wootters, Phys. Rev. Lett. 70 (1993) 1895–1899.

[22] Peter W. Shor, SIAM J. Comput. 26 (5) (1997) 1484.