QUBIT4MATLAB V3.0: A program package for quantum information science and quantum optics for MATLAB

Géza Tóth

ICFO-Institut de Ciències Fotòniques, E-08860 Castelldefels (Barcelona), Spain

Research Institute for Solid State Physics and Optics, Hungarian Academy of Sciences, P.O. Box 49, H-1525 Budapest, Hungary

Abstract

A program package for MATLAB is introduced that helps calculations in quantum information science and quantum optics. It has commands for the following operations: (i) Reordering the qudits of a quantum register, computing the reduced state of a quantum register. (ii) Defining important quantum states easily. (iii) Formatted input and output for quantum states and operators. (iv) Constructing operators acting on given qudits of a quantum register and constructing spin chain Hamiltonians. (v) Partial transposition, matrix realignment and other commands related to the detection of quantum entanglement. (vi) Generating random state vectors, random density matrices and random unitaries.

Key words: quantum register; spin chain; entanglement
PACS: 03.65.Ud 03.67.-a 75.10.Pq

Program Summary

Title of program: QUBIT4MATLAB V3.0
Program summary URL: http://arxiv.org/abs/0709.0948
Program available from:
http://optics.szfki.kfki.hu/∼toth/qubit4matlab.html
http://www.mathworks.com/matlabcentral/fileexchange/
Operating systems: Any which supports MATLAB 6.5; e.g., Microsoft Windows XP, Linux.
Programming language used: MATLAB 6.5; runs also on Octave

Email address: toth@alumni.nd.edu (Géza Tóth).

1 Introduction

Quantum information science[1] is one of the most rapidly developing field in physics. Many calculations can be done analytically, however, many tasks need extensive numerics. Also, analytical calculations can very efficiently be checked for possible errors by calculating concrete examples numerically. The subroutine package presented in this paper was written to help the researcher in quantum information and quantum optics in doing such numerical calculations.

The programming effort necessary for scientific calculations in quantum physics depends a lot on the programming language used. In particular, one has to be able to handle easily large matrices, compute eigenvalues, eigenvectors, etc. This is certainly possible with MATLAB which is an interpreter language for mathematical calculations running both under Windows and Linux. Other alternatives may need extensive usage of complicated bracketing or definitions of complicated data types.

The subroutine package presented is intended to fit smoothly to the philosophy of MATLAB, and makes it possible to write down relatively complex expressions in a concise way. Even simple functions are defined if they are often used or their definition makes the structure of programs clearer. After the programmer runs the main MATLAB code, the relevant quantities, such as ground state energies of Hamiltonians or the smallest eigenvalue of the reduced density matrix, can be printed out writing short expressions interactively.

In this paper the commands offered by the QUBIT4MATLAB V3.0 program package are summarized. The first version appeared in September 2005 on the MATLAB Central File Exchange [2]. Since there are excellent books on quantum physics [3] and quantum information science [1], an introduction on these topics is not given, however, appropriate citations help the reader. The paper is organized as follows. In Sec. 2. basic commands for defining state vectors and density matrices are described. In Sec. 3 commands follow that are related to reordering the qudits or tracing out some of the qudits. Sec. 4 definitions of interesting quantum states, quantum gates and operators are presented. Sec. 5 is about commands for formatted input and output. Sec. 6 lists commands for defining two-qudit interactions and spin chain Hamiltonians. Sec. 7 is about commands related to the separability problem. Sec. 8 is about commands using random matrices. Sec. 9 lists miscellaneous simple commands that make programming easier. Finally, Sec. 10 summarizes commands that give sparse matrices.

The variable names most often used in the descriptions of commands are the following:

- `rho`: Density matrix
- `v,v1,v2,phi,phi1,phi2,psi`: State vector
- `v/rho`: A density matrix is expected. If it is not normalized, then it is automatically normalized. If a state vector is given then it is converted automatically into a properly normalized density matrix.
- `M`: Matrix
- `OP,`OP1, OP2 : Matrix corresponding to a quantum operation
- `N`: Positive integer indicating the number of qudits
- `d`: Positive integer indicating the dimension of qudits
- `k,`l, m, n, k1, k2, n1, n2 : Non-negative integer
- `list`: List of indices of qudits of a qudit register
- `perm`: List of indices indicating how to reorder the qudits (see later in detail)
- `s`: String

The square brackets `[` and `]` are used to indicate optional parameters. If such a parameter is not given then a default value specific to the command is taken. In particular, for `[N]` the default value is the value of the global variable `N`. For `[d]` the default value is 2 (qubits).

2 Bras and kets: State vectors and density matrices

The most basic mathematical object for quantum mechanics is the state vector. It is a vector of complex elements with unit norm. It can be used to describe *pure states*. With QUIBIT4MATLAB it can be defined with the `ket` command. (Next, "bra" and "ket" refers to the usual notation introduced by Dirac [3].) For example,

```
phi0=ket([1 0])
```

defines a two element column vector as a "ket" vector with elements $(1, 0)$. In the $\{|0\rangle, |1\rangle\}$ basis this corresponds to the $|\Phi_0\rangle = |0\rangle$ state. Another example is

```
phi01=ket([1 1]).
```

This defines a column vector with elements $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ which corresponds to $|\Phi_{01}\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. Note that `ket` normalized the vector given in its argument.

The other fundamental object of quantum mechanics is the density matrix. It is a Hermitian positive semi-definite matrix with unit trace. Beside pure states, it can also be used to describe *mixed states.* A density matrix corresponding to the previous state vector can be defined as

```
rho=ketbra(phi01)
```

If we type now `rho` we obtain

```
rho =
    0.5000    0.5000
    0.5000    0.5000
```

`ketbra` normalizes the vector in its argument, in case it is not normalized.

There are also further elements of the Dirac notation implemented in QUBIT4MATLAB. One can define "bra" vectors, that is the conjugate transpose of "ket" vectors. Hence

```
phi01b=bra([1 1])
```

is a row vector with elements $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ which corresponds to $\langle\Phi_{01}| = \frac{1}{\sqrt{2}}\langle 0| + \frac{1}{\sqrt{2}}\langle 1|$. There is one more additional property of `bra`. It computes the complex conjugate of its argument. Thus

```
phi01c=bra([1 i])
```

will result in a row vector with elements $(\frac{1}{\sqrt{2}}, -\frac{i}{\sqrt{2}})$.

Moreover, one can define a "braket" with the command `braket`.

```
braket(phi1,phi2)
```

denotes the scalar product of two state vectors. It is identical to `bra(phi1)*ket(phi2)`. The expression

```
braket(phi1,OP,phi2)
```

where `OP` is a matrix, denotes `bra(phi1)*OP*ket(phi2)`.

Finally, `nm(v/rho)` normalizes its argument. If its argument is a vector `v` then it gives back `v/sqrt(v'*v)`. If the argument is a row vector then it also converts it into a column vector. This results in a unit vector. If the argument is a density matrix `rho` then `nm` gives back `rho/trace(rho)`. Latter results in a matrix with a unit trace.

The summary of commands for implementing the braket notation in MATLAB and related commands are given in the following list.

- `bra(v)`: Dirac's "bra" vector. Normalizes `v` and converts it into a column vector in case it was a row vector

- `ket(v)`: Dirac's "ket" vector. Normalizes `v`, carries out an element-wise complex conjugation, and converts `v` into a row vector in case it was a column vector
- `ketbra(v)`: Obtaining a density matrix from the state
- `ketbra2(v/rho)`: Like `ketbra(v)`, however, a density matrix can also be given as an argument. If this is the case, `ketbra2` normalizes `rho`.
- `braket(v1,v2)`: Equivalent to `bra(v1)*ket(v2)`
- `braket(v1,OP,v2)`: Equivalent to `bra(v1)*OP*ket(v2)`
- `ex(OP,v/rho)`: Expectation value of an operator for a state vector or a density matrix. For normalized `v` and `rho`, it is equivalent to `bra(v)*OP*ket(v)` or `trace(OP*rho)`.
- `va(OP,v/rho)`: Variance of an operator for a state vector or a density matrix. For normalized `v` and `rho`, it is equivalent to `bra(v)*OP^2*ket(v)-(bra(v)*OP*ket(v))^2` or `trace(OP^2*rho)-trace(OP*rho)^2`.
- `nm(v/rho)`: Normalization of a state vector or a density matrix.

3 Basic operations on the quantum register: Reordering qudits

The basic object QUBIT4MATLAB handles is an array of N qudits of dimension d. These qudits are numbered from 1 to N. For example, if `phi1` and `phi2` are single-qudit state vectors, then a two-qudit state vector can be defined as

```
phi=kron(phi2,phi1).
```

This defines a product vector. The state of qubit #1 is `phi1` and the state of qubit #2 is `phi2`. In order to make it easier to handle multi-qudit registers, a Kronecker product command with more than two arguments is defined: `mkron`. For example, `mkron(M1,M2,M3)=kron(kron(M1,M2),M3)`. Moreover, there is also a "Kronecker power" function that multiplies a matrix with itself given times using the Kronecker product. For example,
`pkron(M,4)=kron(kron(kron(M,M),M),M)`.

Many of the following commands have `N` and `d` as parameters. Typically, if `d` is omitted then it is considered to be 2, while if the parameter `N` is omitted then the value of the global variable `N` is taken instead. Other speciality of the commands is that at most of the places where a density matrix is expected, a state vector can also be given. It is automatically converted into a normalized density matrix.

Next, let us see an example. Let us define the state $|\phi\rangle = (|00\rangle + |11\rangle)|1\rangle/\sqrt{2}$ as

```
phi=ket([0 1 0 0 0 0 0 1])
```

Then we can flip the last two qudits with the command

```
phi2=reorder(phi,[3 1 2])
```

When we print out `phi2`, the result is $(|010\rangle + |111\rangle)/\sqrt{2}$. Thus the right and the middle qubits are exchanged. In general, the second argument of `reorder` is a list describing, how to reorder (permute) the qubits. For N qubits, $N, N-1, N-2, ..., 2, 1$ corresponds to the original configuration. Thus the following command does not change the state `phi`

```
phi3=reorder(phi,[3 2 1])
```

The command

```
phi4=reorder(phi,[1 3 2])
```

shifts the qudits cyclically to the right. When we print out `phi4`, the result is $(|100\rangle + |111\rangle)/\sqrt{2}$. The meaning of the parameter describing the permutation is even clearer if we write the numbering of qudits and the row vector describing the permutations below each other

```
[3 2 1]
[1 3 2]
```

This means that qudit #3 will move to qudit #1, qudit #2 will move to qudit #3, and qudit #1 will move to qudit #2. The command `reorder` also works for qudits with a dimension larger than two, if a third argument is given with the dimension.

Another fundamental operation is computing the reduced density matrix, after tracing out some of the qubits. The following operation shows how to compute the reduced state, after tracing out qubits 2 and 3

```
rho_red=remove(phi,[3 2])
```

The second argument contains the list of qubits that have to be traced out. This command also works for qudits with a dimension larger than two, if a third argument is given. A related command is `keep`. It is essentially the same as `remove`, except that the list of the qubits that should be kept must be given.

The summary of commands for ordering/reordering qudits are given in the following list.

- `mkron(M1,M2,M3,...)`: Kronecker product with several arguments
- `pkron(M,n)`: Kronecker product of `M` with itself `n` times

- `reorder(rho,perm,[d])`: Reorder the qudits of the density matrix `rho` according the permutation given in `perm`. If a state vector is given instead of `rho`, then the result is also a state vector.
- `reordermat(perm,[d])`: The matrix corresponding to the quantum operation realizing a given permutation of qudits. It gives the matrix that realizes the permutation given by `perm` on a state vector of the qudit register.
- `reordervec(perm,[d])`: The vector corresponding to the permutation of qudits. The *ith* element of the vector tells us where to move the *ith* element of a state vector during the multi-qudit register reordering.
- `shiftquditsleft(rho,[d])`: Shifts the qudits of `rho` to the left
- `shiftquditsright(rho,[d])`: Shifts the qudits of `rho` to the right
- `swapqudits(rho,k,l,[d])`: Swaps the qudits `k` and `l` of a quantum state `rho`
- `remove(rho,list,[d])`: Reduced density matrix obtained from `rho`, after the qudits given in `list` are traced out
- `keep(rho,list,[d])`: Reduced density matrix obtained from `rho`, after the qudits *not* given in `list` are traced out. Thus only the qudits given in `list` are kept.

4 Definitions of important quantum states, quantum gates and operators

There are several commands defining important quantum states and useful operators. E.g., the simple command `paulixyz` defines the Pauli spin matrices `x`, `y` and `z`. Moreover `e` is defined as the 2×2 identity matrix. `paulixyz` is often used in programs dealing with spin chains. The list of such commands are:

- `ghzstate([N])`: State vector for the `N`-qubit Greenberger-Horne-Zeilinger state [4]
- `wstate([N])`: State vector for the `N`-qubit W-state
- `cstate([N])`: State vector for the `N`-qubit cluster state [5]
- `rstate([N])`: State vector for the `N`-qubit ring cluster state [6]
- `dstate(e,[N])`: State vector for the `N`-qubit symmetric Dicke state with `e` excitations [7,8]
- `mmstate([d],[N])`: Density matrix of the maximally mixed state of `N` qudits of dimension `d`
- `mestate(d)`: State vector for the maximally entangled state of two qudits of dimension `d`
- `singlet([N])`: State vector for the singlet of N qubits; implemented for `N`$= 2$ and 4.
- `smolinstate`: Density matrix of the state defined by Smolin [9]

- `gstate(Gamma)`: State vector for a graph state that was created with the Ising interaction pattern given in the $N \times N$ matrix `Gamma` [6]
- `gstate_stabilizer(Gamma)`: Gives the generators as a cell array for the stabilizer of the graph state mentioned above [6,10]
- `BES_Horodecki3x3(a)`: Density matrix of Horodecki's 3×3 bound entangled state [11]. Parameter a must have a value between 0 and 1.
- `BES_Horodecki4x2(a)`: Density matrix of Horodecki's 4×2 bound entangled state [11]. Parameter a must have a value between 0 and 1.
- `BES_UPB3x3`: Density matrix of the 3×3 bound entangled state based on unextendible product bases [12]
- `U_CNOT`: 4×4 unitary matrix of a CNOT gate
- `U_H`: 2×2 unitary matrix for the Hadamard gate
- `paulixyz`: Defines Pauli matrices `x`,y, z and `e=eye(2)`
- `su3`: Defines the $SU(3)$ generators (Gell-Mann matrices) `m1`, `m2`,...,`m8` and `ee` as the 3×3 identity matrix
- `su3_alternative`: Defines alternative $SU(3)$ generators [13]

5 Formatted input and output

The basic command for formatted output of a quantum state is `printv`. It prints a state vector as the superposition of the computational basis states. This function works only for qubits at present. The form `printv`(v,treshold) makes it possible to give the threshold below which an element is considered zero. Its usage is demonstrated on the following example

```
printv(phi2)
ans = 0.70711|010>+0.70711|111>
```

The command that can be used for the formatted output of matrices is `decompose`. Its use is shown on the example

```
% Define the pauli spin matrices x,y, and z
paulixyz
% Define Heisenberg interaction for two qubits
H_H=kron(x,x)+kron(y,y)+kron(z,z)
H_H =
     1     0     0     0
     0    -1     2     0
     0     2    -1     0
     0     0     0     1
% Print out the decomposition of H_H
decompose(H_H)
ans =
```

```
xx+yy+zz
```

It decomposes an operator into the linear combinations of products of Pauli spin matrices. Giving a second argument different from zero makes `decompose` print the results in LaTeX format. Giving a third argument makes it possible to give the threshold below which a coefficient is considered zero (and because of that it is not printed).

- `printv(v,[threshold])`: A result is a string, giving the state vector as the superposition of multi-qubit computational basis states. The parameter `threshold` defines the limit value below which a vector element is considered zero. If it is omitted then it is taken to be 10^{-4}.
- `decompose(M,[p],[threshold])`: The result is a string. It contains an expression describing the matrix `M` as the linear combination of products of Pauli spin matrices. If `p` is not zero then the result is given in LaTeX format. The parameter `threshold` defines the limit value below which a coefficient is considered zero. If it is omitted then it is taken to be 10^{-14}.
- `paulistr(s)`: Converts a string describing an operator constructed as a sum of products of Pauli spin matrices into an operator. E.g., `op=paulistr('5*xye+xyz')` is equivalent to `paulixyz;`op=5*mkron(x,y,e)+mkron(x,y,z) .

6 Two-qudit interactions and spin chains

When handling multi-qudit systems, one has to be able to concisely define operators working on a given qudit. The basic command for that is `quditop(OP,k,[N])`. It defines an `N`-qudit quantum operator which corresponds to operator `OP` acting on the `k`th qudits. Qudit position is interpreted as with `reorder`. The dimension of the qudit is deduced from the size of `OP`. If `OP` is sparse, `quditop` will also produce a sparse matrix.

Two-qudit operators can be defined by `twoquditop(OP,k1,k2,[N])`. It defines an `N`-qudit quantum operator which corresponds to the two-qudit operator `OP` acting on the `k1`th and `k2`th qudits. If `OP` is sparse, `twoquditop` will also produce a sparse matrix. The command `interact`(OP1,OP2,n1,n2,[N]) is an alternative way to construct an operator acting on two qubits. It gives an operator acting on qudits `n1` and `n2`, respectively, with operators `OP1` and `OP2`. `N` is the number of qudits. If argument `N` is omitted than the default is taken to be the value of global variable `N`. The dimension of the qudit is obtained from the size of `OP1`.

When modeling spin chains, it is needed to construct expressions with two-body interactions acting between nearest-neighbors. A general form of such a

nearest-neighbor interaction, for aperiodic boundary condition, is

$$H_{\rm nn}(a,b,N) := \sum_{k=1}^{N-1} a^{(k)} b^{(k+1)}, \tag{1}$$

where a and b are some single-qudit operators. Their superscript indicates on which qudit they act on. $H_{\rm nn}(a,b)$ can be obtained by writing `nnchain(a,b,[N])`. The same command for the case of periodic boundary conditions corresponding to

$$H_{\rm nn,p}(a,b,N) := \sum_{k=1}^{N-1} a^{(k)} b^{(k+1)} + a_N b_1 \tag{2}$$

is `nnchainp(a,b,[N])`. When modeling spin chains, it is also needed to define expressions of the type

$$H_{\rm coll}(a,N) := \sum_{k=1}^{N} a^{(k)}, \tag{3}$$

where a is again a single-qudit operator. Such an expression can be obtained writing `coll(a,N)`. They are used for defining external fields for spin chains.

After the general commands, we discuss commands specific to particular spin chains. `ising(B,[N])` gives the ferromagnetic Ising Hamiltonian in a transverse field

$$H_{\rm Ising}(B,N) := -\sum_{k=1}^{N-1} \sigma_z^{(k)} \sigma_z^{(k+1)} + B \sum_{k=1}^{N} \sigma_x^{(k)}. \tag{4}$$

Similarly, `heisenberg(N)` gives the Heisenberg Hamiltonian defined as

$$H_{\rm Heisenberg}(N) := \sum_{k=1}^{N-1} \sigma_x^{(k)} \sigma_x^{(k+1)} + \sum_{k=1}^{N-1} \sigma_y^{(k)} \sigma_y^{(k+1)} + \sum_{k=1}^{N-1} \sigma_z^{(k)} \sigma_z^{(k+1)}. \tag{5}$$

Both commands have versions for periodic boundary conditions: `isingp(B,[N])` and `heisenbergp([N])`. Finally, the XY chain is in external field is defined as

$$H_{\rm XY}(J_x,J_y,B) := J_x \sum_{k=1}^{N-1} \sigma_x^{(k)} \sigma_x^{(k+1)} + J_y \sum_{k=1}^{N-1} \sigma_y^{(k)} \sigma_y^{(k+1)} + B \sum_{k=1}^{N} \sigma_x^{(k)}. \tag{6}$$

The command giving the minimum for the XY chain for separable states is

`xy_classical_ground(Jx,Jy,B)`. As the name suggests, this minimum is the same as the ground state of the classical XY chain.

- `quditop(OP,k,[N])`: Operator acting on the `k`*th* qudit of an `N`-qudit register
- `twoquditop(OP,k1,k2,[N])`: Operator acting on qudits `k1` and `k2` of a `N`-qudit register
- `coll(OP,[N])`: Defines a collective multi-qudit operator
- `interact(OP1,OP2,n1,n2,[N])`: Two-qudit interaction acting on qudits `n1` and `n2` of a `N`-qudit register
- `nnchain(OP1,OP2,[N])`: Spin chain Hamiltonian with a nearest-neighbor interaction with an aperiodic boundary condition
- `nnchainp(OP1,OP2,[N])`: Spin chain Hamiltonian with a nearest-neighbor interaction with a periodic boundary condition
- `ising(B,[N])`: Hamiltonian for an Ising spin chain in a transverse field; aperiodic boundary condition
- `isingp(B,[N])`: Hamiltonian for an Ising spin chain in a transverse field; periodic boundary condition
- `ising_ground(B)`: Computes the ground state energy per qubit for an Ising chain in transverse field `B` for the thermodynamic limit. The form `ising_ground(B,N)` computes the same thing for an N-qubit chain with a peridodic boundary condition.
- `ising_free(B,T)` Free energy per qubit for an Ising chain in a transverse field `B` for the thermal state for the thermodynamic limit.
- `ising_thermal(B,T)`: Internal energy per spin for an Ising chain in transverse field `B` for the thermodynamic limit. The form `ising_thermal(B,N)` computes the same thing for an N-qubit chain with a peridodic boundary condition.
- `ising_classical_ground(B)`: Ground state energy per spin for the classical Ising chain
- `heisenberg(B,[N])`: Heisenberg spin chain Hamiltonian
- `heisenbergp(B,[N])`: Heisenberg spin chain Hamiltonian with a periodic boundary condition
- `xy_classical_ground(Jx,Jy,B)`: Ground state energy per spin for the classical XY chain
- `grstate(H)`: Normalized ground state of a Hamiltonian
- `thstate(H,T)`: Thermal state of a Hamiltonian `H` at temperature `T`. It uses the formula $\rho_T = \exp(-H/T)/\mathrm{Tr}[\exp(-H/T)]$.
- `orthogobs(d)`: Orthogonal observables for a qudit with dimension `d`. The orthogonal observables used are the ones defined in Ref. [14]. That is, these are the observables of the form $|k\rangle\langle k|$, $(|k\rangle\langle l| + |l\rangle\langle k|)/\sqrt{2}$, or $(|k\rangle\langle l| - |l\rangle\langle k|)/\sqrt{2}i$. Let us denote these Hermitian observables by $\{M_m\}_{m=1}^{d^2}$. They satisfy the condition $\mathrm{Tr}(M_m M_n) = 0$ if $m \neq n$ and $\mathrm{Tr}(M_m^2) = 1$.

7 Separability

A quantum state is separable if its density matrix can be written as the convex combination of product states, i.e., as [15]

$$\rho = \sum_k \rho_k^{(1)} \otimes \rho_k^{(2)} \otimes \rho_k^{(3)} \otimes ... \otimes \rho_k^{(N)}, \tag{7}$$

where N is the number of qudits, $p_k \geq 0$ and $\sum_k p_k = 1$. If a quantum state is not separable, then it is entangled.

Entangled states can be used as a resource in several quantum information processing tasks. To decide whether a state is entangled or separable is a very important, yet, in general, unsolved question of quantum information science. However, there are powerful sufficient condition for entanglement in the literature, such as the positive partial transpose (PPT) criterion [17,18] or the computable cross norm-realignment (CCNR) criterion [19,20]. For small systems we even have necessary and sufficient conditions, and even the amount of entanglement can be computed.

For two-qubit systems the entanglement of formation, or equivalently, the concurrence [16] can be computed directly from the density matrix. This can be done with the command `concurrence(rho)`.

A central notion is the partial transposition in quantum information. The following command computes the partial transpose of `ketbra(phi)` with respect to the third qubit

```
rho_pt=pt(phi,3)
```

In general, the second argument is a list of the indices of qudits. The transposition will be carried out for the qubits of this list. The command also works for qudits with dimension larger than two, if a third argument is given. The sum of the absolute values of the negative eigenvalues of the partial transpose is called negativity [21]. This can be computed by the command `negativity`. It needs the same parameters as `pt`, however, it returns a scalar value.

Beside partial transposition, there are other useful rearrangements of the density matrix elements. Such an operation is called realignment. For bipartite system, such a command is `realign`. If the trace-norm of the realigned matrix is larger than one then the state is entangled. This can be checked by the `ccnr` command.

There are some numerical routines looking for the maximum of an operator for product states. This is useful for experiments: If the operator is measured and a

larger expectation value is obtained then we know that the state is entangled. The routines are based on simple annealing-like search for the maximum. While it is not guaranteed that these routines find really the global maximum, they work quite well for systems of a couple of qubits [22]. For example, the maximum for separable states for a 4-qubit operator can be obtained

```
>> % Define the pauli spin matrices x,y, and z
paulixyz
% Define Collective operators
Jx=coll(x,4)/2; Jy=coll(y,4)/2;
% Print out the maximum for separable states
ms=maxsep(Jx^2+Jy^2)
ms =
   5.0000
```

Analitical calculation shows that this is indeed the maximum for separable states [8]. Now, the maximum for quantum states in general can be obtained as

```
>>  maxeig(Jx^2+Jy^2)
ans =
    6
```

Thus there are quantum states for which the expectation values of `Jx^2+Jy^2` is larger than 5. These states are all entangled.

Finally, we briefly mention, that in a multi-qubit experiment it is typically not enough to show that a quantum state is entangled. One has to prove that *genuine multi-qubit entanglement* was present [27]. It is defined as follows. If a pure state can be written as state separable with respect to some bipartition of the qubits then it is called biseparable. E.g., such a state is $(|01\rangle - |10\rangle)/\sqrt{2} \otimes (|01\rangle - |10\rangle)/\sqrt{2}$. This state is the tensor product of two two-qubit singlets. While it is entangled, it is separable with respect to the bipartition (12)(34). A mixed state is biseparable if it can be obtained by mixing biseparable pure states. If a quantum state is not biseparable then it is genuine multi-qubit entangled. Several of the commands in QUBIT4MATLAB are related to the detection of genuine multi-qubit entanglement. For example, the maximum for biseparable states for the previous operator can be obtained as

```
>>  maxb(Jx^2+Jy^2)
ans =
   5.2320
```

Analytical calculation gives $\frac{7}{2} + \sqrt{3} \approx 5.2321$ [8,28].

The list of commands related to separability problem is summarized in the following table.

- `pt(rho/v,list,[d])`: Partial transposition of `rho`. `list` contains the list with indices of qudits. The qudits on this list are transposed.
- `pt_nonorm(M,list,[d])`: Like `pt` but the matrix given is not normalized.
- `negativity(rho/v,list,[d])`: Negativity of `rho`
- `realign(M)`: Computes the matrix obtained from realigning `M`.
- `mrealign(M,iperm,[d])`: Computes the matrix obtained from realigning the multi-qudit operator `M`. `iperm` has now twice as many elements as the number of qudits. It shows how to permute the indices of the density matrix, if for the parties we use multiply indices. E.g., $\rho_{i_1 i_2 i_3, j_1 j_2 j_3}$ with $i_k, j_k = 0, 1$ would describe a three-qubit state.
- `cnnr(rho/v)`: Gives directly the trace norm of the realigned matrix. A state is entangled, if the trace norm of the realigned matrix is larger than one.
- `optspinsq(rho)`: Optimal spin squeezing inequalities [29]. Gives back a negative value if the multi-qubit state rho is detected as entangled by the optimal spin squeezing inequalities. The form `[fmin,f123]=optspinsq(rho)` gives back in `f123` a three element array. Each element of the array gives -1 times the violation of the corresponding spin squeezing inequality. `fmin` is the minimum of the three values. If one of them is negative then the state is detected as entangled. Beside the inequalities themselves, a method is also implemented that looks for the optimal choice of x, y, and z coordinates. (See Ref. [29].)
- `maxsep(OP,[d],[par])`: Looks for the maximum for product states. It does not necessarily find the global maximum, but for small systems it produces good results. `d` gives the dimension of the qudits. `par` gives the parameters for the search algorithm. It has three elements. First element: Number of random trials in the first phase. Second element: Number of random trials in the second phase. In the second phase the routine looks for the maximum around the maximum found in the first phase. Third element: Constant determining accuracy. The default value for `par` is `[ 10000 20000 0.005 ]`.
- `maxsymsep(OP,[d],[par])`: Computes the maximum only for a special case, i.e., for symmetric states. Because of that it is faster than `maxsep`. `d` gives the dimension of the qudits. `par` plays the same role as for `maxsep`.
- `maxbisep(OP,list,[par])`: Gives maximum value for an operator for biseparable states. `list` determines the biparitioning. That is, the bipartition is considered in which qubits given in `list` are in one group, the rest of the qubits are in the other group. `par` plays the same role as for `maxsep`. At the moment works only for qubit registers.
- `maxb(OP,[par])`: Considers the maximum for all bipartitions. It is based on numerical optimization. `par` plays the same role as for `maxsep`. It can be used, for example, when making calculations for entanglement witnesses detecting genuine multi-qubit entanglement in experiments. One can check with it the bounds calculated analytically. At the moment works only for

qubit registers.

- `schmidt(v,list)`: Schmidt coefficients for a pure state `v` for the bipartition determined by `list`. At the moment works only for qubit registers.
- `overlapb(v)`: Maximum overlap with biseparable states for a state vector `v`. It is not based on numerical search, always gives correct result. In fact, the maximum overlap is just the square of the largest Schmidt coefficient over all bipartition [30]. At the moment works only for qubit registers.

8 Commands using random matrices

Very often it is needed to generate random state vectors, density matrices or random unitaries. QUBIT4MATLAB has a number of commands for these.

A random state vector (a vector of complex elements with unit length) can be generated in the following way [31]: (i) Generate a vector such that both the real and the imaginary parts of the vector elements are random numbers that have a normal distribution with a zero mean and unit variance. (ii) Normalize the vector. It is easy to prove that the random vectors obtained this way are equally distributed on the unit sphere.

An N-qudit random density matrix with a distribution uniform according to the Hilbert-Schmidt norm can be obtained in two steps [32]: (i) Generate a a $2N$-qudit pure state with a distribution uniform over the unit sphere. (ii) Trace out half of the qudits.

Finally, an $N \times N$ random unitary with a distribution uniform according to the Haar measure can be obtained as follows [31]: (i) Generate N vectors with N complex elements and with a uniform distribution over the unity sphere. (ii) Orthogonalize the vectors.

The list of commands using these ideas is the following:

- `rvec(N,d)`: Gives a random state vector for a system of `N` qudits of dimension `d`. The distribution is uniform on the complex sphere of radius 1.
- `rproduct(N,d)`: Gives the tensor product of `N` random state vectors of size `d`.
- `rdmat(N,d)`: Gives a random density matrix for a system of `N` qudits of dimension `d`. The distribution of the matrix is uniform according to the Hilbert-Schmidt norm.
- `runitary(N,d)`: Gives a random unitary matrix for a system of `N` qudits of dimension `d`. The distribution of the matrix is uniform according to the Haar measure.
- `twirl(rho,[d],[Nit])`: Twirls the multi-qudit density matrix `rho`. `d` is the

dimension of the qudits. `Nit` is the number of iterations. The algorithm used is not simply averaging over random unitaries and converges very fast (for the algorithm, see Ref. [33].) If d is omitted then it is taken to be 2. If `Nit` is omitted, it is taken to be 100. The form `[rho2,difference]=twirl(rho)` gives also the norm of the difference between the original and the twirled state. The difference is computed through the matrix norm $\|A\| = \sum_{kl} \|A_{kl}\|^2$. The difference is zero for Werner states.

- `twirl2(rho,[d],[Nit])`: Gives the maximal difference between a multi-qudit state `rho` and the state obtained from it by a multilateral unitary rotation of the form $U \otimes U \otimes U \otimes ... \otimes U$. The difference is computed through the matrix norm $\|A\| = \sum_{kl} \|A_{kl}\|^2$. `d` is the dimension of qudits. If omitted then it is taken to be 2. `Nit` is the number of random unitaries used for finding the maximum. If omitted then it is taken to be 100. The form `[difference,U0]=twirl2(rho)` gives also back the unitary `U0` for which the difference is the largest between the original and the rotated state.

9 Miscellaneous simple commands

The following simple commands help to write programs concisely. We discuss two of them in more detail.

`trnorm(M)` gives the trace-norm of the matrix `M`. The trace-norm is defined as $||M|| = \mathrm{Tr}(\sqrt{M^\dagger M})$. It equals the sum of the singular values of the matrix.

`addnoise(rho/v,p)` adds white noise to quantum state, i.e., it computes

$$\rho'(p) = p\rho + (1-p)\frac{\mathbb{1}}{\mathrm{Tr}(\mathbb{1})}. \tag{8}$$

The second term is the appropriately normalized identity matrix which corresponds to the density matrix of the completely mixed state.

- `proj_sym(N,[d])`: Projector to the symmetric subspace of an `N`-qudit register with qudits of dimension `d`. At the moment only `N`= 2 is implemented.
- `proj_asym(N,[d])`: Projector to the antisymmetric subspace of an `N`-qudit register with qudits of dimension `d`. At the moment only `N`= 2 is implemented.
- `maxeig(M)`: Maximum eigenvalue of a matrix. Defined as `max(real(eig(M)))`.
- `mineig(M)`: Minimum eigenvalue of a matrix. Defined as `max(real(eig(M)))`.
- `trace2(M)`: Trace-square of a matrix
- `trnorm(M)`: Trace-norm of a matrix
- `comm(A,B)`: Commutator, i.e., `comm`(A,B) = A*B-B*A
- `addnoise(rho/v,p)`: Adds white noise to a quantum state

- `binom(m,n)`: Binomial; defined as `factorial(n)/factorial(n-m)/factorial(m)`
- `qvec([N],[d])`: Empty state vector, filled with zeros, for `N` qudits of dimension `d`
- `qsize(rho/v,[d])`: Size of state vector or density matrix in qudits of dimension `d`
- `qeye([N],[d])`: Identity matrix for `N` qudits of dimension `d`

10 Commands for sparse matrices

Sparse matrices make it possible to store large matrices with many zero entries very efficiently. In QUBIT4MATLAB there are several commands that are realized both for full matrices and for sparse matrices. There are even commands that are only realized for sparse matrices. These are related to two-dimensional spin systems.

Next, the sparse commands of QUBIT4MATLAB are listed. For those that have a non-sparse version, only a short description is given.

- `spreordermat`: Sparse version of reordermat
- `spcoll`: Sparse version of coll
- `spinteract`: Sparse version of interact
- `spnnchain`: Sparse version of nnchain
- `spnnchainp`: Sparse version of nnchainp
- `spising`: Sparse version of ising
- `spisingp`: Sparse version of isningp
- `spquditop`: Sparse version of quditop
- `sptwoquditop`: Sparse version of twoquditop
- `splatticep(op1,op2,Nx,Ny)`: Gives a two-dimensional lattice Hamiltonian for nearest-neighbor interaction, periodic boundary condition, sparse version. `op1` and `op2` define the two-qudit interaction, `Nx` and `Ny` define the size of the two-dimensional lattice.
- `splattice(op1,op2,Nx,Ny)`: Gives a two-dimensional lattice Hamiltonian for nearest-neighbor interaction, aperiodic boundary condition, sparse version. `op1` and `op2` define the two-qudit interaction, `Nx` and `Ny` define the size of the two-dimensional lattice.
- `spising2Dp(B,Nx,Ny)`: Gives the two-dimensional ferromagnetic Ising Hamiltonian, periodic boundary condition, sparse version. Gives the two-dimensional Hamiltonian with an Ising interaction and an transverse field. `B` defines the strength of the external field. `Nx` and `Ny` define the size of the two-dimensional lattice.

11 Summary and outlook

The QUBIT4MATLAB 3.0, a program package for MATLAB was introduced. This package helps with the calculations in quantum information science and quantum optics. The basic object it handles is an array of qudits. All qudits of the array are supposed to have the same dimension. The program package has routines for reordering the qudits, tracing out some of the qudits, etc. It has several commands for helping to define easily Hamilton operators for spin chains. It has several commands related to entanglement detection, such as the partial transposition or the realignment of the density matrix. In future, it would be interesting to extend the routines to handle arrays of qudits of various dimensions. This should be done without making the notation much more complicated or making the routines much slower.

12 Acknowledgement

We thank J.J. García-Ripoll, O. Gühne and M.M. Wolf for fruitful discussions. This work was supported by the Spanish MEC (Ramon y Cajal Programme, Consolider-Ingenio 2010 project "QOIT"). We also thank the support of the National Research Fund of Hungary OTKA (Contract No. T049234) and the Hungarian Academy of Sciences (János Bolyai Programme).

References

[1] M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, Cambridge, 2000.

[2] The package can be downloaded from the MATLAB file exchange at http://www.mathworks.com/matlabcentral/fileexchange/

[3] D.J. Griffiths, Introduction to Quantum Mechanics, Prentice-Hall, New Jersey, 1995.

[4] D.M. Greenberger, M.A. Horne, A. Shimony, A. Zeilinger, Am. J. Phys. 58 (1990) 1131.

[5] H.J. Briegel, R. Raussendorf, Phys. Rev. Lett. 86 (2001) 910.

[6] M. Hein, J. Eisert, H.J. Briegel, Phys. Rev. A 69 (2004) 062311.

[7] R.H. Dicke, Phys. Rev. 93 (1954) 99.

[8] G. Tóth, J. Opt. Soc. Am. B 24 (2007) 275.

[9] J.A. Smolin, Phys. Rev. A 63 (2001) 032306.

[10] D. Gottesman, Phys. Rev. A 54 (1996) 1862.

[11] P. Horodecki, Phys. Lett. A 232 (1997) 333.

[12] C.H. Bennett, D.P. DiVincenzo, T. Mor, P.W. Shor, J.A. Smolin, B.M. Terhal, Phys. Rev. Lett. 82 (1999) 5385.

[13] J. Lawrence, Phys. Rev. A 70 (2004) 012302.

[14] S. Yu, N.-L. Liu, Phys. Rev. Lett. 95 (2005) 150504.

[15] R.F. Werner, Phys. Rev. A 40 (1989) 4277.

[16] W.K. Wootters, Phys. Rev. Lett. 80 (1998) 2245.

[17] M. Horodecki, P. Horodecki, R. Horodecki, Phys. Lett. A 223 (1996) 1.

[18] A. Peres, Phys. Rev. Lett. 77 (1996) 1413.

[19] O. Rudolph, quant-ph/0202121.

[20] K. Chen, L.-A. Wu, Quant. Inf. Comp. 3 (2003) 193.

[21] G. Vidal, R.F. Werner, Phys. Rev. A 65 (2002) 032314.

[22] For works on the numerical solution of the separability problem see Refs. [23,24,25,26].

[23] F.G.S.L. Brandão, R. O. Vianna, Phys. Rev. Lett. 93 (2004) 220503.

[24] F.G.S.L. Brandão, R. O. Vianna, Phys. Rev. A 70 (2004) 062309.

[25] J. Eisert, P. Hyllus, O. Gühne, M. Curty, Phys. Rev. A 70 (2004) 062317.

[26] A.C. Doherty, P.A. Parrilo, F.M. Spedalieri, Phys. Rev. A 71 (2005) 032333.

[27] A. Acín, D. Bruß, M. Lewenstein, A. Sanpera, Phys. Rev. Lett. 87 (2001) 040401.

[28] N. Kiesel, C. Schmid, G. Tóth, E. Solano, H. Weinfurter, Phys. Rev. Lett. 98 (2007) 063604.

[29] G. Tóth, C. Knapp, O. Gühne, H.J. Briegel, quant-ph/0703018.

[30] M. Bourennane, M. Eibl, C. Kurtsiefer, S. Gaertner, H. Weinfurter, O. Gühne, P. Hyllus, D. Bruß, M. Lewenstein, A. Sanpera, Phys. Rev. Lett. 92 (2004) 087902 .

[31] M.M. Wolf, private communication (2005).

[32] K. Życzkowski, H.-J. Sommers, J. Phys. A 34 (2001) 7111.

[33] G. Tóth, J.J. García-Ripoll, Phys. Rev. A 75 (2007) 042311.