QDENSITY/QCWAVE: A MATHEMATICA QUANTUM COMPUTER SIMULATION EXTENSION AND UPGRADE.

Frank Tabakin[a]

[a] *Department of Physics and Astronomy*
University of Pittsburgh, Pittsburgh, PA, 15260
tabakin@pitt.edu

Abstract

The Mathematica quantum computer simulation packages QDENSITY and QCWAVE are extensively extended and upgraded. The density matrix is featured in QDENSITY while in QCWAVE a quantum state vector approach is stressed. The present versions are provided in several associated packages; namely, QDensity, QCWave, BTSystem and Circuits . Tutorials are presented, some of which update earlier ones, plus several new ones that illustrate the capabilities of the packages.

This version includes improved treatment of tensor products of states and density matrices, based on new features that are included in Mathematica 9 - 10.3. A major extension to include qutrit (triplet), as well as qubit(binary) and hybrid qubit/qutrit systems is described in tutorials and in the associated BTSystem package. Many other new features are also illustrated in tutorial notebooks. Updated sample quantum computation algorithms and entanglement studies are presented, including Schmidt decomposition, entropy, mutual information, partial transposition, and calculation of the quantum discord. Examples of Bell's theorem are also included. These extensions and upgrades will hopefully be instructive and also aid in studies of QC dynamics, stability, and efficacy of error correction methods.

Program Summary

Title of programs: QDensity, QCWave, BTSystem, Circuits
Catalogue identifier:
Program summary URL: http://cpc.cs.qub.ac.uk/summaries
Program available from: CPC Program Library, Queen's University of Belfast, N. Ireland.
Operating systems: Any operating system that supports Mathematica.
Programming language used: Mathematica 9.0-10.3.0.
Number of bytes in distributed program, including test code and documentation: *55MB*
Distribution format: zip

Nature of Problem: Simulation of quantum algorithms, Qubit and Qutrit hybrid systems, entanglement criteria, (including Schmidt decomposition, entropy, mutual information, partial transposition, quantum discord) and Bell's theorem.
Method of Solution: A Mathematica package containing commands to create and analyze quantum circuits is upgraded and extended. Mathematica tutorials and notebooks illustrate the capabilities of the packages and demonstrate quantum computation applications.

Contents

List of Figures

1. INTRODUCTION

This paper is part of a series that provides a flexible simulation of a quantum computer. Here the Mathematica(MM) packages QDensity [1], QCWave [2] are greatly improved and extended. A Fortran-90 quantum computer (QC) simulation package called QCMPI has been published [3], which uses parallel processing and message passing interface(MPI) capabilities. Some features of that work have been included in our present MM renditions. [1]

In our earlier papers, we described qubit state vectors and associated amplitudes for one, two and multi-qubit states and methods for handling one, two and three-qubit operators or gates acting on state vectors. The basic idea of a density matrix and its formation have also been described earlier. For guidance with these ideas and a description of our notation and usage, we refer the reader to our earlier works. We have attempted to maintain consistent and clear notation to aid the user.

In the present paper, we first describe qubit, qutrit and hybrid (mixed qubit & qutrit) states in section 2. In section 3, the qubit gates are reviewed and extended to qutrit and hybrid networks. The methods used to evaluate entropy, mutual information, partial trace, quantum discord, partial transpose, and Bell's theorem are presented in section 4. Tutorials on the Bell inequalities and on the Schmidt decomposition for two qubit, two qutrit and qubit-qutrit states are also provided.

In section 5, additional upgrades, such as updated QC algorithms, a sample extension of teleportation to qutrits, examples for random states and for Werner and X-states are briefly mentioned.

Plans for future applications and enhancements are presented in section 6. A list of associated tutorial notebooks and worksheets is in the appendix.

2. Qubit and Qutrit States

In this section, we discuss qubit and qutrit states. A qubit is a doublet (2-level) system, for which we use a binary (B) designation; whereas, a qutrit is a triplet (3-level) system, for which we use a ternary or triplet label (T). A mixed system consisting of qubits and qutrits is called a BT system.[2]

After discussing the B, T, and BT states, we turn to one-body and two-body operators or gates, section 3. Sample Mathematica(MM) cases are provided in

[1] The full QCMPI methodology can be implemented for Mathematica once MPI, is incorporated into Mathematica, as accomplished in the commercial product Pouch (see: http://daugerresearch.com/pooch/mathematica.shtml).

[2] The BT system is also called a Mixed Radix(MR) system, which is for example a mixed binary and ternary counting scheme. Such MR counting schemes are quite common; one example, is days (24 hours), weeks(7 days), year(52 weeks). We provide Mixed Radix commands DtoMR and MRtoD. The command MixedRadix is now included in MM10.2.

figures showing how various concepts are implemented in the packages, and as an introduction to the extensive package tutorials.

2.1. Single Qubit and Qutrit 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 [4, 5, 6, 7]. A brief review was given in our earlier papers in this series [1, 2, 3]. Here we review aspects of one, two and multi-qubit states in preparation for the extension to qutrit cases.

Recall that a general one qubit state is a superposition of the two states associated with the 0 and 1 bits:

$$\mid \Psi_1\rangle = C_0^{(1)} \mid 0\rangle + C_1^{(1)} \mid 1\rangle, \tag{1}$$

where the basic kets $|0\rangle$ $|1\rangle$ and the adjoint bra states $\langle 0|$ $\langle 1|$ are an orthonormal(ON) basis, Their unit normalization $\langle 0 \mid 0\rangle = 1, \langle 1 \mid 1\rangle = 1$ is simply an assertion for example that a state known to be in the $\mid 0\rangle$ state has unit probability of being in that state. Orthogonality $\langle 0|1\rangle = \langle 1|0\rangle = 0$ simply asserts that the $\mid 0\rangle$ & $\mid 1\rangle$ states are distinct.

Using the above ON properties, we observe that $C_0^{(1)} \equiv \langle 0 \mid \Psi_1\rangle$ and $C_1^{(1)} \equiv \langle 1 \mid \Psi_1\rangle$ are complex probability amplitudes for finding the general qubit Ψ_1 in the state $\mid 0\rangle$ or $\mid 1\rangle$, respectively. The normalization of the state $\langle \Psi_1 \mid \Psi_1\rangle = 1$, yields $\mid C_0^{(1)} \mid^2 + \mid C_1^{(1)} \mid^2 = 1$. Note that the spatial aspects of the wave function are being suppressed; which corresponds to the particle being in a fixed region. The kets $\mid 0\rangle$ and $\mid 1\rangle$ can be represented as $\mid 0\rangle \rightarrow \left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right)$ and $\mid 1\rangle \rightarrow \left(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right)$. Hence a 2×1 matrix representation of this one-qubit state is:

$$\mid \Psi_1\rangle \rightarrow \begin{pmatrix} C_0^{(1)} \\ C_1^{(1)} \end{pmatrix}. \tag{2}$$

This description applies to any two-level quantum system that can be associated with $\mid 0\rangle$ and $\mid 1\rangle$.

2.2. Multi-Qubit and Qutrit states

For a quantum systems with three states (qutrits), the prior discussion can be generalized for single qutrit states:

$$\mid \Psi_{T1}\rangle = C_0^{(T1)} \mid 0\rangle + C_1^{(T1)} \mid 1\rangle + C_2^{(T1)} \mid 2\rangle, \tag{3}$$

where the ON qutrit basis states are three kets $\mid 0\rangle$, $\mid 1\rangle$, and $\mid 2\rangle$, which can be represented as $\mid 0\rangle \rightarrow \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ $\mid 1\rangle \rightarrow \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$. and $\mid 2\rangle \rightarrow \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$. Hence a 3×1 matrix representation of this one-qutrit state is:

$$\mid \Psi_{T1}\rangle \rightarrow \begin{pmatrix} C_0^{(T1)} \\ C_1^{(T1)} \\ C_2^{(T1)} \end{pmatrix}. \tag{4}$$

The qutrit normalization is $\mid C_0^{(T1)} \mid^2 + \mid C_1^{(T1)} \mid^2 + \mid C_2^{(T1)} \mid^2 = 1$. The three states are simply alternate labels for the angular moment eigenstates for a spin one system $\mid SM_S\rangle$, with the connection being $\mid 0\rangle \rightarrow \mid 1,1\rangle$, $\mid 1\rangle \rightarrow \mid 1,0\rangle$, and $\mid 2\rangle \rightarrow \mid 1,-1\rangle$.

This description applies to any three-level quantum system that can be associated with $\mid 0\rangle$, $\mid 1\rangle$, and $\mid 2\rangle$.

The MM display of single qubit and qutrit states are illustrated in Figs. 1 & 2. The MM commands are in bold and the results are presented below the asterisks line.

```
DForm[Ket[0]]
DForm[Ket[1]]
DForm[C0 Ket[0] + C1 Ket[1]]
DFormA[C0 Bra[0] + C1 Bra[1]]
(* ****************************** *)

 + (1) |0 >

 + (1) |1 >

 + (C0) |0 > + (C1) |1 >

 +  ( C0 )  < 0 | +  ( C1 )  < 1 |
```

Figure 1: Single qubit states. DForm and DFormA display qubit kets and bras.

```
DFormT[KetT[0]]
DFormT[KetT[1]]
DFormT[KetT[2]]
DFormT[C0 KetT[0] + C1 KetT[1] + C2 KetT[2]]
DFormTA[C0 BraT[0] + C1 BraT[1] + C2 BraT[2]]
(* ********************************** *)
 + (1) |0 >

 + (1) |1 >

 + (1) |2 >

 + (C0) |0 > + (C1) |1 > + (C2) |2 >

 + (C0) < 0 | + (C1) < 1 | + (C2) < 2 |
```

Figure 2: Single qutrit states. DFormT and DFormTA display qutrit kets and bras.

2.2.1. Two-Qubit states

For two qubits, we have a product state $\mid q_1\ q_2\rangle = \mid q_1\rangle \mid q_2\rangle$, where q_1, q_2 take on the values 0 and 1. This product is called a tensor product and is symbolized as

$$\mid q_1\ q_2\rangle = \mid q_1\rangle \otimes \mid q_2\rangle. \tag{5}$$

In QDENSITY, the kets $\mid 0\rangle, \mid 1\rangle$ are invoked by the commands **Ket**[0] and **Ket**[1], and the two-qubit product state by $\mid 00\rangle = AF[\mathbf{Ket}[0] \otimes \mathbf{Ket}[0]]$, where "AF" denotes array flatten [3].

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

$$\mid 00\rangle \rightarrow \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}; \mid 01\rangle \rightarrow \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}; \mid 10\rangle \rightarrow \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}; \mid 11\rangle \rightarrow \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{6}$$

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

$$\mid \Psi_2\rangle = C_0^{(2)} \mid 00\rangle + C_1^{(2)} \mid 01\rangle + C_2^{(2)} \mid 10\rangle + C_3^{(2)} \mid 11\rangle, \tag{7}$$

is:

$$\mid \Psi_2\rangle \rightarrow \begin{pmatrix} C_0^{(2)} \\ C_1^{(2)} \\ C_2^{(2)} \\ C_3^{(2)} \end{pmatrix}. \tag{8}$$

Again $C_0^{(2)} \equiv \langle 00 \mid \Psi_2\rangle, C_1^{(2)} \equiv \langle 01 \mid \Psi_2\rangle, C_2^{(2)} \equiv \langle 10 \mid \Psi_2\rangle$, and $C_3^{(2)} \equiv \langle 11 \mid \Psi_2\rangle$, are complex probability amplitudes for finding the two-qubit system in the states $\mid q_1\ q_2\rangle$. The normalization of the state $\langle \Psi_2 \mid \Psi_2\rangle = 1$, yields

$$\mid C_0^{(2)} \mid^2 + \mid C_1^{(2)} \mid^2 + \mid C_2^{(2)} \mid^2 + \mid C_3^{(2)} \mid^2 = 1. \tag{9}$$

Note that we label the amplitudes using the decimal equivalent of the bit product $q_1\ q_2$, so that for example a binary label on the amplitude $C_{10}^{(2)}$ is equivalent to the decimal label $C_2^{(2)}$.

2.2.2. Two-Qutrit states

For two qutrits, we have a product state $\mid q_1\ q_2\rangle = \mid q_1\rangle \mid q_2\rangle$,where q_1, q_2 take on the values $0, 1$, and 2. This product is called a tensor product and is also symbolized as $\mid q_1\ q_2\rangle = \mid q_1\rangle \otimes \mid q_2\rangle$. In QDENSITY, the kets $\mid 0\rangle, \mid 1\rangle, \mid 2\rangle$ are invoked by the commands KetT[0], KetT[1], and KetT[2], and the product state by $\mid 02\rangle = AF[\mathrm{KetT}[0] \otimes \mathrm{KetT}[2]]$, where "AF" denotes array flatten. The

[3]The command $\otimes$ is now included in MM as a tensor product; however, it needs to be corrected by the command AF to be in proper matrix form. **Warning:**There are two types of $\otimes$ in MM 9-10; we now use $\otimes$ defined as [*TensorProduct*]; not as [*CircleTimes*]. Best practice is to invoke the QDENSpalette14.

nine kets $\mid 00\rangle, \mid 01\rangle, \mid 02\rangle, \mid 10\rangle, \mid 11\rangle, \mid 12\rangle, \mid 20\rangle, \mid 21\rangle, \mid 22\rangle$ can be represented as 9×1 matrices

$$\mid 00\rangle \rightarrow \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} ; \mid 01\rangle \rightarrow \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \cdots\cdots \mid 22\rangle \rightarrow \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{10}$$

Hence, a Dirac ket representation of the general two-qutrit state is,

$$\begin{aligned} \mid \Psi_{T2}\rangle = \; & C_0^{(T2)} \mid 00\rangle + C_1^{(T2)} \mid 01\rangle + C_2^{(T2)} \mid 02\rangle + C_3^{(T2)} \mid 10\rangle + C_4^{(T2)} \mid 11\rangle \\ & + C_5^{(T2)} \mid 12\rangle + C_6^{(T2)} \mid 20\rangle + C_7^{(T2)} \mid 21\rangle + C_8^{(T2)} \mid 22\rangle, \end{aligned} \tag{11}$$

or as:

$$\mid \Psi_{T2}\rangle \rightarrow \begin{pmatrix} C_0^{(T2)} \\ C_1^{(T2)} \\ C_2^{(T2)} \\ C_3^{(T2)} \\ C_4^{(T2)} \\ C_5^{(T2)} \\ C_6^{(T2)} \\ C_7^{(T2)} \\ C_8^{(T2)} \end{pmatrix}, \tag{12}$$

where the nine complex coefficients $C_0^{(T2)} \cdots C_8^{(T2)}$ are subject to the normalization of the state $\langle \Psi_{T2} \mid \Psi_{T2} \rangle = 1$, which yields $\sum_{i=0}^{8} \mid C_i^{(T2)} \mid^2 = 1$. The nine complex coefficients: $C_0^{(T2)} \equiv \langle 00 \mid \Psi_{T2} \rangle$, $C_1^{(T2)} \equiv \langle 01 \mid \Psi_{T2} \rangle \cdots\cdots C_8^{(T2)} \equiv \langle 22 \mid \Psi_{T2} \rangle$, are complex probability amplitudes for finding the two-qutrit system in the states $\mid q_1 \; q_2 \rangle$. Note that we label the amplitudes using the ternary (base 3) equivalent of the qutrit product $q_1 \; q_2$, so that for example a ternary label on the amplitude $C_{21}^{(T2)}$ is equivalent to the decimal label $C_7^{(T2)}$. See notebook TernaryTutorial.nb.

Two-qubit and two-qutrit states are illustrated in Figs. 3 and 4 [4]. A more complicated multi-qubit state is illustrated in Fig. 5

[4] Note in MM qutrits appear as italic blue numerals.

```
AF[Ket[0]⊗Ket[0]]
DForm[%]
DForm[C0 AF[Ket[0]⊗Ket[0]] +
  C1 AF[Ket[0]⊗Ket[1]] + C2 AF[Ket[1]⊗Ket[0]] +
  C3 AF[Ket[1]⊗Ket[1]]]
DFormA[C0 AF[Bra[0]⊗Bra[0]] +
  C1 AF[Bra[0]⊗Bra[1]] + C2 AF[Bra[1]⊗Bra[0]] +
  C3 AF[Bra[1]⊗Bra[1]]]
(* ************************************ *)

⎛1⎞
⎜0⎟
⎜0⎟
⎝0⎠
 + (1) |00 >
 + (C0) |00 > + (C1) |01
  > + (C2) |10 > + (C3) |11 >
 + ( C0 ) < 00 | + ( C1 ) < 01
  | + ( C2 ) < 10 | + ( C3 ) < 11 |
```

Figure 3: Two qubit states. DForm and DFormA display two-qubit kets and bras.

```
DFormT[AF[KetT[0]⊗KetT[0]]]
DFormTA[AF[BraT[2]⊗BraT[0]]]
DFormT[C0 AF[KetT[0]⊗KetT[0]] +
  C1 AF[KetT[0]⊗KetT[1]] +
  C2 AF[KetT[0]⊗KetT[2]] + C3 AF[KetT[1]⊗KetT[0]] +
  C4 AF[KetT[1]⊗KetT[1]] + C5 AF[KetT[1]⊗KetT[2]] +
  C6 AF[KetT[2]⊗KetT[0]] +
  C7 AF[KetT[2]⊗KetT[1]] + C8 AF[KetT[2]⊗KetT[2]]]
(* ************************************ *)

 + (1) |00 >
 + (1) < 20 |
 + (C0) |00 > + (C1) |01 > + (C2) |02 >
  + (C3) |10 > + (C4) |11 > + (C5) |12 >
  + (C6) |20 > + (C7) |21 > + (C8) |22 >
```

Figure 4: Two qutrit states. DFormT and DFormTA display two-qutrit kets and bras.

```
KetV[{0, 1, 1}] == AF[Ket[0]⊗Ket[1]⊗Ket[1]]
DForm[KetV[{0, 1, 1, 0, 1}]]
DForm[a KetV[{0, 1, 1, 0, 1}] +
  b  KetV[{0, 0, 1, 0, 1}]]
(* ************************************ *)

True
 + (1)  |01101 >
 + (b)  |00101 > + (a)  |01101 >
```

Figure 5: Multi-qubit states using the KetV command.

2.2.3. Hybrid Qubit - Qutrit (BT) states

Hybrid qubit-qutrit states, which we denote as BT systems, consist of both qubits and qutrits [5]. That mixture is stipulated by an array **QA**, such as **QA= {B,T,T,B}**, which denotes a binary ⊗ triplet ⊗ triplet⊗ binary state. A numeric array **QD** corresponding to that **QA** array is **QD= {2,3,3,2}** . The dimension of the hybrid state vector is then $2^{nq} \times 3^{nt}$ where nq is the number of qubits and nt the number of qutrits (triplets) in **QA**. In Fig. 6, a qubit-qutrit **QA= {B,T}** state is displayed. A **QA= {B,B,T,B,T}** state is displayed in Fig. 7. In Fig. 8, a more complicated three-qubit,four qutrit hybrid state is shown along with the KetBT and DFormBT commands.

```
QA = {B, T};
DFormBT[QA, AF[Ket[0]⊗KetT[0]]]
(* ************************************ *)
 + (1)  |0₂0₃ >

C₀ AF[Ket[0]⊗KetT[0]] + C₁ AF[Ket[0]⊗KetT[1]]
  +C₂ AF[Ket[0]⊗KetT[2]] +
  C₃ AF[Ket[1]⊗KetT[0]] + C₄ AF[Ket[1]⊗KetT[1]]
  +C₅ AF[Ket[1]⊗KetT[2]];
DFormBT[QA, %]
(* ************************************ *)
 + (C₀)  |0₂0₃ > + (C₁)  |0₂1₃ > + (C₂)  |0₂2₃ > +
 (C₃)  |1₂0₃ > + (C₄)  |1₂1₃ > + (C₅)  |1₂2₃ >
```

Figure 6: A hybrid qubit-qutrit state. DFormBT (DFormBTA) display hybrid BT kets (bras). Subscripts indicate if the entry is a qubit or qutrit.

[5] Examples of how to produce BT systems are discussed in the article [10]

```
QA = {B, B, T, B, T};
AF[Ket[0]⊗Ket[1]⊗KetT[2]⊗Ket[0]⊗KetT[1]];
DFormBT[QA, %]
(* ************************************ *)
 + (1) |0₂1₂2₃0₂1₃ >
```

Figure 7: A hybrid **QA= {B,B,T,B,T}** state .

```
QA = {B, T, T, B, B, T, T}; Row[QA]
QD = QA /. {B → 2, T → 3}; Row[QD]
nq = Count[QA, B]
nt = Count[QA, T]
dim = 2^nq × 3^nt
(* ************************************ *)

BTTBBTT

2332233

3

4

648
```

A three−qubit− four qutrit hybrid state

```
KetVBT[nq + nt, QA, {0, 2, 1, 1, 1, 2, 2}];
DFormBT[QA, %]
(* ************************************ *)
 + (1) |0₂2₃1₃1₂1₂2₃2₃ >
```

Figure 8: A hybrid or Mixed Radix **QA= {B,T,T,B,B,T,T}** state.

3. Qubit and Qutrit Gates

In the previous section, general qubit and qutrit states were expressed as superpositions of associated basis states. These basis states, such as $|0>$ & $|1>$ for qubits, and $|0>, |1>$ & $|2>$ for qutrits, are used to construct multi-qubit, multi-qutrit, and hybrid states, which form a basis for the construction of general states for such systems. Now we turn to operators that act on such states and their associated operator basis. These operators are represented by $N \times N$ matrices where N is the dimension of the state vector. The general operator can be expressed as combinations of the $N \times N$ operator basis, as discussed in the following sections.

First, qubit (B) basis operators and gates are reviewed and then generalized to the qutrit(T) and hybrid (BT) cases.

3.1. Single-Qubit Basis

Operators or gates acting on a single qubit are represented by 2×2 matrices. The dimension of the state vector is $N = 2^{nq} = 2$, here $nq = 1$. The Pauli matrices provide an operator basis of all such matrices. The Pauli-spin matrices are: $\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$. These are all Hermitian matrices $\sigma_i = \sigma_i^\dagger$. We use the labels $(1, 2, 3)$ to denote the directions (x, y, z). A fourth Pauli matrix is simply the unit matrix: $\sigma_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Any 2×2 matrix can be constructed from these four Pauli matrix. , which therefore are an operator basis. Consider the general combination $a_0\sigma_0 + a_1\sigma_1 + a_2\sigma_2 + a_3\sigma_3 = a_0\sigma_0 + \vec{a} \cdot \vec{\sigma} = \begin{pmatrix} a_0 + a_3 & a_1 - ia_2 \\ a_1 + ia_2 & a_0 - a_3 \end{pmatrix}$.

The Pauli operators are equivalent to the following Ket $\otimes$ Bra tensor products:

$$\begin{array}{l} \sigma_0 = \mid 0\rangle \otimes \langle 0 \mid + \mid 1\rangle \otimes \langle 1 \mid \\ \sigma_3 = \mid 0\rangle \otimes \langle 0 \mid - \mid 1\rangle \otimes \langle 1 \mid \end{array} \quad \& \quad \begin{array}{l} \sigma_1 = +1 \mid 0\rangle \otimes \langle 1 \mid +1 \mid 1\rangle \otimes \langle 0 \mid \\ \sigma_2 = -i \mid 0\rangle \otimes \langle 1 \mid +i \mid 1\rangle \otimes \langle 0 \mid \end{array}. \quad (13)$$

Note Ket $\otimes$ Bra tensor products can also be used to construct projection operators:

$$\begin{array}{l} \mathcal{P}_0 = \mid 0\rangle \otimes \langle 0 \mid = \dfrac{\sigma_0 + \sigma_3}{2} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \\ \mathcal{P}_1 = \mid 1\rangle \otimes \langle 1 \mid = \dfrac{\sigma_0 - \sigma_3}{2} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}. \end{array} \quad (14)$$

These projection operators project out the associated Ket part by virtue of the properties $\mathcal{P}_0 \mid 0\rangle = \mid 0\rangle, \mathcal{P}_0 \mid 1\rangle = 0$ and $\mathcal{P}_1 \mid 0\rangle = 0, \mathcal{P}_1 \mid 1\rangle = \mid 1\rangle$.

Two important single qubit operators (or gates) can now be identified. One is the **NOT** gate, which is simply $\mathbf{NOT} = \sigma_1$. It has the property (see Eq. 13)

```
σ0
σ1
σ2
σ3

P0
P1
P0 == (σ0 + σ3) / 2 && P1 == (σ0 - σ3) / 2
P0.Ket[0] == Ket[0] && P1.Ket[1] == Ket[1]
Flatten[P0.Ket[1]] == ZeroT[2] && Flatten[P1.Ket[0]] == ZeroT[2]
(* ************************************ *)
```

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

```
True

True

True
```

Figure 9: Qubit: Pauli and projection operators.

NOT $\mid 0\rangle = \mid 1\rangle$ and **NOT** $\mid 1\rangle = \mid 0\rangle$. The second single qubit operator is the Hadamard

$$\mathcal{H} = \frac{\sigma_1 + \sigma_3}{\sqrt{2}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{15}$$

which has the property $\mathcal{H} \mid 0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \mathcal{H} \mid 1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$.

The above steps for qubits, illustrated in Figs. 9-10, will be generalized to qutrits later.

```
ℋ
DForm[ℋ.Ket[0]]
DForm[ℋ.Ket[1]]
ℋ == (σ1 + σ3) / Sqrt[2]
(* ************************************ *)
```

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

$$+ \left(\frac{1}{\sqrt{2}}\right) \mid 0 > + \left(\frac{1}{\sqrt{2}}\right) \mid 1 >$$

$$+ \left(\frac{1}{\sqrt{2}}\right) \mid 0 > + \left(-\frac{1}{\sqrt{2}}\right) \mid 1 >$$

```
True
```

Figure 10: Qubit: Hadamard operator.

3.2. Multi-Qubit Operator Basis

Consider a multi-qubit operator for nq=2. The most important one is the CNOT gate, which is defined by

$$\mathbf{CNOT} = \mathcal{P}_0 \otimes \mathcal{I} + \mathcal{P}_1 \otimes \sigma_1 \;=\; \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{16}$$

where $\mathcal{I} = \sigma_0$ is the 2×2 identity matrix. The matrix above is for the case that qubit 1 is the control and the **NOT** gate acts on qubit 2 only when the control qubit 1 has the value 1. This CNOT gate produces the changes: $\mid 00 > \rightarrow \mid 00 >$, $\mid 01 > \rightarrow \mid 01 >, \mid 10 > \rightarrow \mid 11 >$ & $\mid 11 > \rightarrow \mid 10 >$.

```
CNOT[2, 1, 2]
CNOT[2, 1, 2] == AF[P0⊗σ0] +AF[P1⊗σ1]
Toffoli[3, 1, 2, 3]
Toffoli[3, 1, 2, 3] ==
 AF[(P0⊗P0 + P0⊗P1 + P1⊗P0)⊗σ0] + AF[P1⊗P1⊗σ1]
(* ********************************** *)
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

```
True
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

```
True
```

Figure 11: Qubit: CNOT and Toffoli gates.

For nq=3 the most important gate is the Toffoli gate, which is defined by

$$\mathbf{Toffoli} = (\mathcal{P}_0\otimes\mathcal{P}_0+\mathcal{P}_0\otimes\mathcal{P}_1+\mathcal{P}_1\otimes\mathcal{P}_0)\otimes\mathcal{I}+\mathcal{P}_1\otimes\mathcal{P}_1\otimes\sigma_1 \;=\; \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} . \tag{17}$$

This is for the case that qubits 1 & 2 are the control qubits and the **NOT** gate acts on qubit 3 only when the control qubits both have the value 1.

This qubit gate is generated as shown in Fig. 11; it will be generalized to qutrits later.

3.3. Single-Qutrit Basis and Gates

The operator basis for qutrits consists of nine matrices; there are several possible choices for the 9 operators. One choice uses the unit matrix plus the three spin-one spin matrices, along with a rank 2 Cartesian tensor. Another

is the Gell-mann basis. The third is the generalized Pauli basis. They are displayed in Figs. 12- 14. The spin-one and Gell-Mann sets are hermitian ; whereas , the generalized Pauli matrices are not hermitian [6] Spin observables are defined using either the spin or Gell-mann basis.

```
Row[Table[MatrixForm[w[i]], {i, 0, 8}], "      "]
(* ******************************** *)
"Spin−One Basis for Qutrits"
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 0 & \frac{\sqrt{3}}{2} & 0 \\ \frac{\sqrt{3}}{2} & 0 & \frac{\sqrt{3}}{2} \\ 0 & \frac{\sqrt{3}}{2} & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & -\frac{i\sqrt{3}}{2} & 0 \\ \frac{i\sqrt{3}}{2} & 0 & -\frac{i\sqrt{3}}{2} \\ 0 & \frac{i\sqrt{3}}{2} & 0 \end{pmatrix}$$

$$\begin{pmatrix} \sqrt{\frac{3}{2}} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -\sqrt{\frac{3}{2}} \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & -i\sqrt{\frac{3}{2}} \\ 0 & 0 & 0 \\ i\sqrt{\frac{3}{2}} & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & \frac{\sqrt{3}}{2} & 0 \\ \frac{\sqrt{3}}{2} & 0 & -\frac{\sqrt{3}}{2} \\ 0 & -\frac{\sqrt{3}}{2} & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & -\frac{i\sqrt{3}}{2} & 0 \\ \frac{i\sqrt{3}}{2} & 0 & \frac{i\sqrt{3}}{2} \\ 0 & -\frac{i\sqrt{3}}{2} & 0 \end{pmatrix} \quad \begin{pmatrix} -\frac{1}{2} & 0 & \frac{\sqrt{3}}{2} \\ 0 & 1 & 0 \\ \frac{\sqrt{3}}{2} & 0 & -\frac{1}{2} \end{pmatrix} \quad \begin{pmatrix} -\frac{1}{2} & 0 & -\frac{\sqrt{3}}{2} \\ 0 & 1 & 0 \\ -\frac{\sqrt{3}}{2} & 0 & -\frac{1}{2} \end{pmatrix}$$

Figure 12: Qutrit Basis—Spin-one case

[6] see BTtutorial2014 for an exploration of the properties of the three qutrit bases.

```
Row[Table[MatrixForm[wG[i]], {i, 0, 8}], "          "]
(* ********************************* *)
"Gell–Mann Basis for Qutrits"
```

$$\begin{pmatrix} \sqrt{\frac{2}{3}} & 0 & 0 \\ 0 & \sqrt{\frac{2}{3}} & 0 \\ 0 & 0 & \sqrt{\frac{2}{3}} \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & -i & 0 \\ i & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & -i \\ 0 & 0 & 0 \\ i & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -i \\ 0 & i & 0 \end{pmatrix} \quad \begin{pmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & -\frac{2}{\sqrt{3}} \end{pmatrix}$$

Figure 13: Qutrit Basis—Gell-Mann case

```
Row[Table[MatrixForm[wP[i]], {i, 0, 8}], "          "]
(* ********************************* *)
"Pauli Basis for Qutrits"
```

$$\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}\quad\begin{pmatrix}0&0&1\\1&0&0\\0&1&0\end{pmatrix}\quad\begin{pmatrix}0&0&1\\e^{\frac{2i\pi}{3}}&0&0\\0&e^{-\frac{2i\pi}{3}}&0\end{pmatrix}$$

$$\begin{pmatrix}1&0&0\\0&e^{\frac{2i\pi}{3}}&0\\0&0&e^{-\frac{2i\pi}{3}}\end{pmatrix}\quad\begin{pmatrix}0&1&0\\0&0&1\\1&0&0\end{pmatrix}\quad\begin{pmatrix}0&1&0\\0&0&e^{\frac{2i\pi}{3}}\\e^{-\frac{2i\pi}{3}}&0&0\end{pmatrix}$$

$$\begin{pmatrix}1&0&0\\0&e^{-\frac{2i\pi}{3}}&0\\0&0&e^{\frac{2i\pi}{3}}\end{pmatrix}\quad\begin{pmatrix}0&0&1\\e^{-\frac{2i\pi}{3}}&0&0\\0&e^{\frac{2i\pi}{3}}&0\end{pmatrix}\quad\begin{pmatrix}0&1&0\\0&0&e^{-\frac{2i\pi}{3}}\\e^{\frac{2i\pi}{3}}&0&0\end{pmatrix}$$

Figure 14: Qutrit Basis—Generalized Pauli

Qutrit gates are often defined using the Pauli basis. For example, one defines a qutrit NOT gate as the generalized $\mathbf{X}$ operator, which is given in Fig. 15, where we see that $\mathbf{X} \mid i\rangle = \mid \mathrm{Mod}[i+1,3]\rangle$. A qutrit Hadamard acts on a single qutrit as shown in Fig. 16. The qutrit Hadamard generates three orthonormal linear combinations of three basic qutrit kets, incorporating a phase factor $\xi(k) = \exp^{-\frac{2\pi i k}{3}}$.

```
"X gate for Qutrits"
X = wP[1]; X
```

$$\begin{pmatrix}0&0&1\\1&0&0\\0&1&0\end{pmatrix}$$

```
(* ******X gate acting on qutrit kets*************** *)
DFormBT[{T}, X.KetT[0]]
DFormBT[{T}, X.KetT[1]]
DFormBT[{T}, X.KetT[2]]
```

$+ (1) \mid 1_3 >$

$+ (1) \mid 2_3 >$

$+ (1) \mid 0_3 >$

Figure 15: Qutrit Basis—NOT gate

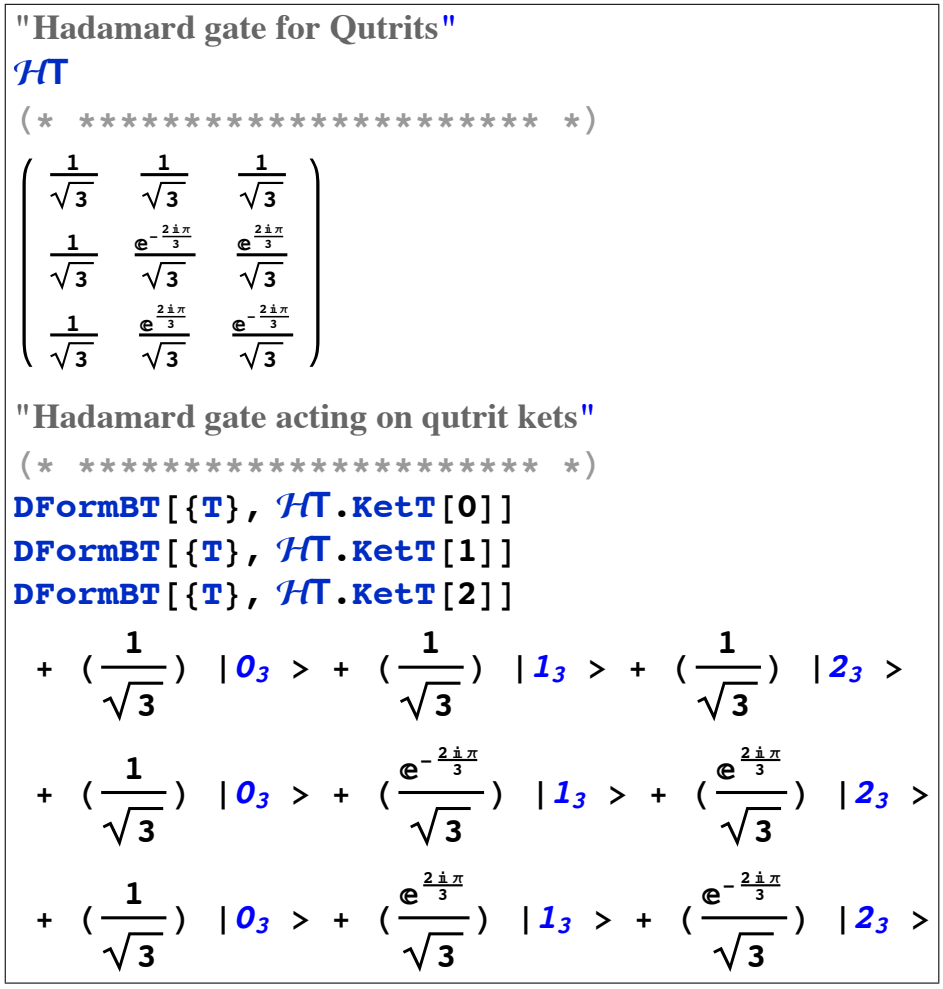


Figure 16: Qutrit Basis—the qutrit Hadamard $\mathcal{HT}$.

3.4. Multi-Qutrit Basis and Gates

In the previous section, the single qutrit states and gates were presented. Now we extend that discussion to two or more qutrits. Later, we will discuss hybrid cases wherein a mixture of qubits and qutrits are stipulated. The tutorial GatesQudits2015 explores these topics in great detail.

To generate an operator basis for two qutrits, we form a tensor product w(i)⊗ w(i), as shown in Fig. 17. An example of a two-qutrit operator is two Hadamards acting on both qutrits $\mathcal{H}_\mathcal{T} \otimes \mathcal{H}_\mathcal{T}$, as shown in Fig. 18.

"Multi-qutrit operators"

SPT[2, {6, 8}] == AF[w[6] ⊗ w[8]]
SPT[4, {1, 2, 6, 8}] == AF[w[1] ⊗w[2] ⊗w[6] ⊗ w[8]]

True

True

AF[w[6] ⊗ w[8]]

$$\begin{pmatrix} 0 & 0 & 0 & \frac{i\sqrt{3}}{4} & 0 & \frac{3i}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{i\sqrt{3}}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{3i}{4} & 0 & \frac{i\sqrt{3}}{4} & 0 & 0 & 0 \\ -\frac{i\sqrt{3}}{4} & 0 & -\frac{3i}{4} & 0 & 0 & 0 & -\frac{i\sqrt{3}}{4} & 0 & -\frac{3i}{4} \\ 0 & \frac{i\sqrt{3}}{2} & 0 & 0 & 0 & 0 & 0 & \frac{i\sqrt{3}}{2} & 0 \\ -\frac{3i}{4} & 0 & -\frac{i\sqrt{3}}{4} & 0 & 0 & 0 & -\frac{3i}{4} & 0 & -\frac{i\sqrt{3}}{4} \\ 0 & 0 & 0 & \frac{i\sqrt{3}}{4} & 0 & \frac{3i}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{i\sqrt{3}}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{3i}{4} & 0 & \frac{i\sqrt{3}}{4} & 0 & 0 & 0 \end{pmatrix}$$

Figure 17: Multi-qutrit basis and the SPT command

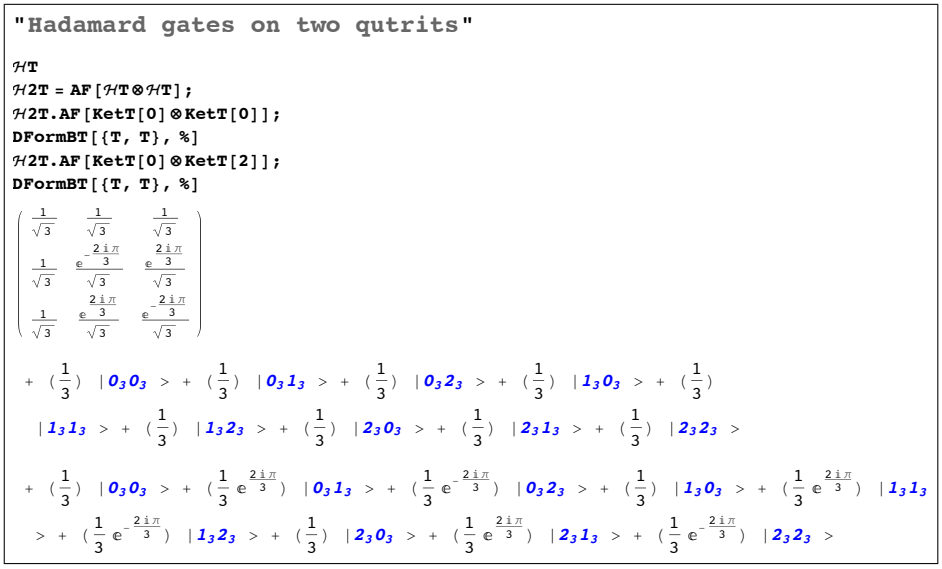


Figure 18: Two qutrit Hadamards on two qutrits

Another two-qutrit operator can be defined using the same procedure discussed earlier for the CNOT two-qubit gate. First we need the qutrit projection operators:

$$\begin{aligned}
\mathcal{P}_0^T &= |\,0\rangle \otimes \langle 0\,| = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
\mathcal{P}_1^T &= |\,1\rangle \otimes \langle 1\,| = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
\mathcal{P}_2^T &= |\,2\rangle \otimes \langle 2\,| = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.
\end{aligned} \tag{18}$$

These projection operators project out the associated qutrit kets. The sum of these three projection operators equals a 3×3 unit matrix $\mathcal{I}_3$.

Now, we can define a controlled-not gate for qutrits in a variety of ways. The first uses qutrit 1 as the control, with the qutrit Pauli **NOT** gate $\mathbf{X}$=wP[1] acting on qutrit 2 as follows:

$$\mathcal{C}NOT1 = \mathcal{P}_0^T \otimes \mathcal{I}_3 + \mathcal{P}_1^T \otimes \mathbf{X} + \mathcal{P}_2^T \otimes \mathbf{X} \cdot \mathbf{X}. \tag{19}$$

Since $\mathbf{X}$ is not Hermitian $\mathcal{C}NOT1^\dagger \neq \mathcal{C}NOT1$, in contrast to the qubit case where σ_1 is Hermitian.

The second uses qutrit 2 as the control, with the qutrit NOT gate $\mathbf{X}$ acting on qutrit 1.

$$\mathcal{C}NOT2 = \mathcal{I}_3 \otimes \mathcal{P}_0^T + \mathbf{X} \otimes \mathcal{P}_1^T + \mathbf{X} \cdot \mathbf{X} \otimes \mathcal{P}_2^T. \tag{20}$$

The explicit form for $\mathcal{C}NOT1$ is

$$CNOT1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix},$$

and for $\mathcal{C}NOT2$

$$CNOT2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Control-Z, and Control-Y Gates can also be constructed using the qutrit Pauli $\mathbf{Y}$=wP[2] and & $\mathbf{Z}$=wP[3] gates

$$\begin{aligned} \mathcal{C}ZT1 &= \mathcal{P}_0^T \otimes \mathcal{I}_3 + \mathcal{P}_1^T \otimes \mathbf{Z} + \mathcal{P}_2^T \otimes \mathbf{Z} \cdot \mathbf{Z} \\ \mathcal{C}ZT2 &= \mathcal{I}_3 \otimes \mathcal{P}_0^T + \mathbf{Z} \otimes \mathcal{P}_1^T + \mathbf{Z} \cdot \mathbf{Z} \otimes \mathcal{P}_2^T \\ \mathcal{C}YT1 &= \mathcal{P}_0^T \otimes \mathcal{I}_3 + \mathcal{P}_1^T \otimes \mathbf{Y} + \mathcal{P}_2^T \otimes \mathbf{Y} \cdot \mathbf{Y} \\ \mathcal{C}YT2 &= \mathcal{I}_3 \otimes \mathcal{P}_0^T + \mathbf{Y} \otimes \mathcal{P}_1^T + \mathbf{Y} \cdot Y \otimes \mathcal{P}_2^T. \end{aligned} \tag{21}$$

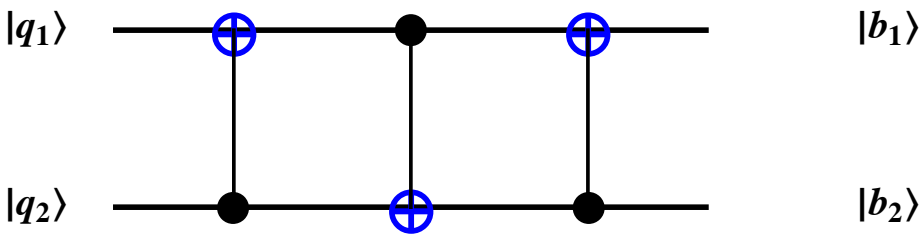
The above CNOT1 & CNOT2 gates do not provide a swap gate following the qubit pattern. To produce a qutrit swap gate another pair of CNOT gates are introduced. [8, 9], as shown in Fig. 19- 21. This modified swap also works when two qutrit operators are used within a multi-qutrit system, see later.

$$\begin{aligned} \mathcal{C}NOTH1 &= (\mathcal{I}_3 \otimes \mathcal{HT}^\dagger) \cdot \mathcal{C}ZT1 \cdot (\mathcal{I}_3 \otimes \mathcal{HT}^\dagger) \\ \mathcal{C}NOTH2 &= (\mathcal{HT}^\dagger \otimes \mathcal{I}_3) \cdot \mathcal{C}ZT2 \cdot (\mathcal{HT}^\dagger \otimes \mathcal{I}_3). \end{aligned} \tag{22}$$

Here $\mathcal{HT}$ is the qutrit Hadamard.

Two Qubit swap

```
SWAPC = Circuit[ (CX  C  CX
                  C  CX  C) , {}]
ψ = Sum[ca[i] Ket[i], {i, 0, 1}];
ϕ = Sum[cb[i] Ket[i], {i, 0, 1}];
psiI = AF[ψ⊗ϕ];
Flatten[AF[Swap[2, 1, 2].psiI] - AF[ϕ⊗ψ]] == ZeroT[4]
Swap[2, 1, 2] == CNOT[2, 2, 1].CNOT[2, 1, 2].CNOT[2, 2, 1]
```




```
True

True
```

Figure 19: Qubit swap gates

```
ψ = Sum[ca[i] KetT[i], {i, 0, 2}];
ϕ = Sum[cb[i] KetT[i], {i, 0, 2}];
psiI = AF[ψ⊗ϕ];
AF[CNOT1.CNOT2.CNOT1.psiI] - AF[ϕ⊗ψ] == ZeroT[8]
AF[CNOT2.CNOT1.CNOT2.psiI] - AF[ϕ⊗ψ] == ZeroT[8]
(*  Swap based on CNOT1 and CNOT2 fails *)
False

False
```

Figure 20: Qutrit swap gates based on CNOT1 & CNOT2 failure.

```
ψ = Sum[ca[i] KetT[i], {i, 0, 2}];
DFormT[ψ]
ϕ = Sum[cb[i] KetT[i], {i, 0, 2}];
DFormT[ϕ]
psiI = AF[ψ⊗ϕ];
psiF = AF[ϕ⊗ψ];
AF[CNOTH1.CNOTH2.CNOTH1.psiI] == psiF
AF[CNOTH2.CNOTH1.CNOTH2.psiI] == psiF
(* ************************************ *)
 + (ca[0]) |0 > + (ca[1]) |1 > + (ca[2]) |2 >
 + (cb[0]) |0 > + (cb[1]) |1 > + (cb[2]) |2 >
True
True
```

Figure 21: Qutrit swap gates based on CNOTH success

3.5. Qubit-Qutrit Operators

For a hybrid system consisting of one qubit and one qutrit, the single qubit and single qutrit operators are as defined earlier. The two-body BT or TB control operators can be defined in various ways. For example, CNOTBT1 with qubit 1 as control can be stipulated as [7]

$$\mathcal{C}NOTBT1 = \mathcal{P}_0 \otimes \mathcal{I}_3 + \mathcal{P}_1 \otimes X,$$

whereas, for CNOTTB1 with qutrit 1 as control and qubit 2 acted on by σ_1 can be defined in several ways. One way is

$$\mathcal{C}NOTTB1 = \mathcal{P}_0^T \otimes \mathcal{I}_2 + \mathcal{P}_1^T \otimes \sigma_1 + \mathcal{P}_2^T \otimes \mathcal{I}_2,$$

Another way is

$$\mathcal{C}NOTTB1 = \mathcal{P}_0^T \otimes \mathcal{I}_2 + \mathcal{P}_1^T \otimes \sigma_1 + \mathcal{P}_2^T \otimes \sigma_1.$$

Based on these examples, the user can explore other such qubit-qutrit operators.

[7] $\mathcal{I}_3$ denotes a unit 3×3 matrix and $\mathcal{I}_2$ denotes a unit 2×2 matrix.

3.6. General Operators

3.6.1. General Qubit Operators

In the pure multi-qubit case, the general form of multi-qubit operators are provided by several QDensity commands. For example, **had[n, i]** is an operator for n qubits with a single Hadamard acting on qubit "i;" **Had[n,Q]** is an n-qubit operator with Hadamards acting on all members of the set Q={q1,q2,... } of length n, where if qi is set to 1 that qubit is acted on by a Hadamard, whereas a value of 0 specifies no action on that qubit. For example,**Had[3,{1,0,1}]** has Hadamards acting on qubits 1 and 3 for a three qubit system. To get a Hadamard acting on all qubits, include all qubits in Q, e.g., use Q=1,1,1,..... **HALL[n]=Had[n,1,1,1....]** is also provided where the array Q of 1's has length n to include all the qubits. See Fig. 22.

A similar setup is included in QDensity for projection operators; **proj[n,i,a]** is an n-qubit operator, which yields a projection operator $\mathcal{P}_0$ for a=0 or $\mathcal{P}_1$ for a=1 acting on qubit "i." Also, an n-qubit projection operator **PJ[n,Q,A]** is defined which acts on set q1,q2,... of the n qubits along with an array **A ={a1....}** which stipulates if the projection on the qubit is either zero or one; if ai is not equal to zero or one then the unit operator $\mathcal{I}$=s[0] is taken for the ith qubit . Thus, one can build an operator of the type: $\mathcal{I}\otimes\mathcal{P}_0\otimes\mathcal{P}_1\otimes\mathcal{I}\otimes\mathcal{P}_0\otimes\mathcal{P}_1\otimes\mathcal{I}$, which asks if the qubits 2,3,5,6 have the values 0101 .

3.6.2. General Qutrit Operators

Similar commands are now available for pure qutrit systems Fig. 23. . The command **SPT[n,Q]** for n qutrits produces a spin basis tensor product with components stipulated by the array Q; for example, **SPT[3,{2,1,2}]** yields the spin-basis tensor product w[2]$\otimes$ w[1] $\otimes$ w[2]. To extend this procedure to qutrit Hadamards and projection operators, we set the w[$i > 8$] matrices as follows: w[10] =$\mathcal{HT}$, w[11] =**X** , w[12+i] =$\mathcal{PT}_i$, w[16] =**X.X** , w[33] =**Z** , w[34] =**Z.Z** . Here **X** and **Z** are the Pauli qutrit operators. Using these settings we see in Fig. 23 that **SPT[n,Q]** for n=3, Q={10,11,12} yields $\mathcal{HT}\otimes\mathbf{X}\otimes\mathcal{PT}_0$. A more general command **SPBTA[n,QA,Q]** will be discussed next.

3.6.3. General BT Hybrid Operators

For hybrid (mixed radix, BT) systems we have a mixture of qubits and qutrits as stipulated by the array QA. Examples of general operators for such BT systems are presented in Figs. 23- 24. For example, we see that a system with QA={ B,T,B,T } (i.e. a $qubit \times qutrit \times qubit \times qutrit$ system) with a NOT operator on the qubits 1 and 3, a Hadamard on the qutrit 3 and a projection operator on the qutrit 4 is generated by the command **SPBTA[4,QA,Q]** , with Q={ 1,10,,11,12}. Here s[11] is equal to s[1]=NOT. Special cases of pure B or pure T systems can be invoked and seen to be equivalent to earlier SP and SPT forms.

As a further generalization of earlier commands for general operators, see the examples in Fig. 25. There we display commands HadBT, hadBT, OneOpBT

and TwoOpBT. These are used in the BTSystems tutorial, along with ThreeOpBT, to construct generalized BT operators such as control-not, swap, control-z, TofolliBT, etc gates.

Another method to generate general operators for BT systems, is to apply the one and two body operators OpBT1, OpBT2 or OpBT3 to the full set of basis vectors, as is included in the commands BTOp1 and BTOp2 (see the tutorials).

With these tools any general BT operation can be constructed and used to study such systems. Cases of random B,T and BT states and density matrices are also incorporated into the package as illustrated in the associated tutorials.

```
DForm[Had[2, {1, 1}].AF[Ket[0]⊗Ket[0]]]
(*  ********************************************* *)

 + (1/2) |00 > + (1/2) |01 > + (1/2) |10 > + (1/2) |11 >

DForm[Had[3, {1, 1, 1}].AF[Ket[0]⊗Ket[0]⊗Ket[0]]]
Had[3, {1, 1, 1}] == AF[ℋ⊗ℋ⊗ℋ] &&
 Had[4, {1, 1, 1, 1}] == AF[ℋ⊗ℋ⊗ℋ⊗ℋ]

(*  ********************************************** *)
 + (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 >

True
```

Figure 22: General qubit operators, Hadamard example.

```
𝓗𝕋
```

$$\begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & \frac{e^{-\frac{2 i \pi}{3}}}{\sqrt{3}} & \frac{e^{\frac{2 i \pi}{3}}}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & \frac{e^{\frac{2 i \pi}{3}}}{\sqrt{3}} & \frac{e^{-\frac{2 i \pi}{3}}}{\sqrt{3}} \end{pmatrix}$$

```
SPT[3, {2, 1, 2}] == AF[w[2]⊗w[1]⊗w[2]] &&
 SPT[3, {10, 11, 12}] == AF[𝓗𝕋⊗𝕏⊗𝒫T₀] ==
  SPBTA[3, {T, T, T}, {10, 11, 12}]
```

```
True
```

```
w[10] == 𝓗T && w[11] == 𝕏 && w[12] == 𝒫T₀ && w[13] == 𝒫T₁ && w[14] == 𝒫T₂
wP[1] == 𝕏 && wP[3] == ℤ && w[16] == 𝕏.𝕏 && w[33] == ℤ && w[34] == ℤ.ℤ
```

```
True
```

```
True
```

```
SPBTA[3, {T, T, T}, {10, 11, 12}] == AF[𝓗𝕋⊗𝕏⊗𝒫T₀] ==
 SPT[3, {10, 11, 12}]
SPBTA[5, ConstantArray[T, 5], ConstantArray[10, 5]] ==
 AF[𝓗𝕋⊗𝓗𝕋⊗𝓗𝕋⊗𝓗𝕋⊗𝓗𝕋]
SPBTA[5, ConstantArray[T, 5], ConstantArray[13, 5]] ==
 AF[𝒫T₁⊗𝒫T₁⊗𝒫T₁⊗𝒫T₁⊗𝒫T₁]
```

```
True
```

```
True
```

```
True
```

Figure 23: General qutrit operator examples

```
SPBTA[2, {B, B}, {0, 10}] == AF[s[0]⊗ℋ] &&
 SPBTA[2, {B, B}, {10, 10}] == AF[ℋ⊗ℋ] &&
 SPBTA[2, {T, B}, {10, 10}] == AF[ℋT⊗ℋ]
```

True

```
SPBTA[3, {T, B, T}, {2, 1, 2}] == AF[w[2]⊗s[1]⊗w[2]] &&
 SPBTA[3, {T, T, B}, {10, 11, 12}] == AF[ℋT⊗𝕏⊗𝒫0] &&
 SPBTA[2, {B, B}, {0, 10}] == AF[s[0]⊗ℋ] &&
 SPBTA[2, {B, B}, {10, 10}] == AF[ℋ⊗ℋ] &&
 SPBTA[4, {B, T, B, T}, {1, 10, 11, 12}] == AF[s[1]⊗ℋT⊗s[1]⊗𝒫T0]
```

True

```
sB[10] == ℋ == s[10] && sB[12] == 𝒫0 == s[12] && sB[11] == s[11] == s[1]
```

True

```
SPBTA[5, {B, B, T, T, B}, ConstantArray[10, 5]] ==
  AF[ℋ⊗ℋ⊗ℋT⊗ℋT⊗ℋ] &&
 SPBTA[5, {B, B, T, T, B}, ConstantArray[13, 5]] == AF[𝒫1⊗𝒫1⊗𝒫T1⊗𝒫T1⊗𝒫1]
```

True

Figure 24: General hybrid BT operator examples

```
HadBT[2, {B, B}, {10, 10}] == AF[ℋ⊗ℋ] &&
  HadBT[3, {B, T, B}, {10, 0, 10}] == AF[ℋ⊗w[0]⊗ℋ] == SPBTA[3, {B, T, B}, {10, 0, 10}] &&
  HadBT[3, {B, T, T}, {10, 0, 10}] == AF[ℋ⊗w[0]⊗ℋT] == SPBTA[3, {B, T, T}, {10, 0, 10}]
q = 2; QA = {B, T, T}; L = Length[QA];
hadBT[L, QA, q] == AF[s[0]⊗ℋT⊗w[0]]
```

True

True

```
OneOpBT[4, {T, T, T, T}, 3, 1] == AF[w[0]⊗w[0]⊗w[1]⊗w[0]] &&
 OneOpBT[3, {B, B, B}, 3, 1] == AF[s[0]⊗s[0]⊗s[1]] &&
 OneOpBT[3, {B, B, T}, 3, 1] == AF[s[0]⊗s[0]⊗w[1]] &&
 OneOpBT[4, {B, T, T, B}, 3, 1] == AF[s[0]⊗w[0]⊗w[1]⊗s[0]] &&
 OneOpBT[4, {B, T, T, B}, 3, 10] == AF[s[0]⊗w[0]⊗wP[10]⊗s[0]] ==
  AF[s[0]⊗w[0]⊗ℋT⊗s[0]] &&
 OneOpBT[4, {B, T, T, B}, 4, 10] == AF[s[0]⊗w[0]⊗w[0]⊗s[10]] ==
  AF[s[0]⊗w[0]⊗w[0]⊗ℋ]
```

True

```
TwoOpBT[2, {T, T}, 1, 2, 1, 2] == AF[w[1]⊗w[2]] &&
 TwoOpBT[2, {T, T}, 1, 2, 1, 3] == AF[w[1]⊗w[3]] &&
 TwoOpBT[2, {T, T}, 1, 2, 2, 1] == AF[w[2]⊗w[1]] &&
 TwoOpBT[3, {T, T, B}, 1, 2, 1, 2] == AF[w[1]⊗w[2]⊗s[0]] &&
 TwoOpBT[4, {B, T, T, B}, 2, 3, 1, 2] == AF[s[0]⊗w[1]⊗w[2]⊗s[0]] &&
 TwoOpBT[2, {T, B}, 1, 2, 12, 2] == AF[w[12]⊗s[2]]
```

True

Figure 25: Additional hybrid BT operator examples

4. Entanglement

4.1. Schmidt decomposition

The Schmidt decomposition tutorials (SchmidtTutorial2014 and SchmidtTutorial-Qutrits2015) show how to decompose a B,T, or BT bipartite state into Schmidt form. See Fig. 26 for a sample run for a random two qubit case. The command Schmidt is based on the MM SingularValueDecomposition(SVD) command. The Schmidt number is defined as the number of nonzero entries in the diagonal matrix ws generated by the SVD. From the Schmidt decomposition, a bipartite state is entangled if and only if ws has Schmidt number greater than 1. Several special states, such as Werner and X states are examined in tutorials. Generalization to bipartite, random BT cases are demonstrated in SchmidtTutorial-Qutrits2015 . It is also shown how to maintain a right handed coordinate system under a SVD.

```
m = RandomKetN[2];
DForm[%]
Schmidt[m];
Print["ϕA1=", ϕA1]
Print["ϕA2=", ϕA2]
Print["ϕB1=", ϕB1]
Print["ϕB2=", ϕB2]
Print["ws=", MatrixForm[ws]]
Print["Snos=", Snos]
```

```
 + (0.660221 + 0.0769241 i) |00 > + (-0.0400115 - 0.193402 i) |01 >
  + (-0.448846 + 0.443145 i) |10 > + (0.295271 + 0.184827 i) |11 >

ϕA1= + (-0.693073) |0 > + (0.457815 - 0.556827 i) |1 >

ϕA2= + (-0.720868) |0 > + (-0.440163 + 0.535357 i) |1 >

ϕB1= + (-0.915242 - 0.100963 i) |0 > + (0.0603508 + 0.385353 i) |1 >

ϕB2= + (-0.378547 - 0.0940286 i) |0 > + (-0.0200276 - 0.920576 i) |1 >

ws=( 0.994081     0.
     0.     0.108641 )

Snos=2
```

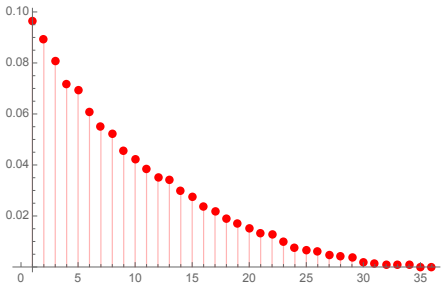
Check on SVD and on Schmidt decomposition

```
ws
U
Vs
U.ws.Vs
U.ws.Vs == Coeff[m]
```

```
( 0.994081     0.
     0.     0.108641 )

(     -0.693073 + 0. i          -0.720868 + 0. i      )
( 0.457815 - 0.556827 i   -0.440163 + 0.535357 i )

(  -0.915242 - 0.100963 i   0.0603508 + 0.385353 i )
( -0.378547 - 0.0940286 i  -0.0200276 - 0.920576 i )

( 0.660221 + 0.0769241 i  -0.0400115 - 0.193402 i )
( -0.448846 + 0.443145 i   0.295271 + 0.184827 i  )

True
```

Figure 26: Schmidt decomposition for a random 2 qubit state m.

```
QA = {B, T, B, T};
Ω = RandomBT[QA];
dim = Length[Ω];
ev = Chop[Eigenvalues[Ω]];
ListPlot[ev, Filling → Axis, Joined → False, AxesOrigin → {1, 0}, PlotStyle → {Red, PointSize[Large]}]
Chop[Tr[Ω.Ω]]
Tr[ev] == Sum[ev[[i]], {i, 1, dim}] == Tr[Ω] == 1.
ev.ev == Sum[ev[[i]]^2, {i, 1, dim}] == Purity[Ω] == Chop[Tr[Ω.Ω]]
SAB = EnTropy[Ω]
```




```
0.0550858

True

True

4.45628

ΩB = PTrBT[QA, {1, 2}, Ω];
Chop[Tr[%]] == 1.
ΩA = PTrBT[QA, {3, 4}, Ω];
Chop[Tr[%]] == 1.
SA = EnTropy[ΩA];
SB = EnTropy[ΩB];
Mutual = SA + SB - SAB
Length[ΩA] Length[ΩB] == Length[Ω]

True

True

0.673053

True
```

Figure 27: Mutual Information example for a random BT density matrix Ω.

```
Ω = RandomQubit2;
{SA, SB, SAB, ℐ, CA, QA, enTa} = Discord[Ω, 1, 50];
{SA, SB, SAB, ℐ, CB, QB, enTb} = Discord[Ω, 2, 50];
Print["SA=", SA, "    SB=", SB, "    SAB=", SAB, "    ℐ=", ℐ]
Print["CA=", CA, "    DA=", QA]
Print["CB=", CB, "    ", "DB=", QB]
```

```
SA=0.985157    SB=0.991471    SAB=1.38868    ℐ=0.587947

CA=0.331379    DA=0.256568

CB=0.335641    DB=0.252306
```

```
pl1 = Plot3D[enTa[x, y], {x, 0, Pi}, {y, 0, 2 Pi}]; pl2 = Plot3D[enTb[x, y], {x, 0, Pi}, {y, 0, 2 Pi}];
GraphicsRow[{pl1, pl2}]
```

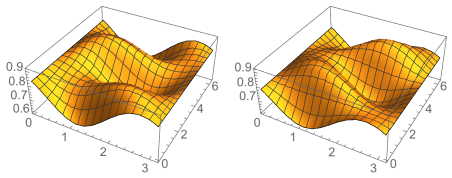


Figure 28: Discord example. Here Ω is a random 2-qubit density matrix, $\mathcal{I}$ is the mutual information, CA the classical and QA the discord. Surfaces for classical search are also shown.

```
Discord[Ω_ /; MatrixQ[Ω], Icase_, npoints_] :=
  Module[{rhoAB, rhoA, rhoB, SAB, SA, SB, ℐ, M, op, ic, p, pv, ρ, rho,
   data, smin, C, Q, enT, npp },
npp = npoints;
rhoAB = Ω; rhoA = PTr[{2}, rhoAB]; rhoB = PTr[{1}, rhoAB];
SAB = EnTropy[rhoAB]; SA = EnTropy[rhoA]; SB = EnTropy[rhoB];
ℐ = Chop[SA + SB - SAB];
M[k_] := FullSimplify[ProjG[k, {Sin[θ] Cos[ϕ], Sin[θ] Sin[ϕ], Cos[θ]}],
    {ϕ, θ} ∈ Reals];
op = If[Icase == 1, AF[Subscript[σ, 0]⊗M[k]], AF[M[k]⊗Subscript[σ, 0]]];
ic = If[Icase == 1, 2, 1];
p[k_] := Simplify[Tr[op.rhoAB.op], {ϕ, θ} ∈ Reals];
pv[k_, thet_, phi_] := Chop[p[k] /. {θ -> thet, ϕ -> phi}];
ρ[k_] := Simplify[PTr[{ic}, op.rhoAB.op], {ϕ, θ} ∈ Reals];
rho[k_, thet_, phi_] := Chop[ρ[k] / p[k] /. {θ -> thet, ϕ -> phi}];
  Clear[enT];
enT[thet_, phi_] := Sum[pv[k, thet, phi] * EnTropy[rho[k, thet, phi]],
    {k, 0, 1}];
et[i_, j_] := enT[i * Pi / (npp), j * 2 Pi / npp];
data = MapThread[et, {Range[0, npp], Range[0, npp]}];
smin = Min[data];
  C = If[Icase == 1, Chop[SA] - smin, Chop[SB] - smin ];
Q = ℐ - C;
{SA, SB, SAB, ℐ, C, Q, enT}]
```

Figure 29: The module Discord$[\rho, Ic, np] = \mathcal{I}[\rho] - \mathcal{C}[\rho]$, where $\mathcal{C}$ is the classical evaluation of the conditional entropy $< \mathcal{J} >_C$. Note Ic selects the discord case $\mathcal{J}A$ or $\mathcal{J}B$ and np stipulates the space used in the requisite minimization and in the plot. Discord provides the output array (SA, SB, SAB, $\mathcal{I}, \mathcal{C}, \mathcal{D}$, plot).

The entropy command EnTropy [8] is used to evaluate the von Neumann entropy from the real, positive eigenvalues of a density matrix ρ, $EnTropy[\rho] = -Tr[\rho \ln \rho]$. The density matrix can be generated from a random density matrix command, (RandomQubit1, RandomQubit2, RandomQubitN, RandomQutrit, or RandomBT), or from a state Ψ just by the command $\rho[\Psi]$. Density matrix construction for BT systems is explored in the notebook DensityMatrixTutorial.

4.2.1. Partial Trace

Once the entropy is determined from a given density matrix, the PartialTrace and PartialTraceBT commands can be used to determine the sub-system density matrices and their associated entropy, and thus the mutual information is specified. For example, using PartialTraceBT for a RandomBT density matrix, the mutual information $\mathcal{I} \equiv SA + SB - SAB$ is determined in Fig. 27. Here $SAB = \text{EnTropy}[\rho], SA = \text{EnTropy}[\rho_A], SB = \text{EnTropy}[\rho_B]$, where SAB is the full system entropy, and SA, SB are the subsystem entropies. The subsystem density matrices are: $\rho_A = Tr_B[\rho] = PTr[\{2\}, \rho]$ and $\rho_B = Tr_A[\rho] = PTr[\{1\}, \rho]$

4.2.2. Discord

The quantum discord involves using two forms for the mutual information that are equal for a classical system, but differ for a quantum system. The idea of quantum discord was introduced in references [11, 12], where Ollivier and Zurek[11] describes it as " Two classically identical expressions for the mutual information generally differ when the systems involved are quantum. This difference defines the quantum discord. It can be used as a measure of the quantumness of correlations." The discord indicates that quantum effects, but not necessarily quantum entanglement, exist in the system,.

The two quantum relations for the mutual information are

$$\begin{aligned} \mathcal{I} &= S[\rho_A] + S[\rho_B] - S[\rho] \\ &and \\ \mathcal{J}A &= S[\rho_B] - S[B|A], \\ &or \\ \mathcal{J}B &= S[\rho_A] - S[A|B], \end{aligned} \tag{23}$$

where $S[\rho_A]$ and $S[\rho_B]$ are the von Neumann entropies for the A and B subsystems, respectively. That is : $S[\rho_A] = EnTropy[\rho_A], S[\rho_B] = EnTropy[\rho_B]$, where $\rho_A = Tr_B[\rho], \rho_B = Tr_A[\rho]$, and $S[A, B] = EnTropy[\rho]$. Here $S[A, B]$ is the joint entropy. The quantities $S[A|B] = S[\rho_A|\rho_A], S[B|A] = S[\rho_B|\rho_A]$ denote quantum conditional entropies. Note that for the case that A and B are separate and unconnected systems, the density matrix is a product $\rho = \rho_A \otimes \rho_B$ and then the mutual information $\mathcal{I} = 0$.

[8] We use the command EnTropy to obtain the von Neumann entropy, The MM command Entropy is not the same quantity.

For a classical system, classical renditions of the two expressions for the mutual information $\mathcal{I}, \mathcal{J}A$ yield identical results,; however, for the quantum case $\mathcal{I} \neq \mathcal{J}A$. The difference between the two results are used to define the discord

$$\begin{aligned} \mathcal{D}_A(\rho) &= \mathcal{I} - \mathcal{J}A \\ &\textit{and} \\ \mathcal{D}_B(\rho) &= \mathcal{I} - \mathcal{J}B. \end{aligned} \tag{24}$$

In general, $\mathcal{D}_A(\rho) \neq \mathcal{D}_B(\rho)$. The problem now is how to evaluate $\mathcal{J}A$ and $\mathcal{J}A$. The quantities $\mathcal{J}A$ and $\mathcal{J}A$ represent "the part of the correlations that can be attributed to classical correlations and varies in dependence on the chosen eigenbasis; therefore, in order for the quantum discord to reflect the purely nonclassical correlations independently of basis, it is necessary that J first be maximized over the set of all possible projective measurements onto the eigenbasis." [13] That process, which entails an evaluation of the conditional entropy using a classical limit method, is incorporated into Qdensity as shown in Fig. 29.

Sample cases are shown in the three discord notebooks.

4.3. Partial Transposition

Examples of partial transposition are shown in the tutorial " PartialTranspose Tutorial." In this package, the partial transpose is obtained by expanding the density matrix in the computational (Pauli tensor product) basis. Then for the stipulated subsystem, the subsystem Pauli matrices are transposed, and then the transposed density matrix is reconstructed. The only subsystem operators that are affected are those involving the y-component Pauli operators. This procedure is encoded in the module PartialTranspose[BL,ρ] , where BL stipulates the subsystem and ,ρ is the original density matrix. This procedure is generalized to hybrid (BT) systems by the module PartialTransposeBT[QA,BL,ρ], BL again stipulates the subsystem and now the array QA gives the BT mixture. Examples and tests are provided in the PartialTransposeTutorial and also used to demonstrate the Peres−Horodecki [14] criterion for separable density matrices in tutorials ".

Calculation and display of Entropy, mutual information and quantum discord are also provided in " PartialTranspose Tutorial."

4.4. Bell's theorem

In the Bell theorem and Bell Correlations2015 notebooks, sample cases of Bell's theorem are examined using this package. For discussion, see : [6, 7, 15, 16, 17]

5. Other new aspects

5.1. QC algorithms & Simulations

Updates of various QC algorithms [18, 19, 20] are provided in Teleportation2014, Grover2014, Shor2014, QFT2014 and Cluster2014. A sample of teleportation of a qubit is contained in TeleportationBT, Studies of random states, random density matrices, Werner [21] states, GHZ [22] states and X [23] states are presented in various notebooks included with this package in many cases with qutrit or hybrid system examples (see the appendix). A preliminary study of concurrence is present in the notebooks ConcurrenceTutorial and ConcurrenceTutorialBT [24, 25]

6. Future Plans: Parallel & Cuda versions

This update and extension provides many basic tools for studying the efficacy of quantum computers. Some of the cases that are not included here include: (1) quantum error correctionr [26], (2) dynamical evolution of gates, (3) density matrix dynamics including entropy constraints,(4) solutions of differential equations, (5) single and multiple photon qubit states (6) Quantum Tomography. Preliminary versions of these case are available, but not included in the present release. The applications will hopefully improve, increase, and broaden with time, perhaps by interested users.

The hope is to improve and extend this package, by future application of MPI parallel methods to enable faster and larger system studies. Use of the GPU processor for parallel computation is also a future goal, once the high latency problem for large system tensor product formation is solved. A web page presentation of this package is available and questions, suggestions for future developments, and comments, are welcome, so that these packages can be further improved. Do not hesitate to contact the author for help.

Acknowledgments

The author is very appreciative of the help provided by his collaborator Dr. Bruno Juliá-Díaz, who was one of the original developers of this project. Questions and comments provided by Dr. Kapil K. Sharma are also very much appreciated; he stimulated the extensions to hybrid systems, partial transposition, and to quantum discord. My interest in quantum tomography was greatly enhanced by communications with Dr. Victor Volkov. Earlier versions of this project were supported by the National Science Foundation.

References

[1] Bruno Juliá-Díaz, Joseph M. Burdis and Frank Tabakin, "QDENSITY - A Mathematica Quantum Computer simulation," Comp. Phys. Comm., 174 (2006) 914-934. Also see: Comp. Phys. Comm.,180, (2009) 474.

[2] Frank Tabakin and Bruno Juliá-Díaz, "QCWAVE A Mathematica quantum computer simulation update," Comp. Phys. Comm., 182, (2011)1693.

[3] Frank Tabakin and Bruno Juliá-Díaz, "QCMPI: A parallel environment for quantum computing", Comp. Phys. Comm., 180 (2009) 948-964.

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

[5] Albert Messiah, "Quantum Mechanics" , Dover Publications , ISBN : 0486409244.

[6] Michael A. Nielsen and Isaac I. Chuang, "Quantum Computation and Quantum Information", Cambridge University Press (2000).

[7] John Preskill, "Lecture Notes on quantum information and computation", http://www.theory.caltech.edu/people/preskill/ph229/.

[8] C. M. Wilmott and P. R. Wild, Int. J. Quantum Inform. 10, 1250034 (2012).

[9] Juan Carlos Garcia-Escartin, Pedro Chamorro-Posada,"A SWAP gate for qudits," Quantum Information Processing,Vol .12,(2013).

[10] Peter B R Nisbet-Jones, Jerome Dilley, Annemarie Holleczek, Oliver Barter and Axel Kuhn, "Photonic qubits, qutrits and ququads accurately prepared and delivered on demand," New Journal of Physics 15 (2013) 053007.

[11] H. Ollivier and W. H. Zurek, Quantum discord: A measure of the quantumness of correlations, Phys. Rev. Lett. 88, 017901 (2002).

[12] L. Henderson and V. Vedral: Classical, quantum and total correlations, Journal of Physics A 34, 6899 (2001).

[13] https://en.wikipedia.org/wiki/

[14] Asher Peres, Separability Criterion for Density Matrices, Phys. Rev. Lett. 77, 14131415 (1996) and Michal Horodecki, Pawel Horodecki, Ryszard Horodecki, Separability of Mixed States: Necessary and Sufficient Conditions, Physics Letters A 223, 1-8 (1996).

[15] Bell, John (1964). "On the Einstein Podolsky Rosen Paradox". Physics 1 (3): 195200.

[16] JS Bell (2004), Speakable and Unspeakable in Quantum Mechanics: Cambridge University Press. 2nd Edition(2004) ISBN: 9780521523387.

[17] http://www.lecture-notes.co.uk/susskind/quantum-entanglements/lecture-5/violation-of-bells-theorem/ .

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

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

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

[21] Reinhard F. Werner ,Physical Review A 40 (8) 42774281.

[22] Daniel M. Greenberger, Michael A. Horne, Anton Zeilinger:, "Bell's theorem, Quantum Theory, and Conceptions of the Universe," pp. 73-76, Kluwer Academics, Dordrecht, The Netherlands (1989).

[23] Sai Vinjanampathy and A. R. P. Rau, Phys. Rev. A 82, 032336 (2010).

[24] W. K. Wootters, "Entanglement of Formation and Concurrence," Quantum Information and Computation 1, 27 (2001) and W. K. Wootters,Phys.Rev. Letters 80,2245 (1998).

[25] E. Gerjuoy Phys. Rev. A 67, 052308 (2003).

[26] D. Gottesman, "A Theory of Fault-Tolerant Quantum Computation," Phys. Rev. A 57, 127-137 (1998).

A. Tutorials

The following tutorials, workbooks and algorithms are part of the package and should aid the user in generating their own examples. Guidance for installation is provided in the notebook INSTALL. Email tabakin@pitt.edu for the packages and for these and additional tutorials.

1. BTGates
2. BTtutorial2014
3. Bell Correlations2015
4. Belltheorem
5. CircuitTutorial2014
6. CircuitTutorialBT2015
7. Cluster2014
8. ConcurrenceTutorial
9. ConcurrenceTutorial2
10. ConcurrenceTutorial3
11. ConcurrenceTutorialBT
12. DensityMatrixTutorial
13. Discord-Tests 2015
14. DiscordTutorial2015
15. Entanglement2014
16. EntropyTutorial2015
17. FunctionsTour2014
18. GatesQudits2015
19. Grover2014
20. HybridGates
21. INSTALL
22. Measurement
23. PartialTransposeTutorial
24. QCwaveTutorial2014
25. QFT2014
26. Quantum EntropyTutorial2014
27. Qutrit operators
28. RandomObs
29. SchmidtTutorial-Qutrits2015
30. SchmidtTutorial2014
31. Shor2014
32. Teleportation2014
33. TeleportationQutrit
34. TernaryTutorial
35. Tutorial2014
36. TutorialTP
37. WernerStates2014
38. WorkBookPartnerBT
39. XState Tutorial 2015
40. XState Tutorial