Exercise 8
Density Matrices

Michele Guadagnini - ID 1230663

December 1, 2020

Abstract

The aim of this exercise is to create a program to compute the Density Matrix starting from the state of the system represented by a vector of coefficients both in the separable and entangled case. The program is then tested on a 2-Spin $\frac{1}{2}$ system.

1 Theory

Given a Hilbert space of dimension D, $\mathcal{H}^D$, a general wave function Ψ of N-body quantum system can be described as:

$$|\Psi\rangle = \sum_{\alpha_1,\ldots,\alpha_N} C_{\alpha_1,\ldots,\alpha_N} |\alpha_1\rangle \otimes |\alpha_N\rangle \tag{1}$$

The wave function above can be rewritten as:

$$|\Psi\rangle = \sum_{i}^{D^N} C_i |i\rangle , \quad where: \quad |i\rangle \in \mathcal{H}^{D^N} \tag{2}$$

The vector of coefficients C_i contains D^N elements that, in general, are of complex type. In the particular case that the total wave function can be separated into N definite pure states, it becomes:

$$|\Psi\rangle = \sum C_{i,\alpha_i} |\alpha_i\rangle = \sum_{i}^{N} C_i |i\rangle \tag{3}$$

This time the total wave function only requires $D \times N$ coefficients to be stored.

To characterize the statistical state of a quantum system it is convenient to use the *density matrix*. For a general pure state Ψ, it is defined as:

$$\rho = |\Psi\rangle \langle\Psi| \tag{4}$$

A general density matrix have the following properties:

- it is *hermitian*;
- the normalization of Ψ implies that $Tr(\rho) = 1$;
- it is a positive semi-definite matrix.

Also, for a pure state, we have that $\rho^2 = \rho$.

When dealing with *composite* systems it is possible, starting from the general density matrix, to retrieve the state of the subsystem by computing the *reduced density matrix*. Supposing to have 2 systems, A and B, the reduced density matrix of system A can be derived from the general ρ_{AB} by tracing over the basis of the system B:

$$\rho_{A_{\alpha,\alpha'}} = \sum_{\beta} \langle\beta| \langle\alpha| \rho_{AB} |\alpha'\rangle |\beta\rangle = Tr_B(\rho_{AB}) \tag{5}$$

The reduced density matrix has the same properties of the general density matrix described above.

2 Code Development

2.1 Design and Implementation

The implementation started by building up the module *DensityMatrices*, that contains all the subroutines used in this assignment. In particular:

- *InitParamsFromFile* : it reads the parameters from the configuration file: $Hdim$, the dimension of the Hilbert space, $Nsys$, the number of subsystems, Sep, a logical flag for separability of the pure state, and the logical flag to active $Debug$.
- *AllocPurePsi* : it allocates the memory needed to store the wavefunction. It reserve a space of $D \times N$ elements if the state is separable, otherwise it reserves D^N memory places.
- *PureDensityMat* : it computes the *Density Matrix* from the Ψ vector, directly if it is not separable or taking care to calculate the proper indexes if it is separable. The code is reported in Listing 1.
- *RedDensityMat* : it computes the left or right reduced density matrix according to a flag passed to the subroutine. This part of the code is reported in Listing 2. Note that it assumes that N, the number of subsystems is 2.
- finally, it contains other subroutines to random initialize a state, compute *Norm* and *Trace* and to store a density matrix on file.

```
122         !Distinguish between SEPARABLE and NOT SEPARABLE cases
123         IF (psi%separable) THEN
124             DO kk=1,MatSize
125                 !computing the indeces i and j from k
126                 jidx = psi%DD +1 + MOD((kk-1),psi%DD)
127                 iidx = (kk-1)/psi%DD +1
128                 temp(kk) = psi%WF(iidx)*psi%WF(jidx)
129             ENDDO
130             temp = temp / L2Norm(temp) !normalization
131             DO kk=1,MatSize
132                 Dmat(kk,:) = temp(kk)*DCONJG(temp(:))
133             ENDDO
134         ELSE    !Not separable case
135             DO kk=1,MatSize
136                 Dmat(kk,:) = psi%WF(kk)*DCONJG(psi%WF(:))
137             ENDDO
138         ENDIF
```

Listing 1: Density Matrix computation.

```
164         IF (left) THEN
165         !Tracing out the right system (i.e. computing rho_A from rho_AB)
166             DO ii=1,Pars%Hdim
167                 ki = 1 + Pars%Hdim*(ii-1)
168                 DO jj=1,Pars%Hdim
169                     kj = 1 + Pars%Hdim*(jj-1)
170                     temp = DCMPLX(0d0,0d0)
171                     DO tt=0,Pars%Hdim-1
172                         temp = temp + DMat(ki+tt,kj+tt)
173                     ENDDO
174                     RedDMat(ii,jj) = temp
175                 ENDDO
176             ENDDO
177         ELSE
178         !Tracing out the left system (i.e. computing rho_B from rho_AB)
179             DO ii=1,Pars%Hdim
180                 DO jj=1,Pars%Hdim
181                     temp = DCMPLX(0d0,0d0)
182                     DO tt=0,Pars%Hdim-1
183                         temp = temp + DMat(ii+tt*Pars%Hdim,jj+tt*Pars%Hdim)
184                     ENDDO
185                     RedDMat(ii,jj) = temp
186                 ENDDO
187             ENDDO
188         ENDIF
```

Listing 2: Reduced Density Matrix computation.

To complete this exercise, two main program has been created. The first one, *Ex8-Guadagnini-Test-CODE.f90*, is used to test all the subroutines and functions of the module described above. By editing its configuration file, *Config_ Pars.txt*, it is possible to set any value for $Hdim$ and $Nsys$. Since some of the subroutines only work for $N = 2$ a part of the program (the one that computes the reduced densities) is skipped if $N > 2$, while for the value of D in principle any value is possible.

The second program, *Ex8-Guadagnini-2SpinHalf-CODE.f90*, performs the same computation but in a system composed by two particles of spin 1/2. The state of the system is not set randomly as in the previous case. This time one over two possible states is chosen according to the separability flag.

2.2 Debug and Test

The compilation and execution commands are quite straightforward since no particular external libraries has been used. The first test done has been verifying the maximum values of the input parameters before crashing due to memory errors. It happened that, with $Hdim = 10$, $Nsys = 4$ and $Sep = F$ the program still works, but it consumes almost all the available memory (of the particular machine used). Increasing one of the two parameters leads to the OS preventing the allocation of the memory. In fact, for the non separable case the number of memory places needed to store the density matrix is exponential, precisely D^N.

Other tests and checks has been done during implementation, such as verifying the normalization of the wavefunction, the trace of the density matrix that must be 1, etc.

3 Results

The results of the test program are reported in the file *DensityMat.txt*. The parameters used in this case are: $Hdim = 3$, $Nsys = 2$, $Sep = FALSE$. The wavefunction has been initialized with random numbers whose *real* and *imaginary* parts are uniformly distributed between 0 and 1. It can be seen that the trace of this matrix is 1 and that it is hermitian, as expected. The same considerations can be done also on the left and right reduced density matrices, stored respectively in the files *RedMat_ Left.txt* and *RedMat_ Right.txt*.

The program regarding *2-Spin* $\frac{1}{2}$ has been run twice changing the value of the separability flags in its configuration file, *Config_ 2SpinHalf.txt*. The two states that are coded in the program are:

$$|\Psi_{sep}\rangle = |0_A\rangle \otimes \frac{1}{\sqrt{2}}(|0_B\rangle + |1_B\rangle) \tag{6}$$

$$|\Psi_{ent}\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \tag{7}$$

Equation 6 shows a separable state, while Equation 7 shows an entangled one.

The density matrix obtained in the separable case is showed in Equation 8,

$$\rho_{AB} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{8}$$

while the reduced ones are reported in Equation 9.

$$\rho_A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad \rho_B = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \tag{9}$$

Also this time the properties defined above for the density matrix hold.

In the same way also the results for the entangled case are reported in the following equations.

$$\rho_{AB} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & -\frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{10}$$

$$\rho_A = \rho_B = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \tag{11}$$

4 Self-evaluation

This assignment required to learn how to compute the general and reduced density matrices from a vector of coefficients representing the state of the system.

The program only works for low number of subsystems and low dimension of the Hilbert space. The problem is mainly the memory needed to store the density matrix. One small improvement that can be done is, since the density matrix is hermitian, to store only the upper triangular part of it, since the lower triangular is the complex conjugate of the upper one.

Also, the advantage of storing only $D \times N$ coefficients for the wavefunction in the separable case vanishes since the program, as it is implemented in this moment, computes

and stores anyway the full density matrix. A better idea would be to not store the density matrix, but only compute the elements when they are needed.

Another limit of the program is that the computation of the partial trace is possible only for systems with 2 particles, but generalizing to arbitrary N is not trivial.