Density matrices

Abstract

In this work we write a Fortran program to write the pure wave function of composite systems, compute the density matrix and the reduced density matrix.

1 Theory

Let us consider a quantum composite system which is composed by N subsystems, each described by its wave function $\psi_i \in \mathcal{H}^D$ where $\mathcal{H}^D$ is a D-dimensional Hilbert space. The total wave function of the system can be written as

$$\Psi = \sum_{\alpha_1,\dots,\alpha_N} c_{\alpha_1,\dots,\alpha_N} |\alpha_1\rangle \otimes \cdots \otimes |\alpha_N\rangle \in \mathcal{H}^{D^N} \tag{1}$$

where $\{|\alpha_i\rangle\}_{i=1,\dots,N}$ are the single particle's basis.
In general, we need D^N complex numbers to specify this wave function. However, in the particular case of *separable* pure states we have

$$\Psi = \bigotimes_{i=1}^{N} \sum_{\alpha_i} c_{\alpha_i} |\alpha_i\rangle \tag{2}$$

meaning that we only need DN complex numbers to specify the wave function.

Let us consider the case $N = 2$ and denote the two subsystems with A and B. In this case, the density matrix of a pure state

$$|\Psi\rangle = \sum_{\alpha_A,\alpha_B} c_{\alpha_A,\alpha_B} |\alpha_1\rangle \otimes |\alpha_2\rangle \tag{3}$$

in $\mathcal{H}_{\mathcal{AB}} = \mathcal{H}_A \otimes \mathcal{H}_B$ is given by

$$\rho_{AB} = |\Psi\rangle\langle\Psi| = \sum_{\alpha_A,\alpha_B,\alpha'_A,\alpha'_B} c_{\alpha_A,\alpha_B} c^*_{\alpha'_A,\alpha'_B} |\alpha_A,\alpha_B\rangle\langle\alpha'_A,\alpha'_B| \tag{4}$$

and it is such that

$$\mathrm{Tr}(\rho_{AB}) = 1. \tag{5}$$

We can compute the reduced density matrices for the two subsystems by performing a *partial trace*, in which we compute the trace using the basis of one of the two subsystems. For example, if we want to find the density matrix of subsystem A:

$$\rho_A = \mathrm{Tr}_B(\rho_{AB}) = \sum_{\alpha} \langle\alpha|_B \, \rho_{AB} \, |\alpha\rangle_B \,. \tag{6}$$

2 Code development

The main program is structured as follows. First, we write the total wave function of a N-body composite system both in the case of non-interacting, separable pure state and in the general pure case. In order to do that we use the `InitPureSepStat` and the `InitPureGenStat` subroutines. In particular, the former does not write the $|\psi\rangle$ in its "extended" form, but – taking advantage of the properties of separable states – encodes it in a $D \times N$ matrix. Conversely, the latter directly writes the full $|\psi\rangle$.

```fortran
subroutine InitPureSepState(sep_coeff)
    integer :: ii, jj, N, D
    real*8 :: RePart, ImPart
    complex*16, dimension(:,:) :: sep_coeff
    N = size(sep_coeff, 2)
    D = size(sep_coeff, 1)
    do ii = 1, N
        do jj = 1, D
            call random_number(RePart)
            call random_number(ImPart)
            sep_coeff(jj, ii) = dcmplx(RePart, ImPart)
        end do
        sep_coeff(:, ii) = sep_coeff(:, ii) / sqrt(sum(abs(sep_coeff(:, ii))**2))
        if (abs(sum(abs(sep_coeff(:, ii))**2) - 1) .ge. 1e-4) then
            print *, "Normalization error!"
            stop
        end if
    end do
end subroutine InitPureSepState
```

Then, we compute the density matrix ρ as the outer product $|\psi\rangle\langle\psi|$ by means of the subroutine `OuterProd`. For the separable state, we first build the full $|\psi\rangle$ using the `BuildSeparableStateFromMatr` subroutine, which, starting from the matrix `sep_coeff`, computes the coefficients of the wave function using a base D encoding with N digits:

$$c_{full,i} = \prod_{d=1}^{D} c_d \left(\frac{i}{D^{N-d}} \mod D \right)$$

where $c_d(m)$ is the m-th coefficient of d-th subsystem.

```fortran
subroutine BuildSeparableStateFromMatr(coeff_matrix, psi)
    integer :: ii, jj, D, N
    complex*16, dimension(:,:) :: coeff_matrix
    complex*16, dimension(size(coeff_matrix, 1)**size(coeff_matrix, 2)) :: psi
    D = size(coeff_matrix, 1)
    N = size(coeff_matrix, 2)
    psi = dcmplx(1d0)
    do ii = 1, D**N
        do jj = 1, N
            psi(ii) = psi(ii) * coeff_matrix(mod((ii-1)/(D**(N-jj)),D)+1, jj)
        end do
    end do
end subroutine BuildSeparableStateFromMatr
```

Then, we check if the two density matrices have unitary trace using the `ComplexSquareMatrixTrace` function. Finally, assuming $N = 2$, we compute the reduced density matrix for both subsystems. This is done through two subroutines, `ReducedDensityMatrixB` and `ReducedDensityMatrixA`.

```fortran
subroutine ReducedDensityMatrixA(rho, D, rho_A)
    integer :: D, jj, kk, ll
    complex*16, dimension(:,:) :: rho
    complex*16, dimension(D, D) :: rho_A
    rho_A = 0.d0
    do jj = 1, D
        do kk = jj, D
            do ll = 1, D
                rho_A(jj, kk) = rho_A(jj, kk) + rho((jj-1)*D+ll, (kk-1)*D+ll)
            end do
            if (jj /= kk) rho_A(kk, jj) = conjg(rho_A(jj, kk))
        end do
    end do
end subroutine ReducedDensityMatrixA

subroutine ReducedDensityMatrixB(rho, D, rho_B)
    integer :: D, jj, kk, ll
    complex*16, dimension(:,:) :: rho
    complex*16, dimension(D, D) :: rho_B
    rho_B = 0.d0
    do jj = 1, D
        do kk = jj, D
            do ll = 1, D
                rho_B(jj, kk) = rho_B(jj, kk) + rho((ll-1)*D+jj, (ll-1)*D+kk)
            end do
            if (jj /= kk) rho_B(kk, jj) = conjg(rho_B(jj, kk))
        end do
    end do
end subroutine ReducedDensityMatrixB
```

The partial trace elements are computed as the following sum:

$$\rho^p_{j,k} = \sum_{l=1} \rho_{jl,kl}.$$

The problem is that in this formula we have four indices, but only two of them are in memory. To solve this issue we can consider a mapping between the four indices and the two. Suppose to have two subsystems of dimension m and n. The composite system will have dimension mn and its basis will be $|k\rangle\,|l\rangle$ with $k = 1, \dots, m$ and $l = 1, \dots, n$. Now, we "flatten" these two indices in a single index j, such that

$$j = (k-1)n + l.$$

Since we have four indices, we have to repeat this procedure two times, one to get the row index and one to get the column one:

$$(jl, kl) = ((j-1)n + l, (k-1)n + l).$$

Furthermore, in our case we have $m = n = D$, so the sums over j and l go both from 1 to D. Again, to check the correctness of the results we compute the trace of ρ_A and ρ_B.

2.1 Testing

In order to test the functions and subroutine we wrote to deal with density matrices, we write a simple program in which we set $N = D = 2$ and we consider the state

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$$

and we repeat the procedure above to check if the results are in line with the theoretical ones, which are

$$\rho = \begin{pmatrix} 1/2 & 1/2 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad \rho_A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \qquad \rho_B = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}.$$

The results (ρ, ρ_A and ρ_B) are saved in different output files.

3 Results

As we previously said, the main advantage of having a separable state is that – instead of being stored in a D^N vector – it can be encoded into a $D \times N$ matrix. From the efficiency point of view, we go from an exponential complexity to a linear one.
Figure 1 shows the CPU time as function of the number of subsystems N. Notice that starting from $N = 31$ the CPU time is zero: this is certainly due to a memory error. In fact, since we are using `complex*16` numbers, we have to store $2^{31} \cdot 32 \sim 69$ GB in our RAM, which is impossible because our machine has only 8 GB and even using a swapfile it cannot get such memory.

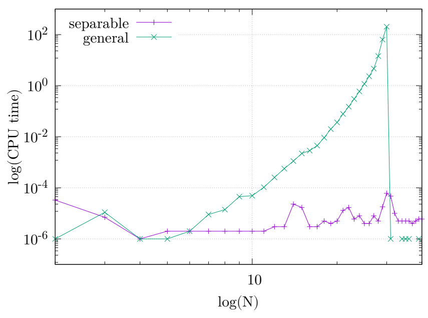

Figure 1: CPU time to initialize the state both in the general and separable cases.

3.1 Testing

The results of the computation for the chosen $|\psi\rangle$ are the following. Full density matrix ρ:

```
MATRIX:
(0.49999999999999989,0.0000000000000000)    (0.49999999999999989,0.0000000000000000) ↩
(0.0000000000000000,0.0000000000000000)     (0.0000000000000000,0.0000000000000000)
(0.49999999999999989,0.0000000000000000)    (0.49999999999999989,0.0000000000000000) ↩
(0.0000000000000000,0.0000000000000000)     (0.0000000000000000,0.0000000000000000)
(0.0000000000000000,0.0000000000000000)     (0.0000000000000000,0.0000000000000000) ↩
(0.0000000000000000,0.0000000000000000)     (0.0000000000000000,0.0000000000000000)
(0.0000000000000000,0.0000000000000000)     (0.0000000000000000,0.0000000000000000) ↩
(0.0000000000000000,0.0000000000000000)     (0.0000000000000000,0.0000000000000000)
TRACE: (0.99999999999999978,0.0000000000000000)
```

Reduced density matrix ρ_A:

```
MATRIX:
(0.99999999999999978,0.0000000000000000)     (0.0000000000000000,0.0000000000000000)
(0.0000000000000000,0.0000000000000000)      (0.0000000000000000,0.0000000000000000)
TRACE: (0.99999999999999978,0.0000000000000000)
```

Reduced density matrix ρ_B:

```
MATRIX:
(0.49999999999999989,0.0000000000000000)     (0.49999999999999989,0.0000000000000000)
(0.49999999999999989,0.0000000000000000)     (0.49999999999999989,0.0000000000000000)
TRACE: (0.99999999999999978,0.0000000000000000)
```

These results are in agreement with the theoretical ones. Of course, other tests with different states were done, all leading to correct results.

4 Self evaluation

This exercise was very instructive since it made me learn how to deal with density matrices in Fortran.

One possible improvement could be to expand the computation of the reduced density matrix for a generic composite system made up of N subsystems.