Quantum Information and Computing

prof: *Simone Montangero*

Report of exercise 8

Many-body quantum systems

Alberto Bassi

November 30, 2020

Abstract

The aim of this exercise is to write a Fortran code to work with general many-body quantum systems states and to reduce density matrices by means of partial traces.

Theory

Let us consider the generic N-particles Hilbert space $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_N$, where $\mathcal{H}_j$ are one-particle Hilbert spaces of dimension D, $\forall j$. Thus, the dimension of $\mathcal{H}$ is D^N. A generic separable (not entangled) pure state $|\psi\rangle$ in $\mathcal{H}$ can be written as $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_N\rangle$, where for every $j = 1, 2, \ldots, N$ we consider the computational basis $\{|i\rangle\}|_{i=1,2,\ldots,D}$ on $\mathcal{H}_j$ and thus write

$$|\psi_j\rangle = \sum_{i_j=0}^{D-1} C_{i_j} |i_j\rangle \ . \tag{1}$$

We remind that the tensor product of two states $|\psi_A\rangle \in \mathcal{H}_A$ and $|\psi_B\rangle \in \mathcal{H}_B$ is given by

$$|\psi\rangle = |\psi_A\rangle \otimes |\psi_B\rangle = \sum_{i,j} c_i c_j |ij\rangle \ , \tag{2}$$

where $\{|i\rangle\}|_{i=0,\ldots,dim\mathcal{H}_A}$ and $\{|j\rangle\}|_{j=0,\ldots,dim\mathcal{H}_B}$ are respectively the basis on $\mathcal{H}_A$ and $\mathcal{H}_B$. We point out that the tensor product index is the one obtained by "putting aside" the two subsystems indexes, with the formal association $(ij) \to (i * dim\mathcal{H}_B + j)$ that it will come up later again for the partial trace in a different form.
We notice that since every pure state is defined unless a non-null complex number, we only need $D-1$ complex coefficients C_{i_j} for each of them. Since the tensor product of normalized states is necessarily normalized, as it can be easily proven, every separable pure state has only $2(D-1)N$ degrees of freedom, despite the dimension of $\mathcal{H}$ is D^N.
Given the computational basis on each one-particle Hilbert space $\mathcal{H}_j$, a generic basis for $\mathcal{H}$ is $\{|i_i i_2 \ldots i_N\rangle\}|_{i_1,i_2,\ldots,i_n=0,\ldots,D-1}$.
Thus, the generic state on $\mathcal{H}$ is

$$|\psi\rangle = \sum_{i_1,i_2,\ldots,i_N} C_{i_1 i_2 \ldots i_N} |i_i i_2 \ldots i_N\rangle \ . \tag{3}$$

The most general description of a quantum mechanical states makes use of density matrices. Given a generic pure state $|\psi\rangle$, the associated density matrix is the projector on the state, namely $\rho = |\psi\rangle \langle\psi|$. For a generic pure state (Equation 3) in $\mathcal{H}$, we have

$$\rho = |\psi\rangle \langle\psi| = \sum_{\substack{i_1,i_2,\ldots i_N \\ i'_1,i'_2,\ldots,i'_N}} D^{i_1,i_2,\ldots,i_N}_{i'_1,i'_2,\ldots,i'_N} |i_i i_2 \ldots i_N\rangle \langle i'_i i'_2 \ldots i'_N| \ \in M_{D^N,D^N}(\mathbb{C}) \ , \tag{4}$$

where $D^{i_1,i_2,\ldots,i_N}_{i'_1,i'_2,\ldots,i'_N} \equiv C_{i_1 i_2 \ldots i_N} C^*_{i'_1 i'_2 \ldots i'_N}$ [1].
This is how a density matrices can be accounted in tensor notation. Instead, in matrix notation we make the

[1] * denotes the complex conjugate.

following associations $i \to (i_1, i_2, \dots, i_N)$, $i' \to (i'_1, i'_2, \dots, i'_N)$. Therefore, the density matrix becomes

$$\rho = \sum_{\substack{i=0,\dots,D^N-1 \\ i'=0,\dots,D^N-1}} \rho_{i,i'} \left|i\right\rangle \left\langle i'\right| \ . \tag{5}$$

More generally, ρ are hermitian, positive-definite, unitary-trace matrices for which $\rho^2 = \rho$ holds if and only if ρ describes a pure state, namely if $|\psi\rangle \in \mathcal{H}$ exists such that $\rho = |\psi\rangle\langle\psi|$. Let us suppose that $\mathcal{H} = \mathcal{H}_A \otimes \mathcal{H}_B$. Given a state $\rho \in \mathcal{H}$, we describe the reduced system in $\mathcal{H}_A$ by means of a partial trace $\rho_A = Tr_B(\rho)$ and equivalently $\rho_B = Tr_A(\rho)$. The system is separable only if $\rho = \rho_A \otimes \rho_B$, namely if we do not lose information by travelling from a system to another subsystem and vice-versa, meaning that there is no Entanglement.
Let us get back to the particular case in which $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_N$ and we have a generic pure state described by Equation 4. Let us suppose to reduce the D^N system into a D^{N-1} one by meaning of a partial trace over the Hilbert space $\mathcal{H}_k$. We will denote the reduced $(N-1)$-particle system as ρ^k_{N-1}. Thus, we have

$$\begin{aligned} \rho^k_{N-1} = Tr_k(\rho) = \sum_{\beta_k=0}^{D-1} \langle\beta_k| \rho |\beta_k\rangle = \sum_{\substack{i_1,i_2,\dots i_N \\ i'_1,i'_2,\dots,i'_N}} \sum_{\beta_k=0}^{D-1} D^{i_1,i_2,\dots,i_N}_{i'_1,i'_2,\dots,i'_N} \langle\beta_k|i_i i_2 \dots i_N\rangle \langle i'_i i'_2 \dots i'_N|\beta_k\rangle = \\ = \sum_{\substack{i_1,\dots,\bar{i_k},\dots,i_N \\ i'_1,\dots,\bar{i'_k},\dots,i'_N}} \sum_{\beta=0}^{D-1} D^{i_1,\dots,i_{k-1},\beta,i_{k+1},\dots,i_N}_{i'_1,\dots,i'_{k-1},\beta,i'_{k+1},\dots,i'_N} |i_i \dots \bar{i_k} \dots i_N\rangle \left\langle i'_i \dots \bar{i'_k} \dots i'_N\right| \ , \end{aligned} \tag{6}$$

where the bar over a index denotes that we are not considering that term in the summation. Formally, we have defined

$$|\beta_k\rangle \equiv \mathbb{1}_1 \otimes \cdots \otimes \mathbb{1}_{k-1} \otimes |\beta_k\rangle \otimes \mathbb{1}_{k+1} \otimes \cdots \otimes \mathbb{1}_N \ . \tag{7}$$

We see that, in tensor notation, the coefficient corresponding to the reduced multi-indexes $(i_1, \dots, i_{k-1}, i_{k+1}, \dots, i_N)$ and $(i'_1, \dots, i'_{k-1}, i'_{k+1}, \dots, i_N)$ is simply the tensor contraction

$$D^{i_1,\dots,i_{k-1},i_{k+1},\dots,i_N}_{i'_1,\dots,i'_{k-1},i'_{k+1},\dots,i'_N} = \sum_{\beta=0}^{D-1} D^{i_1,\dots,i_{k-1},\beta,i_{k+1},\dots,i_N}_{i'_1,\dots,i'_{k-1},\beta,i'_{k+1},\dots,i'_N} \ . \tag{8}$$

Clearly, similar formulas hold also when Hilbert spaces have not the same dimension.
But how to implement efficiently the latter equation? Let us think the multi-index $i = (i_1, \dots, i_{k-1}, i_{k+1}, \dots, i_N)$ as a number written in base D. In our computer, indexes run from 0 to D^{N-1} when computing the partial trace in decimal notation, so we have to convert numbers from base to base to compute the tensor contraction. So, let us say that the number i in decimal notation correspond to the number $(i_1, \dots, i_{k-1}, i_{k+1}, \dots, i_N)$ in base D. Since what we want to do is to put digit β in position k, first we have to divide by D^{N-k}, take the integer part to get rid of digits from position $k+1$ to N and then multiply by D^{N-k+1} in order to free space for β. Eventually, we must add again the latter $N-k$ digits and the digit β in position k. So, we get the association

$$i \to \sum_{\beta=0}^{D-1} \left[\left(D^{N-k+1} - D^{N-k}\right) int\left(i/D^{N-k}\right) + i + D^{N-k}\beta \right] \ , \tag{9}$$

[2]

where *int* returns the integer part (also *floor*).
Then, we repeat it again for the second index i'.

Code Development

We write another module called **manybodyQS** in which we write subroutines to deal with many-body quantum systems. First, we need a subroutine **genpure_state** to initialize a generic pure state in $\mathcal{H}$. We save it as a vector of dimension D^N that will be normalized in the main program. We always initialize generic states with uniform random numbers with real part in $[-1, 1]$ and imaginary part in $[-i, i]$.

```
1 subroutine genpure_state(state,DD,NN,status)
2         implicit none
3         double complex, dimension(:),allocatable :: state
4         integer :: status,DD,NN,ii
5         double precision :: re,im
6         allocate(state(DD**NN),stat=status)
```

[2] Here we are summing the elements labelled by those indexes.

```
          do ii=1,DD**NN
             call random_number(re)
             call random_number(im)
             state(ii)=cmplx(2*re-1,2*im-1)
          enddo
        end subroutine genpure_state
```

To initialize separable states, we could as well write them in a vector of dimension ND. Instead, we prefer to initialize every D-dimensional states one by one and then taking the tensor product. Once we have one-particle states (by putting N=1 in the previous subroutine), we can obtain many-particles separable states by performing the tensor product (Equation 2) with the subroutine **tensor_product**.

```
       subroutine tensor_product(stateA,dimA,stateB,dimB,prod,dim,count)
          implicit none
          integer :: dimA,dimB,dim,count,ii,jj
          double complex, dimension(dimA) :: stateA
          double complex, dimension(dimB) :: stateB
          double complex, dimension(dim) :: prod
          if(dimA*dimB.ne.dim) then
             print *, "Match dimensions failed in tensor product: ERROR"
             count=count+1
          endif
          do ii=0,dimA-1
             do jj=0,dimB-1
                prod(ii*dimB+jj+1)=stateA(ii+1)*stateB(jj+1)
             enddo
          enddo
        end subroutine tensor_product
```

Then, we need a subroutine **density_matrix** which computes the density matrix of a generic pure state of a generic dimension.

```
subroutine density_matrix(state,dim,density)
          implicit none
          integer :: dim,ii,jj
          double complex, dimension(dim) :: state
          double complex, dimension(dim,dim) :: density
          do jj=0,dim-1
             do ii=0,dim-1
                density(ii+1,jj+1)=state(ii+1)*conjg(state(jj+1))
             enddo
          enddo
        end subroutine density_matrix
```

So far, we have written two subroutines to compute the partial trace. The subroutine **partial_trace** computes the partial trace over the system $\mathcal{H}_B$ in the case our Hilbert space is $\mathcal{H} = \mathcal{H}_A \otimes \mathcal{H}_B$, by means of the index identification $i \to \sum_k (i * dim\mathcal{H}_B + k)$ and $j \to \sum_k (j * dim\mathcal{H}_B + k)$ [3].

```
   subroutine partial_trace(density,dim,densityA,dimA,dimB,count)
          implicit none
          integer :: dim, dimA,dimB,ii,jj,kk,count,flag
          double complex, dimension(dim,dim) :: density
          double complex, dimension(dimA,dimA) :: densityA
          if(dimA*dimB.ne.dim) then
             print *, "Subsystems dimensions product different: ERROR"
             count=count+1
          endif
          do jj=0,dimA-1
             do ii=0,dimA-1
                densityA(ii+1,jj+1)=cmplx(0d0,0d0)
                do kk=0,dimB-1
                   densityA(ii+1,jj+1)=densityA(ii+1,jj+1)+density(ii*dimB+kk+1,jj*dimB+kk+1)
                enddo
             enddo
          enddo
        end subroutine partial_trace
```

The other subroutine is **partial_trace_gen**, which computes the partial trace over the subsystem k only, through the indexes identification of Equation 9.

```
    subroutine partial_trace_gen(density,DD,NN,reduced,kk)
          implicit none
```

[3] As before, we are not summing the indexes, but the elements labelled by those indexes.

```
          integer :: DD,NN,kk,ii,jj, ss,alpha,beta
          double complex, dimension(DD**NN,DD**NN) :: density
          double complex, dimension(DD**(NN-1),DD**(NN-1)):: reduced
          do jj=0,DD**(NN-1)-1
             do ii=0,DD**(NN-1)-1
                reduced(ii+1,jj+1)=cmplx(0d0,0d0)
                do ss=0,DD-1
                   alpha=DD**(NN-kk)*ss+(DD**(NN-kk+1)-DD**(NN-kk))*int(ii/DD**(NN-kk))+ii
                   beta=DD**(NN-kk)*ss+(DD**(NN-kk+1)-DD**(NN-kk))*int(jj/DD**(NN-kk))+jj
                   reduced(ii+1,jj+1)=reduced(ii+1,jj+1)+density(alpha+1,beta+1)
                enddo
             enddo
          enddo
        end subroutine partial_trace_gen
```

Moreover, we write a subroutine to compute Von-Neumann entropy, which we expect to be zero for pure states.

```
subroutine entropy_vn(entropy,rho,dim)
          implicit none
          integer :: dim,lda,info,lwork,status,ii
          double complex, dimension(dim,dim):: rho
          double complex, dimension(:),allocatable :: work
          double precision, dimension(:),allocatable :: w,rwork
          double precision :: entropy
          lda=max(1,dim)
          lwork=max(1,2*dim-1)
          allocate(w(dim))
          allocate(work(lwork))
          allocate(rwork(max(1,3*dim-2)))
          call zheev('N','U',dim,rho,lda,w,work,lwork,rwork,info)
          entropy=0d0
          do ii=1,dim
             entropy=entropy-w(ii)*log(abs(w(ii)))
          enddo
          deallocate(w,work,rwork)
        end subroutine entropy_vn
```

We point out that we take the absolute value inside the logarithm, because eigenvalues can be sometimes negative (but always very very small in absolute value) due to numerical instability, while theoretically we expect them to be always positive, since they are probabilities (a pure state has all the eigenvalues 0 except one that is 1).
The other subroutine included in the module are not reported due to the limited space. The more important ones are **norm** to compute the norm of a generic state and **trace** to compute the trace of a generic density matrix. In the module **debugging** we have rewritten the subroutine **check_diff**, which now check if two double complex matrices are different by seeing if at least one element-wise difference norm is greater or equal a certain threshold.

Results

In order to verify the correctness of all the subroutines, we have done the following procedure. In the main program **many_body** in *main* we have initialized three one-particle pure states of dimension D (by putting N=1), normalized them by means of the subroutine **norm**, computed the tensor product of all possible permutations (12,13,23,123), check if they were still normalized, computed the correspondent density matrices, checked if the density matrix of state 13 were equal to the partial trace over system 2 of the density matrix of system 123 [4], checked the same thing for system 12 and partial trace over system 2, printed the density matrix of state 1 and the partial trace over system 2 of the density matrix of system 12. In the end, we have computed Von Neumann entropy for a generic pure state with D=2 and N=10. The results are shown in bash code below.

```
Insert number of sybsystems N
10
Insert their dimension D
2
 Do you want to start the debugging mode?[y]/[n]
y
 ==================
 Debugging mode: ON
 ==================
 Do you want  to print checkpoints?[y]/[n]
y
 ==================================
```

[4] This must hold since the system is separable by construction.

```
 Check dimensions: OK
 ================================
 ================================
 Check dimensions: OK
 ================================
 Allocation: SUCCEEDED
 Allocation: SUCCEEDED
 Allocation: SUCCEEDED
 Allocation: SUCCEEDED
 Norm of tensor product 12:   1.0000000000000002
 Allocation: SUCCEEDED
 Norm of tensor product 13:   1.0000000000000000
 Allocation: SUCCEEDED
 Norm of tensor product 23:  0.99999999999999989
 Allocation: SUCCEEDED
 Norm of tensor product 123:   1.0000000000000000
 Allocation: SUCCEEDED
 Allocation: SUCCEEDED
 Allocation: SUCCEEDED
 Allocation: SUCCEEDED
 Allocation: SUCCEEDED
 Allocation: SUCCEEDED
 Check equality: OK
 Allocation: SUCCEEDED
 Check equality: OK
 Density matrix of state 1:
(0.59600349081498161,0.0000000000000000)     (0.48924132290324918,-3.77658273471739581E-002)
(0.48924132290324918,3.77658273471739581E-002)      (0.40399650918501867,0.0000000000000000)
 Reduced denisty matrix of state 12 over state2:
(0.59600349081498161,0.0000000000000000)       (0.48924132290324929,-3.77658273471739511E-002)
(0.48924132290324929,3.77658273471739511E-002)       (0.40399650918501873,0.0000000000000000)
 Allocation: SUCCEEDED
 Allocation: SUCCEEDED
 Von Neumann entropy of D^N pure state   1.5102679839488443E-014
 Total number of errors:           0
Total computation time (s): 1.1114375591278076
```

We see that this task was performed without errors, as we had verified by putting several checkpoints inside the main program. However, many trials were later performed by increasing the dimension D, resulting all successful.

Self Evaluation

This week we have learned how to deal with many-body quantum systems and how to implement tensor operations, such as index contraction. We have also verified if the system is separable, partial trace commutes with tensor product.
A few thing can be done to improve the code. For example, we could write a subroutine to compute the representation of a generic N-particle separable state in $\mathcal{H}$ by looping on states and calculating the tensor product at each step.
Moreover, we may as well write more subroutines to treat density matrices, such as Schmidt decomposition, mixed states, Entanglement's measures, generalized evolutions and so on and so forth.