Quantum information and computation: homework 8
of Tommaso Tabarelli

Abstract

In this homework we are asked to create a wave function of a system made by N subsystems analyzing two different cases: interacting and non interacting subsystems. Furthermore, fixing $N = 2$ we are asked to write the density matrix of a generic pure state and to reduce it both using "left" or "right" subsystem.

Theory

The generic wave function of a system composed by N particles, each of them $\in \mathscr{H} = \mathbb{C}^d$ is a vector of d^N components $\psi_{\alpha_1 \dots \alpha_N}$. Indeed, given

$$|\alpha_i\rangle = |1\rangle|2\rangle \dots |N\rangle \tag{1}$$

which is a base for the single subsystem i, the total wave function can be written as:

$$|\Psi\rangle = \sum_{\vec{\alpha}} \psi_{\alpha_1 \dots \alpha_N} |\alpha_1\rangle \dots |\alpha_N\rangle \tag{2}$$

In the case of non interacting subsystems (usually taken into account when dealing with a *mean field* approximation), the wave function can be re-written as a separable state, thus as the tensor product of the single subsystem wave functions:

$$|\Psi_{MF}\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \dots \otimes |\psi_N\rangle \tag{3}$$

and in this case only $d \cdot N$ coefficients instead of d^N are needed to describe the whole system. Considering for simplicity the subsystems to be all equal, we can write:

$$|\psi_i\rangle = |\psi\rangle = \sum_{j=1}^{d} A_j |j\rangle \tag{4}$$

and thus eq. 3 can be re-written as:

$$|\Psi_{MF}\rangle = \prod_{k=1}^{N} \left(\sum_{j=1}^{d} A_j |j^{(k)}\rangle \right) \tag{5}$$

In this case the information is "compressed" in only $d \cdot N$ coefficients.

The density matrix of a system is defined as the external product between the system wave function and its adjoint:

$$\rho = |\psi\rangle\langle\psi| \tag{6}$$

Its main properties are:

- it is *Hermitian*, thus $\rho = \rho^\dagger$
- if $|\psi\rangle$ is normalized, ρ's trace is 1:

$$\mathrm{Tr}(\rho) = \sum_m \psi_m^* \psi_m = \sum_m |\psi_m|^2 = 1 \tag{7}$$

To trace out a subsystem one has to take the trace over the states of the considered subsystem. For example, say we want to trace out the subsystem A from a generic system; calling B the remaining system after having traced out the subsystem A, it holds:

$$(\rho_B)_{mn} = \sum_i \langle i_A | \langle m_B | \rho | n_B \rangle | i_A \rangle \tag{8}$$

In the case of a separable system, when tracing out a subsystem, the remaining density matrix should correspond to the reduced density matrix obtained from the tensor product of the remaining subsystems.

Code development

The Fortran program reads the number of subsystems and the dimension of them from two files and checks that they are reasonable (i.e. greater than 0).

To evaluate the function for a generic system (thus it can be a non separable one) the choice was to initialize all the d^N complex values using uniform random numbers in the interval $[-1,1]$ both for real and imaginary parts, then divide them using the norm of the wave function itself so that it is normalized.

Afer that, the separable case is considered and evaluated. Since the wave function of the separable state can be determined using only the coefficients of the subsystems wave functions, the choice was to inizialize only them, thus storing only $d \cdot N$ values. They were inizialized using uniform random numbers in the interval $[-1,1]$ and every subsystem wave function was normalized (every wave function has to represent a subsystem).

```
DO ii=0,num_par-1
   DO jj=1,dim_
      CALL RANDOM_NUMBER(real_temp)
      CALL RANDOM_NUMBER(comp_temp)
      psi_sep_temp(ii*dim_+jj) = CMPLX(2*real_temp-1.,2*comp_temp-1)
   END DO
   ii_min = ii*dim_+1
   ii_max = ii*num_par+jj
   ! Normalizing every single particle wave function
   psi_sep_temp(ii_min:ii_max) = psi_sep_temp(ii_min:ii_max) /
       SQRT(SUM(ZABS(psi_sep_temp(ii_min:ii_max)**2)))

   WRITE(*,*) SQRT(SUM(ZABS(psi_sep_temp(ii_min:ii_max)**2)))
END DO
```

The next step was to evaluate the density matrices for both cases. The density matrix was evaluated using the respective system wave function and acting an external product between them and their adjoint. The main issue here was the evaluation of the separable system wave function. It was carried out using the fact that every "coefficient index" in the system wave function can be coded as a number written using N digits in base d. Using this trick, all wave function coefficients were properly accessed and initialized and the density matrix evaluation was then carried out.

```
! Bulding psi (the complete one)
DO ii=1,dim_**num_par
```

```
   ! The "code" should give the result "0" (all indexes to 0) at beginning, not at
      the end
   !  so here one has to start from 0, not from 1
   res_t = ii-1
   psi_sep(ii) = (1.,0.)

   DO jj=1,num_par
      rem_t = MOD(res_t,dim_)

      ! Evaluating the product using this trick
      psi_sep(ii) = psi_sep(ii) * psi_sep_temp(dim_*(num_par - jj) + rem_t + 1)
      ! Updating the number to divide
      res_t = (res_t-rem_t)/dim_
   END DO

END DO

! SEPARABLE density matrix
DO ii=1,dim_**num_par
   DO jj=1,dim_**num_par
      rho_sep(ii,jj) = psi_sep(jj)*CONJG(psi_sep(ii))
   END DO
END DO
```

Both density matrices were checked to be Hermitian. Then, ρ^2 matrix was evaluated in both cases and it was checked to have trace equal to 1 (within a certain precision).

As final task the procedure to evaluate a reduced matrix was written. Since it is hard to generalize for $N > 2$ particles, the choice was to implement that procedure for the case $N = 2$. The particle to trace out can be selected by setting *particle* parameter equal to 1 or 2.

```
! 1 <= Particle <= num_par: selecting 1 particle
!  In our case, the last particle in psi_sep_temp is that changing factor at every
    "step" in the matrix
particle = 1

! Looping over the reduced matrix, building it
DO ii=0,dim_**(num_par-1)-1
   DO jj=0,dim_**(num_par-1)-1
   ! Building the reduced matrix directly
   DO ii_red=0,dim_-1
      index_i = ii*dim_**(particle-1)+ii_red*dim_**(num_par-particle) + 1
      index_j = jj*dim_**(particle-1)+ii_red*dim_**(num_par-particle) + 1
      rho_red(ii+1,jj+1) = rho_red(ii+1,jj+1) + rho_sep(index_i,index_j)
      END DO
   END DO
END DO
```

Results

A test on the number of particles N was done, incrementing it untill the program could not work anymore due to memory allocation errors. In my case the maximum working N turned out to be 28 (for $d = 2$). Indeed, the required memory allocation for a 2^{28} vector of *COMPLEX*16* elements is, in terms of bytes: $2^{28} \cdot 16 \simeq 4,3$ GB, while for 2^{29} elements the required memory is $2^{29} \cdot 16 \simeq 8,6$ GB, which is larger than my laptop memory (which is about 8 GB).

All wave function norms are tested to be 1, trace of every density matrix is verified to be 1: all tests are succesful. Reduced density matrix for one particle is evaluated reducing the system density matrix (*particle*) parameter is hardcoded); it is then compared with the density matrix of the corresponding subsystem evaluated starting from the wave function. They are also printed to files and reported here as example. They turn out to be equal, so the check is succesful.

The evaluation of the reduced matrix using the implemented procedure and starting from the whole system matrix gives the following result (the values represent the real and imaginary part of the matrix elements):

```
 0.62692454   0.00000000    -0.45100212   0.17460598
-0.45100212  -0.17460598     0.37307546   0.00000000
```

the expected reduced matrix, evaluated using the subsystem wave function, is the following:

```
 0.62692454   0.00000000    -0.45100212   0.17460598
-0.45100212  -0.17460598     0.37307546   0.00000000
```

as one can notice, they have the same values so the tracing procedure seems to be implemented correctly.

Self-evaluation

In this task I learned how to generate wave functions and density matrices for generic systems and subsystems. Furthermore I learned how to evaluate a separable system density matrix using subsystems wave functions. Moreover, I learned how to create density matrices starting from wave functions and how to evaluate a reduced density matrix when $N = 2$.

Correctness

My program does what is expected to do. As aforementioned, many checks are done to be sure the program is doing what it is expected to do.

There are no compilation problems neither memory allocation issues (a special flag is used to compile in order to check for runtime memory errors).

Pre- and post-conditions and checkpoints are not used so much due to the large amount of time they require to be implemented (compared to the time we have to do the task).

Stability

Loops are made only on integers (usually the extreme values of the loop variables are the same used to initialize the dimension of the objects).

Maybe some memory error can arise if the space dimension d or the number of particles N (or even both parameters) become too large due to the fact that the matrix dimension goes as d^{2N}.

Accurate discretization

The only thing to care of is the choice of the random initialization of the wave functinos: as aforementioned, in this case the choice was to use uniform random values in the [-1,1] interval.

Flexibility

I tried to comment as much as possible the code so to have a clear and understandable code.

Efficiency

All routines are implemented by hand. They may be improved.

For what concerns memory and time usage, they depend on the parameters passed.

My work can be improved: for example, *particle* parameters can be taken as input in Fortran program. Moreover, the routine used to reduce the matrix is not general: it can be generalized but it would require more deeper studies and analysis.