Density matrices

Abstract

In this assignment, density matrices are studied, and a set of functions for handling them is developed. Particular care is devoted to analyzing the difference between separable and not separable pure states, especially regarding their computational characteristics.

Theory

Representation of N body wave functions

Given a quantum system, composed of N subsystems, each of them described by a wave function $\psi_i \in \mathcal{H} = \mathbb{C}^d$, a d dimensional Hilbert space, the total wave function $\Psi(\psi_1, \cdots, \psi_N)$ can be written as

$$|\Psi\rangle = \sum_{\vec{\alpha}} \Psi_{\alpha_1,\alpha_2,\cdots,\alpha_N} |\alpha_1\rangle |\alpha_2\rangle \cdots |\alpha_N\rangle \tag{1}$$

where the sum runs over all configurations, $\Psi_{\alpha_1,\alpha_2,\cdots,\alpha_N}$ is a tensor of rank N, and each $|\alpha_i\rangle = |j\rangle$, with $j \in [1, \cdots, N]$. Indeed, each combination $|\alpha_1\rangle |\alpha_2\rangle \cdots |\alpha_N\rangle$ is understood to be a tensor product.

Such a wave function can be effectively written using a base $|n\rangle$ of the resulting Hilbert space $\mathcal{H}^{\otimes N}$, where

$$n = \sum_{k=1}^{N} i_k d^{k-1}, \tag{2}$$

where i_k is the k-th subsystem base's chosen element. This is effectively a base-d encoding: with N figures in base d, one can represent d^N numbers, which are exactly the number of combinations needed for such a composite subsystem.

A similar idea can be used in the context of separable states, and be extended to the coefficients. Indeed, if a state is separable, it can be fully encoded in a $d \times N$ matrix, where each column contains the d coefficients pertaining to each element of that subsystem's base. For a separable state, $|\Psi_{AB}\rangle = |\psi_A\rangle \otimes |\psi_B\rangle$. In the base-$d$ notation above, the j-th coefficient C_j can be written as

$$C_j = \prod_{k=1}^{d} c_k \left(\frac{j}{d^{N-k}} \mod d \right), \tag{3}$$

where $c_k(j)$ denotes the k-th subsystem element at position j (the ordering of the subsystems is from left to right, so that in, say, $\mathcal{H}_{AB} = \mathcal{H}_A \otimes \mathcal{H}_B$, A has index 1).

Density matrices

In the following, unless otherwise specified, the states are assumed to be pure. Given one such state $|\psi\rangle$, its density matrix is defined as

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

The density matrix of a pure state must have unitary trace, $\mathrm{Tr}(\rho) = 1$.

Density matrices can also be defined for composed systems. Suppose that a system is composed by two subsytems A and B; to each of them corresponds a Hilbert space, $\mathcal{H}_A$ and $\mathcal{H}_B$ respectively. The composed system AB will then be represented by the tensor product of the composing Hilbert spaces,

$$\mathcal{H}_{AB} = \mathcal{H}_A \otimes \mathcal{H}_B \tag{5}$$

and a state in this Hilbert space is denoted as $|\Psi\rangle$. The density matrix of the whole system is then $\rho_{AB} = |\Psi\rangle\langle\Psi|$.

If the state is separable, then the density matrix can be written as

$$\rho_{AB} = \sum_i p_i \left(\rho_i^A \otimes \rho_i^B\right) \tag{6}$$

In this case, one can retrieve the density matrix of either subsystem by performing a partial trace:

$$\rho_A = \mathrm{Tr}_B(\rho_{AB}) = \sum_\alpha \langle\alpha|_B \, \rho_{AB} \, |\alpha\rangle_B \tag{7}$$

where α runs over the possible states that subsystem B can take, and similarly,

$$\rho_B = \mathrm{Tr}_A(\rho_{AB}) = \sum_\alpha \langle\alpha|_A \, \rho_{AB} \, |\alpha\rangle_A \tag{8}$$

where α runs over the possible states that subsystem A can take. In this notation, the part of ρ_{AB} that pertains to the subsystems that are not traced over is left untouched.

Computationally, the partial trace operation is not straightforward. Indeed, even representing the matrix is not trivial. Depending on the dimension d of the Hilbert spaces one works on, and on the number of subsystems N that are considered, representing the total density matrix requires $(d^N)^2$ elements.

Code development

The code is organized as follows:

- `utils.f90`: a library of functions and subroutine that is extensively used in the actual source codes. It contains the partial trace function, the function for reconstructing a separable state from its 'compressed' matrix version and an implementation of Kronecker product (useful for testing)
- `debug_module.f90`: a debugging module, extensively used for performing checks on traces and norms
- `ex8_1.f90`: a program in which a separable and non separable states are created
- `ex8_2.f90`: a program in which the partial trace function is tested in different combinations of d and N

The most critical part of the whole assignment was the development of a partial trace function that could work for general values of d and N. Indeed, if one limits himself to $N = 2$, the generalization from $d = 2$ to an arbitrary value is straightforward. The partial trace can only be computed in N ways, so changing d only ends up in changing the number of summands that enter the computation of each element of the resulting reduced trace.

For an N-body system with dimension d, the density matrix has shape (d^N, d^N). Tracing out one subsystem results in a reduced matrix of shape (d^{N-1}, d^{N-1}), where each of its elements is the sum of d elements of the starting matrix. Indeed, the problem of computing these sums is finding which elements to sum. This problem can be summed up as:

- finding the starting indexes for each sum
- finding the 'jumps' from one element to the next in the sum

Both these quantities are functions of d, N and the index of the subsystem to trace out, i. The trace operation has an intrinsic symmetry with respect to the principal diagonal. This means that the wanted starting indexes can be characterized as elements of $\{idx\} \times \{idx\}$, where $\times$ denotes a cartesian product. The set $\{idx\}$ is composed of those numbers that identify the possible starting indexes, given d, N and i. The dependence of such set on i comes from the base-d representation highlighted above. Indeed,

$$\{idx\} = \left\{ j \in [0, d^{N-1} - 1] \quad \text{such that} \quad \frac{j}{d^{N-i}} \quad \text{mod } d = 0 \right\} \tag{9}$$

In the code, all the indexes were increased by 1 in order to follow Fortran's index conventions. As for the 'jumps', they are simply characterized as d^{N-i}. This, once again, comes from the base-d representation.

The resulting algorithm is general, but inefficient: it relies on a matrix representation that, being exponential in N, can easily become unmanageable. Indeed, any algorithm that explicitly computes all the elements of the density matrix should be deemed inefficient.

The code for the partial trace is shown below.

```fortran
  FUNCTION TRACE_OUT(idx,mat,dd,NN)RESULT(red_dmat)
!!$  -------------------------------------------------------
!!$    This func traces out the (idx)-th subsystem, where
!!$    TOT=S_1 X S_2 X S_2 X ... X S_NN,
!!$    and returns a reduced density matrix.
!!$    ARGUMENTS
!!$    - idx    : an INTEGER*4 number, index of subsyst. to trace out
!!$    - mat    : a COMPLEX*16 matrix, shape (dd**NN) x (dd**NN)
!!$    - dd     : an INTEGER*4 number, dimension of Hilbert spaces
!!$    - NN     : an INTEGER*4 number, number of subsystems
!!$    RETURNS
!!$    - red_dmat   : a COMPLEX*16 matrix, shape dd**(NN-1) x dd**(NN-1)
!!$  -------------------------------------------------------
    INTEGER*4 :: idx, dd, NN, ii, jj, kk, jump, cnt
    COMPLEX*16, DIMENSION(:,:) :: mat
    INTEGER*4, DIMENSION(SIZE(mat,1)/dd) :: start_idx,range
    COMPLEX*16, DIMENSION(SIZE(mat,1)/dd,SIZE(mat,2)/dd) :: red_dmat

    cnt=0
    DO ii=0,dd**NN-1
       IF (MOD(ii/(dd**(NN-idx)),dd).EQ.0) THEN
          cnt=cnt+1
          start_idx(cnt)=ii
       END IF
    END DO
    start_idx=start_idx+1
    jump=dd**(NN-idx)
    red_dmat=0.d0
    DO ii=1,SIZE(mat,1)/dd
       DO jj=1,SIZE(mat,2)/dd
          DO kk=0,dd-1
             red_dmat(ii,jj)=red_dmat(ii,jj)+&
                  mat(start_idx(ii)+kk*jump,start_idx(jj)+kk*jump)
          END DO
       END DO
    END DO
    RETURN
  END FUNCTION TRACE_OUT
```

Results

The programs were used in order to test the functions. As for the first part, concerning the efficiency of the representations of both separable and not separable, a simple test was carried out. The number of subsystems was increased up until the program crashed. This happened (in the considered machine) when $d = 2$ and $N = 18$. A segmentation fault error was issued for this value, with no additional information. This behavior was repeated also for all the Ns up to 28 (with increasingly long waiting times before the segmentation fault error appeared). When $N \geq 28$, an operating system error was issued, lamenting excessive memory requirements.

As for the second part, different separable states were described, and then composed using the Kronecker product function. Separable states were chosen because the results are easily checked. The partial trace was capable of retrieving the starting density matrices, deeming it successful. An example follows for $d = 2$, $N = 2$; more can be found in the `results.txt` file.

```
FULL DENSITY MATRIX:
               (0.5000000000000000,0.0000000000000000)            (0.5000000000000000,0.0000000000000000)
              ↪   (0.0000000000000000,0.0000000000000000)             (0.0000000000000000,0.0000000000000000)
               (0.5000000000000000,0.0000000000000000)            (0.5000000000000000,0.0000000000000000)
              ↪   (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)
REDUCED DENSITY MATRIX FOR THE FIRST SUBSYSTEM
               (0.5000000000000000,0.0000000000000000)            (0.5000000000000000,0.0000000000000000)
               (0.5000000000000000,0.0000000000000000)            (0.5000000000000000,0.0000000000000000)
REDUCED DENSITY MATRIX FOR THE SECOND SUBSYSTEM
```

```
        (1.0000000000000000,0.0000000000000000)        (0.0000000000000000,0.0000000000000000)
        (0.0000000000000000,0.0000000000000000)        (0.0000000000000000,0.0000000000000000)
```

Comments and self evaluation

One of the main advantages of dealing with separable states is that, potentially, one only needs to store a $d \times N$ matrix to have full information. Indeed, this removes the need of storing a matrix of shape (d^N, d^N); if any element was to be needed, it could be computed on the spot. As such, the achieved compression, memory efficiency and computational efficiency turn an exponentially complex problem in a linear one.

Another possible improvement would be avoiding the computation of the whole density matrix when extracting a partial trace. This would require to map indexes to sums of products of the elements of a state.

The assignment proved challenging in that generalizing the partial trace was quite hard, but was indeed interesting, especially in the emergence of fractal-like behavior in the construction of the density matrix.