# Assignment 6

## Quantum Information & Computing

**Francesco Barone**

University of Padua, Department of Physics

**December 13, 2022**

# Theory

Let us consider a pure $N$-body quantum system. Each subsystem can be described in a $D$-dimentional Hilbert space using a (single-body) wavefunction $|\psi\rangle = \Sigma_j^D \alpha_j |\alpha_j\rangle \in \mathcal{H}^D$.

The $N$-body quantum system is defined in a Hilbert space of dimension $D^N$. Its most **general representation** is written as

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

Another way to picture a quantum state is the **density matrix** representation, $\rho = |\Psi\rangle\langle\Psi|$ which is crucial when describing mixed states.

**If the system is separable**, meaning that the interactions between the components of the system can be neglected, we shall write the global wavefunction as the tensor product of single-body wavefunctions:

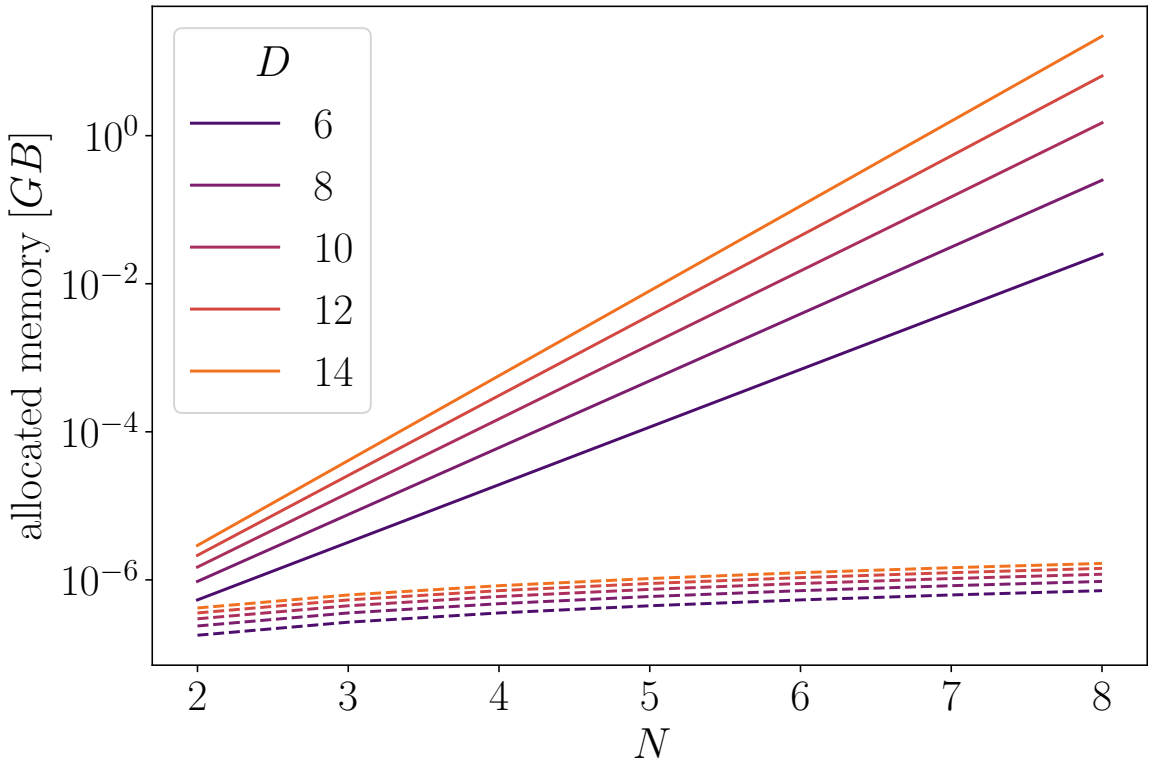
$$|\Psi\rangle_{sep} = \bigotimes_i^N |\psi_i\rangle = \left(\sum_j^D \alpha_j |\alpha_j\rangle\right)_1 \otimes \dots \otimes \left(\sum_j^D \alpha_j |\alpha_j\rangle\right)_N \tag{2}$$

# Memory allocation

The different representations for separable and non-separable states have **huge implications by the point of view of computational complexity**.

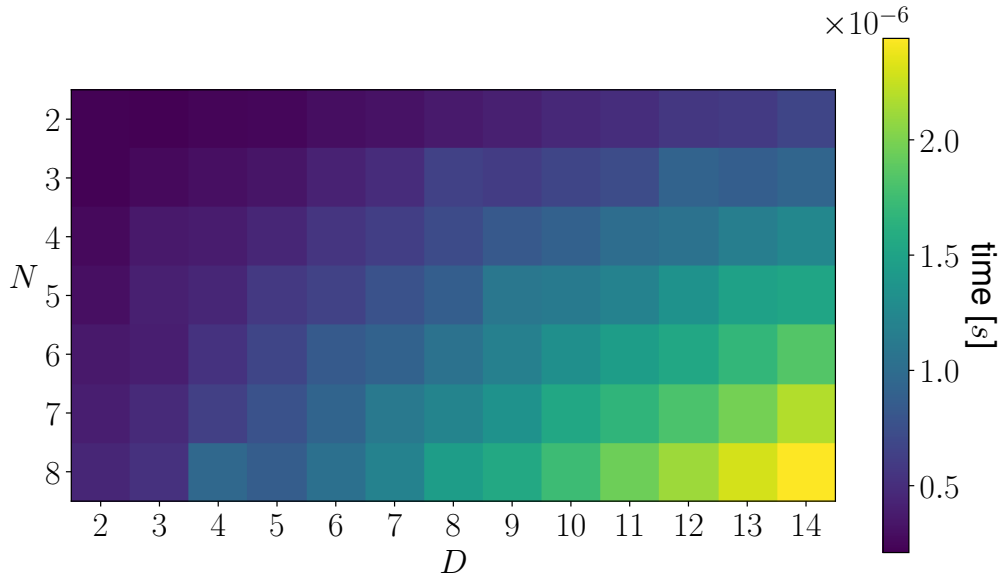
The first representation (non-separable state) requires the allocation of complex numbers which scales exponentially with the number of subsystems. The latter (separable state), scales linearly.

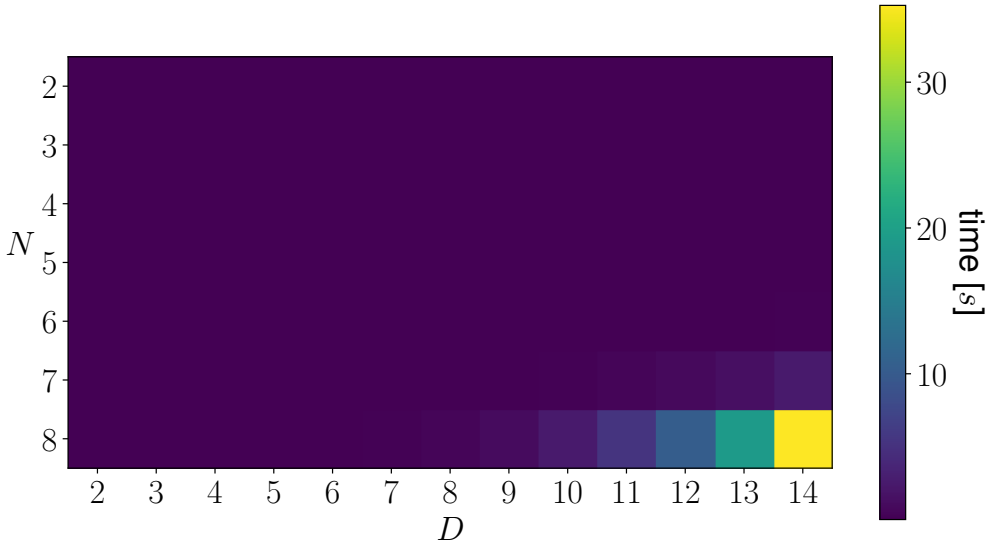
This plot shows the **allocated memory** for sets of qunatum systems with dimension $D$, for both separable states (dashed line) and general states (solid line).



# Malloc benchmarks

I have implemented both the representations of $|\Psi\rangle$ in FORTRAN, benchmarking the CPU time required to initialize the vector with random values such that $\langle\Psi|\Psi\rangle = 1$.




We observe that the allocation of a **separable state** $\sim \mathcal{O}(D \cdot N)$ takes a negligible amount of time. To make this benchmark more robust against statistical variations, each vector has been allocated $100$ times, taking the average time.

Instead, to allocate a **generic state** vector, the CPU time scales exponentially $\sim \mathcal{O}(D^N)$. It is really easy to fill up the available RAM. My top benchmark has allocated an array of $14^8$ double complex values ($\simeq 22$ GB).

# Partial trace

Let us recall that the density matrix of a $N$-body quantum system is defined as $\rho = |\Psi\rangle\langle\Psi|$.

In many applications it is crucial to compute the density matrix of one (or many) particular subsystem(s). This operation is called **partial trace**. For instance, if we consider a two-qubit system (A, B), one could compute the **reduced density matrix** on each subsystem as

$$\rho_A = Tr_B[\rho] = \sum_\beta \langle\beta|\rho|\beta\rangle \qquad \rho_B = Tr_A[\rho] = \sum_\alpha \langle\alpha|\rho|\alpha\rangle$$

More generally, we can consider A and B to be many-body quantum systems themselves:

$$\{S_1, S_2, ..., S_N\} \longrightarrow \{\tilde{A}, \tilde{B}\} \quad \text{with} \quad \tilde{A} = \{S_1, ..., S_i\} \in \mathcal{H}^{D^i}, \tilde{B} = \{S_i, ..., S_N\} \in \mathcal{H}^{D^{(N-i)}}$$

As shown in this paper, the partial traces over two quantum systems of generic size $D_A$ and $D_B$ are written in matrix-loop notation as

$$[\rho_A]_{ij} = \sum_{k=1}^{D_B} \rho[(i-1)D_B + k, (j-1)D_B + k] \qquad [\rho_B]_{ij} = \sum_{k=1}^{D_A} \rho[(k-1)D_A + i, (k-1)D_A + j]$$

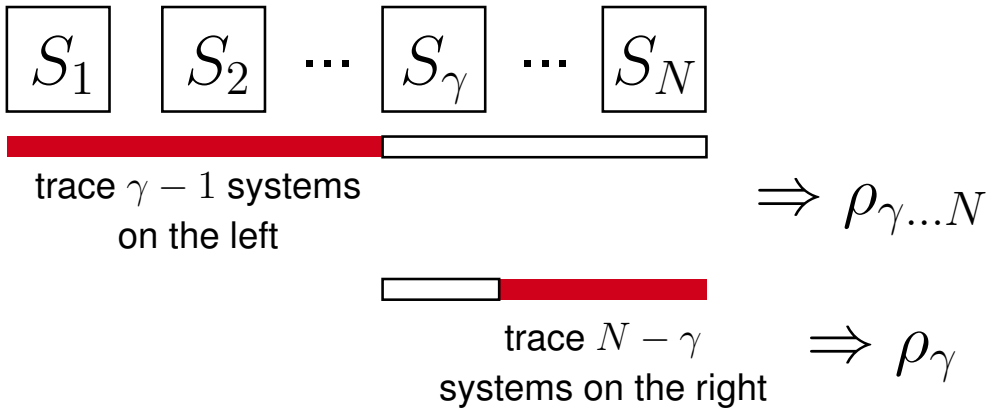
# **Partial trace** // Reduced matrix on a specific subsystem out of $N \neq 2$

The formulas shown in the previous slide are trivial to implement with nested loops*:

```
trdm%data = (0.d0,0.d0)
do ii = 1, lss
do jj = 1, lss
do kk = 1, rss
    trdm%data(ii,jj) = trdm%data(ii,jj) &
     + dm%data( rss*(ii-1) + kk, rss*(jj-1) + kk)
enddo
enddo
enddo
```

```
trdm%data = (0.d0,0.d0)
do ii = 1, rss
do jj = 1, rss
do kk = 1, lss
    trdm%data(ii,jj) = trdm%data(ii,jj) &
     + dm%data( rss*(kk-1) + ii, rss*(kk-1) + jj)
enddo
enddo
enddo
```

Once we know how to left- and right-trace **groups of systems** out of a $N$-body system, it is straightforward to implement a routine which computes the reduced density matrix for a specific subsystem $\gamma$, $1 \leq \gamma \leq N$:




$*$ = I am not a huge fan of this approach. A professional library like Pennylane uses optimized `einsum()` routines, which are not straightforward to be coded in FORTRAN.

# Tests on partial trace

To probe my implementation of the partial trace, I have tested it at first on simple 2-qubit states:

- $|\psi\rangle = |00\rangle$

$$\Rightarrow \rho_0 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \ , \quad \rho_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

- $|\psi\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle)$ (Bell state)

$$\Rightarrow \rho_0 = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix} \ , \quad \rho_1 = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}$$

To do further tests, I have considered **randomly generated density matrices**. In particular, random density matrices for $N = 5$ and $N = 10$ qubits have been generated with **Qiskit** (Hilbert-Schmidt method). I omit to show in this presentation the two random density matrices.

To **test the correctness of my partial trace function**, I have compared the partial trace computed in FORTRAN with the result from Qiskit's partial trace function.

The whole processing of (1) data generation in Python, (2) computing the partial trace in FORTRAN and (3) evaluating the correctness back in Python is made automatic by the Bash script `test_runme.sh`.