Quantum Information and Computing
Assignement 6 Report

Andrea Turci

November 2024

Density Matrix

1 Introduction

This report explores the construction, analysis, and validation of quantum states in multi-particle systems. At the heart of this assignemetn lies the challenge of accurately building quantum states, computing density matrices, and analyzing subsystems, which are essential concepts for studying quantum many-body systems, highlighting the role between separable and entangled states.
The theoretical framework highlights the distinction between separable states—where each subsystem operates independently—and general states, which can exhibit entanglement. The density matrix plays a central role in this analysis, enabling the description of both the entire system and its individual subsystems. By tracing out parts of the system, we gain insight into how information and correlations are distributed across the quantum system.
This report includes a detailed series of tests designed to validate the implementation. These tests cover a range of cases, from simple separable states like $|\psi\rangle = |0\rangle|0\rangle$ to the Bell state $(|00\rangle + |11\rangle)/\sqrt{2}$, which demonstrates the unique properties of entanglement. By comparing the outputs of our code with theoretical expectations, this assignement ensures the correctness and robustness of the implementation.

2 Theoretical Framework

In quantum mechanics, the density matrix provides a powerful tool for describing the state of quantum systems, especially when dealing with mixed states or entangled systems. To understand this framework, we need to start the description with the analysis of composite quantum systems, the properties of the tensor product, the general formulation of quantum states, and finally, the density matrix itself.

Composite Quantum Systems and Hilbert Spaces

In quantum mechanics, a system with multiple subsystems is described by a composite Hilbert space. If a system is composed of N subsystems, where each subsystem has its own Hilbert space $\mathcal{H}_i$ of dimension D, the total system lives in the tensor product space:

$$\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_N.$$

The tensor product allows us to mathematically combine the states of individual subsystems into a single state that represents the entire system. For two subsystems, for example, if $|\alpha\rangle \in \mathcal{H}_1$ and $|\beta\rangle \in \mathcal{H}_2$, their joint state is written as:

$$|\psi\rangle = |\alpha\rangle \otimes |\beta\rangle,$$

and is an element of the composite space $\mathcal{H}$. The dimension of the composite space grows exponentially with the number of subsystems:

$$\dim(\mathcal{H}) = (\dim(\mathcal{H}_1)) \times (\dim(\mathcal{H}_2)) \times \cdots \times (\dim(\mathcal{H}_N)).$$

This exponential growth reflects the vast potential of quantum systems to encode and process information, but it also presents computational challenges when simulating quantum systems, as the size of the Hilbert space is no longer be attacked for large N.
An important feature of the tensor product structure is its ability to encode entanglement. Entangled states are those that cannot be written as a simple tensor product of individual states. For example, the state:

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

is an entangled state of two qubits, as it cannot be decomposed into a product of states in $\mathcal{H}_1$ and $\mathcal{H}_2$. In contrast, a separable state such as:

$$|\psi\rangle = |0\rangle \otimes |1\rangle,$$

describes two uncorrelated subsystems.

Properties of the Tensor Product

The tensor product satisfies several important mathematical properties that are crucial for its role in quantum mechanics. First, it distributes over scalar multiplication:

$$c(|\alpha\rangle \otimes |\beta\rangle) = (c|\alpha\rangle) \otimes |\beta\rangle = |\alpha\rangle \otimes (c|\beta\rangle),$$

where $c \in C$ is a scalar. This ensures consistency in scaling individual states.

Second, it is distributive over addition. For any states $|\alpha_1\rangle, |\alpha_2\rangle \in \mathcal{H}_1$ and $|\beta\rangle \in \mathcal{H}_2$:

$$(|\alpha_1\rangle + |\alpha_2\rangle) \otimes |\beta\rangle = (|\alpha_1\rangle \otimes |\beta\rangle) + (|\alpha_2\rangle \otimes |\beta\rangle).$$

Similarly, for states $|\beta_1\rangle, |\beta_2\rangle \in \mathcal{H}_2$ and $|\alpha\rangle \in \mathcal{H}_1$:

$$|\alpha\rangle \otimes (|\beta_1\rangle + |\beta_2\rangle) = (|\alpha\rangle \otimes |\beta_1\rangle) + (|\alpha\rangle \otimes |\beta_2\rangle).$$

These properties allow for the mathematical manipulation and decomposition of quantum states into simpler components. They also form the foundation for understanding quantum entanglement and the superposition principle in composite systems.

Quantum States: General and Separable

A general quantum state for a composite system is a linear superposition of the tensor products of basis states from each subsystem. If each subsystem is described by a basis $\{|j\rangle_i\}_{j=1}^{D}$, then the total system state is expressed as:

$$|\Psi\rangle = \sum_{\bar{j}} c_{\bar{j}} |j_1\rangle_1 \otimes |j_2\rangle_2 \otimes \cdots \otimes |j_N\rangle_N,$$

where $\bar{j} = (j_1, j_2, \ldots, j_N)$ represents the multi-index and $c_{\bar{j}}$ are the complex coefficients. The total number of coefficients required is D^N, reflecting the exponential growth in complexity.

A separable state, on the other hand, can be written as a product of wavefunctions for individual subsystems:

$$|\Psi\rangle_S = \left(\sum_{j_1} c_{j_1}|j_1\rangle_1\right) \otimes \left(\sum_{j_2} c_{j_2}|j_2\rangle_2\right) \otimes \cdots \otimes \left(\sum_{j_N} c_{j_N}|j_N\rangle_N\right).$$

Separable states are less computationally demanding, requiring only $N(2D-2)$ real parameters to describe, compared to the $2(D^N-2)$ needed for general states. This efficiency comes from the lack of correlations between subsystems.

The Density Matrix

The density matrix generalizes quantum states to both pure and mixed cases. For a pure state $|\psi\rangle$, the density matrix is:

$$\rho = |\psi\rangle\langle\psi|.$$

This encodes the probabilities and coherences of the state in a matrix form. Diagonal elements represent probabilities, while off-diagonal elements describe quantum superpositions. For example, if $|\psi\rangle = c_0|0\rangle + c_1|1\rangle$, the density matrix becomes:

$$\rho = \begin{bmatrix} |c_0|^2 & c_0 c_1^* \\ c_0^* c_1 & |c_1|^2 \end{bmatrix}.$$

For mixed states, the density matrix is a statistical mixture:

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|,$$

where p_i are probabilities.

When dealing with subsystems, we use the reduced density matrix to describe a specific part of the system. Given a composite density matrix ρ, the reduced density matrix of subsystem A is:

$$\rho_A = \mathrm{Tr}_B(\rho).$$

This involves tracing out the degrees of freedom of subsystem B:

$$\rho_A = \sum_j \langle j_B|\rho|j_B\rangle.$$

For an entangled state, this reduced density matrix often represents a mixed state, even if the global state is pure.

3 Methodology

The assignment has been developed using Python as language, due to it's flexibility for operations with matrices and vectors, performing the diagonalizations and solving the eigenvalue problems. In particular, several functions have been defined and split across different files according to their functionalities. For this reason, 3 files have been created:

- `functions.py`
- `rhos.py`

- `test.py`

I will now perform a description for each of these files focusing on the functions they contain that are the most relevant in the discussion (in particular, I will neglect the functions related to the plot of the results).
To better understand how these functions work, I first have to define which are the relevant parameters in our problem, which are going to be changed according to the different situations:

- `N`: This parameter represents the number of subsystems, or lattice sites, in the Quantum Many-Body system. Each subsystem is treated as an individual quantum unit, and the total Hilbert space of the system is the tensor product of these subsystems. Increasing `N` enlarges the total system size, exponentially increasing the Hilbert space dimension and, consequently, the computational complexity. The full dimension of the QMB system is given by $\texttt{D}^{\texttt{N}}$, where `D` is the local dimension of each subsystem.

- `D`: This parameter specifies the local dimension of each subsystem or lattice site, that is the dimension of the individual Hilbert space associated with one site. For example, `D = 2` corresponds to qubits, while larger values represent systems with more degrees of freedom per site. Here, `D = 10`, indicating a high-dimensional local Hilbert space. This affects both the complexity of the state vector ψ, which has $\texttt{D}^{\texttt{N}}$ components, and the size of operators like density matrices, which scale as $\texttt{D}^{\texttt{N}} \times \texttt{D}^{\texttt{N}}$.

- `seed`: This parameter sets the seed for the random number generator used in the simulation, so it's a crucial parameter for debugging and consistent analysis.

- `type`: This parameter defines the structure of the initial state of the QMB system. It can be initialized in either `"separable"`, meaning the the system is considered as a separable state, or `"general"`, for general wavefunctions, as treated in the threory section. This is crucial for the inizialization of the coefficients (and so of the wave function), since different funcions and subroutines need be be called.

Now let's describe the content of each Pythonic file.

3.1 rhos.py

1. `rho(state)` This function computes the density matrix for a given quantum state, which is a crucial step in this Assignement.
It takes as input a quantum state ψ, represented as a complex-valued array (`state`). Using $\rho = |\psi\rangle \langle\psi|$, the density matrix is calculated as the outer product of the state vector with its complex conjugate. In particular, the outer product is built using `np.outer`, which ensures that ρ is Hermitian.

2. `get_reduced_density_matrix` This function extracts the reduced density matrix for a subsystem of a Quantum Many-Body system, which, again as explained in the Theoretical Framework section, describes a subsystem by tracing out the degrees of freedom of the remaining environment.
This function checks that the provided inputs (ψ, `loc_dim`, and `n_sites`) are valid, ensuring proper data types and dimensional consistency. Then ψ is reshaped into a tensor format where each "leg" represents a lattice site, with dimensions determined by the local Hilbert space dimension (`loc_dim`). This is done because this reshaping reflects the structure of the many-body system, enabling easy identification of subsystem and environment indices.
At this point, the indices of the lattice sites to keep (`keep_indices`) are identified, and the remaining indices are treated as the environment, so that the tensor is reordered in sucha a way

that the subsystem indices are grouped together.
The tensor is reshaped into a 2D matrix where the subsystem indices define one dimension and the environment indices define the other, and so by contracting the environment dimension, the function eliminates the environment's degrees of freedom, yielding the reduced density matrix for the subsystem.

3.2 functions.py

1. `initialize_coefficients` This function is the starting point for defining the quantum state coefficients in either separable or general forms. It initializes the coefficients based on the desired structure of the system and handle both random and user-defined inputs.
When working with a separable state, the function creates a set of coefficients for each subsystem. If `random_init` is set to `True`, it generates random complex numbers for each subsystem, ensuring they are properly normalized. Otherwise, it uses `init_coeff` as the predefined coefficients.
Instead, for a general state, all subsystems are treated as a single composite system. The function generates coefficients for the full Hilbert space of dimension D^N, either randomly or using the provided `init_coeff`.
The key to this function is its flexibility: you can quickly switch between separable and general cases and decide whether to use random or custom coefficients. Once the coefficients are created, the function validates them using `validate_coefficients`, which is defined just below.

2. `validate_coefficients` This function ensures that the coefficients provided for a quantum state are correct and physically meaningful.
For separable states, it checks that the number of coefficient arrays matches the number of subsystems (N), and also checks that each array has the correct dimension (D) and is normalized.
Instead, for general states, it verifies that the total number of coefficients equals D^N (the dimension of the composite Hilbert space) and once again that they are properly normalized.
If any issue is detected, it raises a `ValueError` with a clear description of the problem.

3. `create_state` This function builds the full quantum state of a system composed of N subsystems, based on the initialized coefficients.
For a separable state, it starts with the first subsystem's coefficients and constructs a weighted sum of basis vectors. It then computes the tensor product of this subsystem with the states of the remaining subsystems, in the end forming the full quantum state. Instead, for a general state, the function iterates over all possible multi-indices $(j_1, j_2, ..., j_N)$ of the composite system. Each multi-index corresponds to a specific tensor product of basis vectors, which is weighted by the appropriate coefficient. These weighted tensor products are summed to construct the full state in the D^N-dimensional Hilbert space.

4. `comput_time` This function analyzes the computational cost of creating quantum states for varying system sizes. It measures both the time taken to construct the state and the memory required to store it.
it iterates over a range of subsystem counts, from N_{min} to N_{max} in units of N_{step}, and over a range of local dimensions, from D_{min} to D_{max} in units of D_{step}, creating states for each combination.
Then, for each configuration it initializes the coefficients randomly using `initialize_coefficients`, it constructs the full quantum state using `create_state`, and then it records the elapsed CPU time and the memory (in bytes) required for the state. The results are stored in two matrices: `cpu_times_matrix` and `bytes_matrix`.

3.3 test.py

This file serves as a testing subroutine for verifying the correctness of functions that handle quantum states in multi-particle systems. The primary objectives of this file are to ensure the State construction, for both separable and general states and also the Density Matrix calculations of both full and reduced density matrices.

The tests follow a structured approach:

1. First the wavefunction is generated, meaning that the coefficients for the quantum state are initialized using `initialize_coefficients`, and the full state is created using `create_state`.

2. Then a comparison with the expected results is performed; The manually computed expected wavefunction is compared against the generated wavefunction, and the same reasoning, with the same checks, is applied to both the full density matrices and reduced density matrices.

3. In the end a Validation feedback is provided, with the success that is indicated with a "Test passed" message, while failures provide detailed information, including actual and expected results for debugging.

In particular I performed a wide range of tests. I first focused on simple states, verifying that the basic state $|\psi\rangle = |0\rangle|0\rangle$ yielded the correct results. Then I applied the checks to a combination state, still separable, with the superposition, $|\psi\rangle = |0\rangle \otimes (|0\rangle + |1\rangle)/\sqrt{2}$, and also a separable state with balanced probabilities, $|\psi\rangle = (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle)/2$.
Then I moved to the general states, firstly testing a Bell state $|\psi\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ (entangled state), and then a state with only one non-zero component, $|\psi\rangle = |11\rangle$.

4 Results

Let us start analyzing the obtained results.
First of all, we applied the before mentioned code in order to highlight the differences between the separable state and the general N-body state, in terms of both efficiency and memory usage; finally we analyzed the correctness of the computation of the density matrices and reduced density matrices.
The parameters that are relevant to this discussion are N and D, which again are respectively the number of subsystems, or sites, of my N-body state and the dimension of each of those subsystems.
The coefficients have been randomly generated, in order to avoid to manually give them in input, and also to avoid biases in the computation times. The total wave function is generated and both the CPU time for this generation and the allocated memory have been computed, for both the case of separable state and general state. The results are yielded in the following Figures.

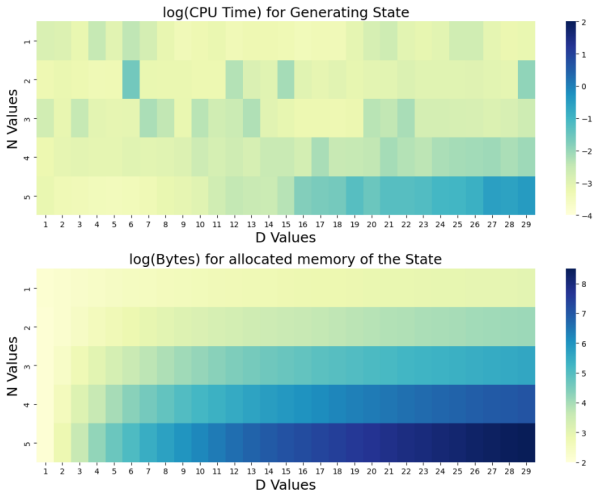

Figure 1: *CPU time and memory storage for Separable state*

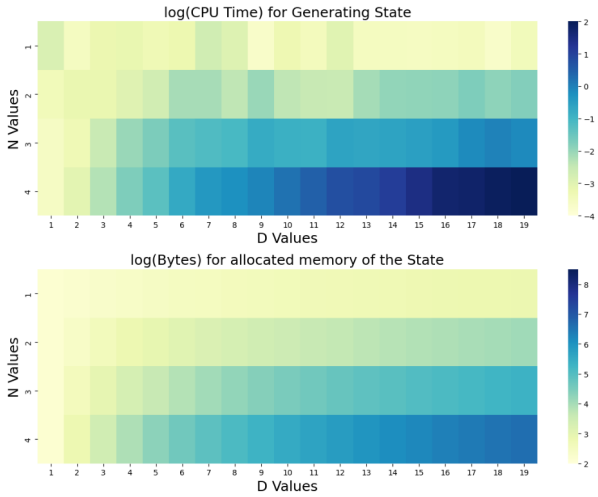

Figure 2: *CPU time and memory storage for General state*

The first thing to comment is that in the general case I had to restrict the ranges of the parameters N and D to take into consideration: keeping the same ranges of the separable case, the computation required more than 2 days, compared to the few seconds of the separable case; this highlights the improved efficiency of the separable case.
However, since we want to comment on some quantitative results, the "reduced" heatmap has been computed for the general state, where the legend values have been normalized in order to allow for a comparison. Also, the values are put in logarithmic scale.
From the CPU heatmaps one clearly sees that when the values of N and D start to increase, the difference between the two scenarios becomes of several orders of magnitudes.
The same considerations can be applied to the memory storage: as expected the memory storage increases when either one between N or D increases. Also in this case, the memory storage scales linearly with N and D in the separable case, and exponentially in the general case.
Afterwards, I moved to the computation of density matrices and reduced density matrices for different subsystems.
In particular, for the separable state scenario, I considered a state composed of $N = 2$ subsystems, as requested, each of them having dimension $D = 3$: I am considering qutrits. I compute the density matrix and the reduced density matrices for both subsystems, and check that they satisfy some basic properties of density matrices: they are hermitian and the trace is unitary.
I repeat the same procedure for the general state case, for arbitrary values of N and D.
The checkpoints that are performed are not enough to conclude the correctness of the code; for this reason, numerical tests with well known wave functions are considered, which validate the generation of quantum states, the computation of their corresponding density matrices, and the reduced density matrices of subsystems. Below, I detail the input states considered and the expected outcomes.

Separable State: Simple Case Input: A separable state $|\psi\rangle = |0\rangle|0\rangle$, with coefficients:

$$\text{Subsystem 1: } \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \text{Subsystem 2: } \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Output: The wavefunction in the computational basis is:

$$|\psi\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}.$$

The full density matrix is:

$$\rho = |\psi\rangle\langle\psi| = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The reduced density matrices for both subsystems are:

$$\rho_{\text{left}} = \rho_{\text{right}} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}.$$

Separable State: Superposition Case Input: A separable state $|\psi\rangle = |0\rangle \otimes (|0\rangle + |1\rangle)/\sqrt{2}$, with coefficients:

$$\text{Subsystem 1: } \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \text{Subsystem 2: } \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}.$$

Output: The wavefunction in the computational basis is:

$$|\psi\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \end{bmatrix}.$$

The full density matrix is:

$$\rho = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The reduced density matrices are:

$$\rho_{\text{left}} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad \rho_{\text{right}} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}.$$

General State: Bell State Input: An entangled Bell state $|\psi\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$, with coefficients:

$$\text{State: } \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}.$$

Output: The full density matrix is:

$$\rho = \begin{bmatrix} 0.5 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0.5 \end{bmatrix}.$$

The reduced density matrices are the same for both subsystems:

$$\rho_{\text{left}} = \rho_{\text{right}} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}.$$

General State: Single Non-Zero Component Input: A general state with a single non-zero component $|\psi\rangle = |11\rangle$, with coefficients:

$$\text{State: } \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}.$$

Output: The full density matrix is:

$$\rho = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The reduced density matrices are:

$$\rho_{\text{left}} = \rho_{\text{right}} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

Separable State: Equal Superposition Input: A separable state with an equal superposition for both subsystems:

$$|\psi\rangle = (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle)/2,$$

with coefficients:

$$\text{Subsystem 1: } \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad \text{Subsystem 2: } \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}.$$

Output: The wavefunction in the computational basis is:

$$|\psi\rangle = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}.$$

The full density matrix is:

$$\rho = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}.$$

The reduced density matrices for both subsystems are:

$$\rho_{\text{left}} = \rho_{\text{right}} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}.$$

The tests confirm that the functions correctly generate quantum states, compute their density matrices, and produce accurate reduced density matrices for both separable and general states. Each case was validated against manually computed expected results, demonstrating the correctness and reliability of the implementation.

5 Conclusions

The results demonstrate that the implemented methods are both reliable and insightful. For example, separable states such as $|\psi\rangle = |0\rangle|0\rangle$ were accurately represented, with wavefunctions and density matrices matching theoretical expectations. Reduced density matrices for subsystems were consistent with independent behavior, confirming that the code correctly handles separable quantum systems.
For more complex scenarios, such as the Bell state $(|00\rangle + |11\rangle)/\sqrt{2}$, the implementation successfully computed the presence of entanglement. The reduced density matrices of the subsystems were maximally mixed, as expected for strongly entangled states.
This assignement also analyzed the computational effort of these methods. Separable states demonstrated efficient scaling, with computational requirements growing linearly with the number of subsystems. However, for general states like the Bell state, the cost in both memory and computation time increased exponentially with system size. This highlights the inherent challenges of simulating entangled quantum systems.

Code

All the data and the code for the data analysis can be found in this public Github repository: https://github.com/sdracia/QIC.git