Assignment 6

Quantum Information and Computing Course 2022/2023

Massimo Colombo

Department of Physics and Astronomy "Galileo Galilei"
DEGREE IN PHYSICS

December 14, 2022

Quantum Many-Body System: theory

A general many-body quantum system is composed of **N** subsystems of local dimension **D**.
The **total wavefunction** is obtained through a linear combination of the tensor products among all combinations of the N-subsystems base elements. A schematic representation is the following:

$$|\psi\rangle = \sum_{\vec{\alpha}} \psi_{\alpha_1\alpha_2\ldots\alpha_N} |\alpha_1\alpha_2\ldots\alpha_N\rangle$$

Where all α indices go from 1 to D.
We define a **separable state,** a state whose wavefunction can be written as a tensor product of certain subsystems wavefunctions:

$$|\psi\rangle_{sep} = \sum_{\alpha_1=1}^{d} \psi^1_{\alpha_1} |\alpha_1\rangle \otimes \sum_{\alpha_2=1}^{d} \psi^2_{\alpha_2} |\alpha_2\rangle \otimes \cdots \otimes \sum_{\alpha_N=1}^{d} \psi^N_{\alpha_N} |\alpha_N\rangle.$$

Where $|\alpha_i\rangle$ is the i-th element of the local base.
Two useful operators to study MB systems are the **density matrix operator** and the **reduced density matrix operator*:**
Given a state $|\psi\rangle$ in our system, the associated **density matrix operator** is defined in the following way:

$$\rho \equiv |\psi\rangle \langle\psi| \qquad \text{with the following properties:} \qquad \rho^2 = \rho, \quad Tr(\rho) = 1$$

The **reduced density matrix operator** is obtained "tracing out" the i-th subsystem from ρ. For instance, considering a 2-bodies system with $|\Psi\rangle \in H_A \otimes H_B$ the reduced density matrix where state B has been *traced out,* results:

$$\rho_A \overset{\text{def}}{=} \sum_{j}^{N_B} (I_A \otimes \langle j|_B) (|\Psi\rangle\langle\Psi|) (I_A \otimes |j\rangle_B) = \mathrm{Tr}_B\ \rho_T$$

Where N_B is the dimension of H_B, I_A is identity operator in H_A and $|j\rangle_B$ are the base elements of H_B.

*(Reduced) Density matrix is a positive semi-definite Hermitian operator.

State Initialization

Initialization of a pure or separable state:

```
subroutine init_mb_state(state, inner_dim, Nbodies, sep)
    state%sep = sep
    IF (sep) THEN
        state%dim = inner_dim * Nbodies
    ELSE
        state%dim = inner_dim ** Nbodies
    END IF
    ALLOCATE(state%comp(state%dim))
    CALL random_number(uu)
    iseed = (/FLOOR((4096)*uu(:)),1/) !Initializes random ISEED
    CALL zlarnv(2, iseed, state%dim, state%comp)
    state = .normalize.state
```

sep is a logical variable. It allows swapping between **separable** and **pure** state.

$D * N$

D^N

It generates random complex vector components

Given D and N, we consider the coefficients of the wavefunction of a pure state ordered in the "***tensorial way***", where the indices (α1, α2, α3, …, αN) go from 1 to D each.

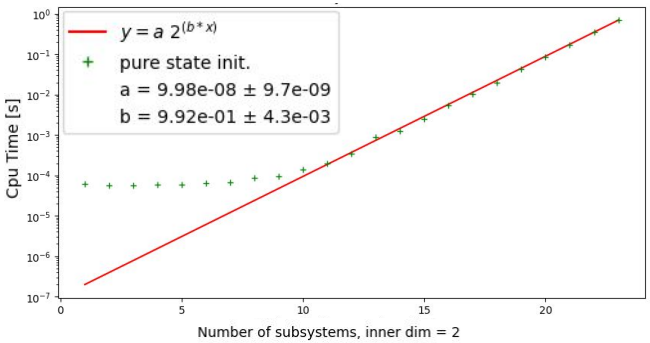
For instance, if N = 3 and D = 2, we get:

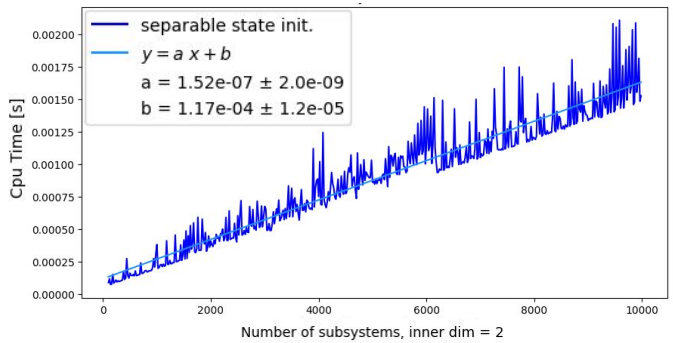
tensorial index: ((1, 1, 1), (2, 1, 1), (1, 2, 1), (2, 2, 1), (1, 1, 2), (2, 1, 2), (1, 2, 2), (2, 2, 2))

vector component: ((1), (2), (3), (4), (5), (6), (7), (8))

Thanks to the **Cpu time** routine, it was possible to measure the needed time to initialize both states for different inputs:

RESULTS:





As expected, we observe a exponential growth in time to initialize the pure state and a linear one for the separable state.

Density Matrix

To speed up normalization and collecting information about a state, a derived `type MB_wave` has been defined.
It contains "components" - `complex*16, allocatable, dimension(:) :: comp` ,
norm - `real*8 :: norm` , dimension - `integer*4 :: dim` and separability - `logical :: sep` .

To obtain the density matrix from a given pure state, we apply the "ket-bra" multiplication:
the tensor product between a vector and its associated dual form.

```
function get_density_matrix(wave) result(density_matrix)
    type(MB_wave), intent(in) :: wave
    ...
    allocate(density_matrix(wave%dim, wave%dim), dual(1, wave%dim), vec(wave%dim,1))
    dual(1,:) = conjg(wave%comp)
    vec(:,1) = wave%comp
    density_matrix = matmul(vec,dual)
```

Considering N = 2, D = 2 and sep = .false. , a random state is initialized and the associated density matrix is calculated:

RESULTS:

```
State components are:
    -0.10452022  -     0.52164418i
    -0.44252436  -     0.33304987i
     0.15807992  +     0.26428082i
    -0.19844092  -     0.52535744i
```

```
density matrix of the state results:
   0.28304  +   0.00000i     0.21999  +   0.19603i    -0.15438  -   0.05484i     0.29479  +   0.04861i
   0.21999  -   0.19603i     0.30675  +   0.00000i    -0.15797  +   0.06430i     0.26279  -   0.16639i
  -0.15438  +   0.05484i    -0.15797  -   0.06430i     0.09483  +   0.00000i    -0.17021  +   0.03060i
   0.29479  -   0.04861i     0.26279  +   0.16639i    -0.17021  -   0.03060i     0.31538  +   0.00000i
```

Let's check that the density matrix respects its theoretical properties:

```
Sq. dens. mat. of the state results:
   0.28304  +   0.00000i     0.21999  +   0.19603i    -0.15438  -   0.05484i     0.29479  +   0.04861i
   0.21999  -   0.19603i     0.30675  +   0.00000i    -0.15797  +   0.06430i     0.26279  -   0.16639i
  -0.15438  +   0.05484i    -0.15797  -   0.06430i     0.09483  +   0.00000i    -0.17021  +   0.03060i
   0.29479  -   0.04861i     0.26279  +   0.16639i    -0.17021  -   0.03060i     0.31538  +   0.00000i


Density matrix trace =
     1.00000000  +     0.00000000i
```

Reduced Density Matrix

The algorithm to calculate the reduced dens. mat., tracing out the i-th subsystem, is presented in the following:

D (dimension of each subsystem)

N (Number of subsystems)

I (Index of the subsystem against which we calculate the trace)

```
function get_red_density_matrix(density_matrix, inner_dim, Nbodies, system_index) result(red_density_matrix)
          ...
          ALLOCATE(red_density_matrix(inner_dim**(Nbodies-1), inner_dim**(Nbodies-1)))
          AA = inner_dim ** (Nbodies - system_index)
          BB = system_index - 1
          CC = inner_dim ** BB
          DD = inner_dim ** system_index

          DO j_v = 0, AA - 1
              DO j_d = 0, AA - 1
                  DO i_v = 1, CC
                      DO i_d = 1, CC
                          trace = 0
                          DO tt = 0, inner_dim -1
                          trace = trace + density_matrix(i_v + j_v * DD + tt * CC, i_d + j_d * DD + tt * CC)
                          END DO
                          red_density_matrix(i_v + j_v * CC,  i_d + j_d * CC) = trace
                  ...
```

*The 4-loops algorithm allows calculating the reduced density matrix for a generic system with D = inner_dim and N = Nbodies, with respect to the I-th (system_index) subsystem (tracing it out). To achieve this, the "**tensorial way**" of representation has been exploited.*

It cycles over the system's components with index lower than **I**.

It cycles over the components with dimension higher than **I**. (*DD) allows it.

Trace loop: (*CC) selects the right element of the **I**-th subsystem

i_v, **i_d**, **j_v** and **j_d** refer to the vectorial and dual (covariant) component of the matrix.

Considering previous state and density matrix, the reduced density matrices for both subsystems (N = 2) are calculated. Moreover, checks on the expected properties are made:

```
Reduced density matrix for state 1 results:
   0.58979  +   0.00000i      0.10840  -   0.22123i
   0.10840  +   0.22123i      0.41021  +   0.00000i


Squared Red. dens. mat for state 1 results:
   0.40854  +   0.00000i      0.10840  -   0.22123i
   0.10840  +   0.22123i      0.22897  +   0.00000i


Reduced density matrix for state 2 results:
   0.37787  +   0.00000i      0.04977  +   0.22663i
   0.04977  -   0.22663i      0.62213  +   0.00000i


Squared Red. dens. mat for state 2 results:
   0.19663  +   0.00000i      0.04977  +   0.22663i
   0.04977  -   0.22663i      0.44089  +   0.00000i
```

```
1st reduced mat trace =
      1.00000000  +     0.00000000i

Sq. 1nd red mat trace =
      0.63751225  +     0.00000000i

2nd reduced mat trace =
      1.00000000  +     0.00000000i

Sq. 2nd red mat trace =
      0.63751225  +     0.00000000i
```

Ok!

Tr < 1, so Ok!

Ok!

Tr < 1, so Ok!

We expect the trace of each square reduced density matrix to be less than one.

Von Neumann Entropy

To investigate how two states are entangled, Von Neumann's entropy S has been exploited. The expectation values are S = 0 for a separable state and S = log (2) for a maximally entangled state (in the case of 2 Qubits).

$$S = -\mathrm{Tr}(\rho_A \log \rho_A) = -\sum_i p_i \log(p_i),$$

Where ρ_A is the Reduced density matrix with respect to the system A, and "p_i" are the eigenvalues of ρ_A.

```
function get_Neumann_entropy(density_matrix) result(entropy)
    dim = size(density_matrix, 1)
    eigenvalues = diagonalize_herm_mat(density_matrix)
    entropy = 0
    DO ii = 1, dim
        IF (eigenval(ii) .GT. 0) THEN
        entropy = entropy - real(eigenval(ii) * log(eigenval(ii)))
        END IF
    END DO
```

For the 2-Qubits system (with D = 2 and N = 2), the **maximally entangled state** (1, 0, 0, 1)/√2
and the **separable state** (1, i, 0, 0)/√2
have been considered to check the correctness of the code.

RESULTS:

Maximally entangled state's entropy:

```
Entropy of 1st state =
.6931471825

Entropy of 2nd state =
.6931471825
```

As expected.

Separable state's Neumann entropy:

```
Entropy of 1st state =
.0000000000

Entropy of 2nd state =
.0000000000
```

The program also returns other interesting results (**DM**, **RDM**, …) for these 2 specific states. Just compile the code, execute it (./Ex1.out -d3) and enter when requested: D = 2, N = 2 and "no" to the separability request (2 Qubits system). Then follow the instructions!