Information Theory and Computation
Exercise 8

Vincenzo Maria Schimmenti - 1204565

December 10, 2019

Theory

In this exercise we are gonna study some structures and methods in order to handle numerically pure N body wavefunctions. We assume that all single particles wavefunctions ψ_i live inside a d dimensional Hilbert space $\mathbb{C}^d$. Hence the Hilbert space of any N body wavefunction Ψ is $\mathbb{C}^d \otimes \cdots \otimes \mathbb{C}^d \cong \mathbb{C}^{d^N}$; a generic state of this kind can be written using single particle basis $\{|\alpha_i\rangle, \alpha_i = 0, \ldots, d-1\}$:

$$\Psi = \sum_{\alpha_1,\alpha_2,\ldots,\alpha_N} C_{\alpha_1,\alpha_2,\ldots,\alpha_N} |\alpha_1\rangle \otimes |\alpha_2\rangle \otimes \cdots \otimes |\alpha_N\rangle \tag{1}$$

Having chosen the basis, any wavefunction needs d^N complex number to be fully specified (up to a phase). However sometimes we deal with special kinds of states, called *separable*, which require only dN complex coefficents and are represented as following:

$$\Psi = \bigotimes_{i=1}^{N} \sum_{\alpha_i} C_{i,\alpha_i} |\alpha_i\rangle \tag{2}$$

This kind of states are extremely important since are the basic ingredient for doing mean field theories.
We are also gonna study density matrices of pure states, $\rho = |\Psi\rangle \langle\Psi|$, and ways to perform operations on them, such as traces.

Code Development

To encode the N body pure wave function we chose to introduce a type, *pstate*:

```
type pstate
        ! Hilbert space dimension
        integer*4 :: dim
        ! Number of particles
        integer*4 :: np
        ! Separable state
        logical :: sep
        ! Wave function coefficents (either dim*np or dim**np)
        complex*16, dimension(:), allocatable :: psi
end type
```

The easiest N body states to handle, as we said, is a pure separable state which needs just dN coefficents, can be either represented using a dN $\mathbb{C}$-vector or a $d \times N$ matrix; we chose to follow the first way with the natural convention that every d coefficents in the vector we find a one particle state. If one wants to get the coefficent of the basis element $|\alpha_1 \alpha_2 \dots \alpha_N\rangle$ (where we omitted the tensor product) we can use the following code:

```
coeff = 1.0
do ii=0,N-1
        coeff = coeff*state%psi(ii*d+sIndex(ii+1)+1)
end do
```

In the code *sIndex* is an integer vector containing the values of $\alpha_i, i = 1 \dots N$. From this implementation we understand that the method to get a N body coefficient has a complexity $O(N)$.
The matter is more complex for indexing generic pure states; indeed we need to construct a map from all possible $\alpha_1, \dots, \alpha_N$ and an integer number that goes from 1 to d^N which spans all the basis elements. The way to do this is to use the following map:

$$\text{state index} = f(\alpha_1, \dots, \alpha_N) = \alpha_1 + d\alpha_2 + \dots + \alpha_N d^{N-1} = \sum_{i=1}^{N} d^{i-1}\alpha_i \quad (3)$$

So if we start from a set of coefficents $\alpha_1, \dots, \alpha_N$ the method to retrieve the index has complexity $O(N)$:

```
accum=1
do ii=1, N
        jj = jj + accum*sIndex(ii)
        accum = accum*d
end do
coeff = state%psi(jj+1)
```

If one wants to find all the indices of the states with a definite value α_s^* of the s-th particle one has to loop through two numbers, $r = 0 \dots d^{s-1}$ and $l = 0, \dots, d^{N-s}$ and build the index as $(l \times d + \alpha_s^*)d^{s-1} + r + 1$: this result is pretty useful when one wants to trace out a subsystem from a density matrix.
Now, given a pure state, if one wants to build a density matrix out of it, we use the following function which exploits the *matmul* function of Fortran:

```
! Performs the outer product between the vector 'x' and 'y'.
! The 'y' is conjugated.
function vector_oproduct(x,y)result(rho)
        complex*16, dimension(:), intent(in) :: x,y
        complex*16, dimension(:,:), allocatable :: rho
        integer*4 :: nn, mm
        nn = size(x)
        mm = size(y)
        rho = matmul(reshape(x, (/nn,1/)), reshape(conjg(y), (/1,mm/)))
end function
```

Assuming the density matrix we are dealing with is build from a pure state described above, for tracing out the subsystem *sSystem*:

```
sz = dd**nn
szp = dd**(nn-1)
rightMax = dd**(sSystem-1)
leftMax = dd**(nn-sSystem)
allocate(rhop(szp, szp))
factor = dd**(sSystem-1)
do rri=0, rightMax-1
        do lli=0, leftMax-1
                do rrj=0, rightMax-1
                        do llj =0, leftMax-1
                                ! index for the traced matrix
                                iip = lli*factor+rri+1
                                jjp = llj*factor+rrj+1
                                temp = complex(0.0,0.0)
                                do val=0,dd-1
                                        ! indices for the original matrix
                                        ii =(lli*dd+val)*factor+rri+1
                                        jj =(llj*dd+val)*factor+rrj+1
                                        temp = temp+rho(ii,jj)
                                end do
                                rhop(iip,jjp)=temp
                        end do
                end do
        end do
end do
```

Here *dd* is the Hilbert space dimension, *nn* the number of particles, *rho* the density matrix and *rhop* the resulting density matrix.

Results

We tried to generate different states for various number of particles (and a 2 dimensional Hilbert space). The maximum number of particles we could acheive is $N = 12$, since for bigger N's Fortran could not allocate memory.
Using $N = 2$, we tested our program both on a density matrix generated from two separable states and one explicitly given:

```
$ ./a.exe states.txt 0
Reading two separable states...
State for system A (  0.70710678118000003     ,  0.0000000000000000     ) (  0.70710678118000003     ,  0.0000000000000000     )
State for system B (   1.0000000000000000     ,  0.0000000000000000     ) (   0.0000000000000000     ,  0.0000000000000000     )
 ----------------------------
Real part of the density matrix A+B
 0.5000| 0.5000| 0.0000| 0.0000|
 0.5000| 0.5000| 0.0000| 0.0000|
 0.0000| 0.0000| 0.0000| 0.0000|
 0.0000| 0.0000| 0.0000| 0.0000|
Imaginary part of the density matrix A+B
 0.0000| 0.0000| 0.0000| 0.0000|
 0.0000| 0.0000| 0.0000| 0.0000|
 0.0000| 0.0000| 0.0000| 0.0000|
 0.0000| 0.0000| 0.0000| 0.0000|
----------------------------
Real part of the reduced density matrix A
 0.5000| 0.5000|
 0.5000| 0.5000|
Imaginary part of the reduced density matrix A
 0.0000| 0.0000|
 0.0000| 0.0000|
----------------------------
Real part of the reduced density matrix B
 1.0000| 0.0000|
 0.0000| 0.0000|
Imaginary part of the reduced density matrix B
 0.0000| 0.0000|
 0.0000| 0.0000|
----------------------------
```

```
$ ./a.exe matrix.txt 1
 Reading qubits density matrix for two particles...
 ------------------------------
 Real part of the density matrix A+B
| 0.5000| 0.0000| 0.0000|-0.5000|
| 0.0000| 0.0000| 0.0000| 0.0000|
| 0.0000| 0.0000| 0.0000| 0.0000|
|-0.5000| 0.0000| 0.0000| 0.5000|
 Imaginary part of the density matrix A+B
| 0.0000| 0.0000| 0.0000| 0.0000|
| 0.0000| 0.0000| 0.0000| 0.0000|
| 0.0000| 0.0000| 0.0000| 0.0000|
| 0.0000| 0.0000| 0.0000| 0.0000|
 ------------------------------
 Real part of the reduced density matrix A
| 0.5000| 0.0000|
| 0.0000| 0.5000|
 Imaginary part of the reduced density matrix A
| 0.0000| 0.0000|
| 0.0000| 0.0000|
 ------------------------------
 Real part of the reduced density matrix B
| 0.5000| 0.0000|
| 0.0000| 0.5000|
 Imaginary part of the reduced density matrix B
| 0.0000| 0.0000|
| 0.0000| 0.0000|
 ------------------------------
```

The resulting density matrix are computed using the aforementioned method. For a consistency check, the first state is build from the separable state

$$\left(\frac{1}{\sqrt{2}}\left|0\right\rangle_A + \frac{1}{\sqrt{2}}\left|1\right\rangle_A\right) \otimes \left|0\right\rangle_B \tag{4}$$

which gives:

$$\rho = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{5}$$

The second is generated from:

$$\frac{1}{\sqrt{2}} |00\rangle - \frac{1}{\sqrt{2}} |11\rangle \tag{6}$$

i.e.:

$$\rho = \begin{pmatrix} \frac{1}{2} & 0 & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{2} & 0 & 0 & \frac{1}{2} \end{pmatrix} \tag{7}$$