Density Matrices

Umberto Maria Tomasini
Quantum Information Course

04/11/2018

Abstract

In this report we show how store the information about the wave function of a N-body non interacting, separable and pure state. The same is shown for a generic N-body pure state. The next step is to write a density matrix, given a pure state. Then we show a subroutine helpful to compute the reduce density matrix of one sub-system. That subroutine traces off a single sub-system from an initial density matrix, hence iterating the application $N - 1$ times the reduced density matrix can be computed. Everything is tested on a system composed of 2 qubits.

Theory

Wave function of a general pure state

Considering a system composed of N sub-systems, with each sub-system with a wave function $\psi_i \in \mathcal{H}^D$, a general N-body pure state wave function $|\Psi\rangle \in \mathcal{H}^{D^N}$ is the following:

$$|\Psi\rangle = \sum_{\alpha_1,...,\alpha_N} C_{\alpha_1,...,\alpha_N} |\psi_{\alpha_1}\rangle \otimes ... \otimes |\psi_{\alpha_N}\rangle \tag{1}$$

$$\alpha_j \in \{1, ..., D\}$$

We can rewrite $|\Psi\rangle$ as following:

$$|\Psi\rangle = \sum_{i=1}^{D^N} d_i |i\rangle \tag{2}$$

$|i\rangle$ is a wavefunction which belongs to a basis in $\mathcal{H}^{D^N}$, and it is the tensor product of N different ψ_{α_i}. Without losing generality, we can impose that d_1 is real (the global phase is meaningless) and the normalization is one. The link between the two different formulations of $|\Psi\rangle$ s given by the following link between N indexes of N different ψ_{α_i} and 1 index of one $|i\rangle$ (i is written in a $D-$base, considering that the α and i start at one):

$$i = (\alpha_1 - 1) \cdot D^0 + (\alpha_2 - 1) \cdot D^1 + ... + (\alpha_N - 1) \cdot D^{N-1} + 1 \tag{3}$$

Wave function of a separable pure state

A separable pure state is such that the general definition (1) can be written as follows:

$$|\Psi\rangle = \sum_{\alpha_1,...,\alpha_N} C_{\alpha_1,...,\alpha_N} |\psi_{\alpha_1}\rangle \otimes ... \otimes |\psi_{\alpha_N}\rangle = \sum_{\alpha_1,...,\alpha_N} C_{\alpha_1} C_{\alpha_2} ... C_{\alpha_N} |\psi_{\alpha_1}\rangle \otimes ... \otimes |\psi_{\alpha_N}\rangle$$

$$|\Psi\rangle = \sum_{\alpha_1} C_{\alpha_1} |\psi_{\alpha_1}\rangle \otimes ... \otimes \sum_{\alpha_N} C_{\alpha_N} |\psi_{\alpha_N}\rangle \tag{4}$$

$$\alpha_i \in \{1, ..., D\}$$

Hence we have the tensor product of N wave functions which belong to N different Hilbert spaces. So we can choose to set real every first coefficient of each one of those wave functions. Furthermore, we can choose to normalize each one od those N wave functions to one. Instead of having D^N coefficents, we have only $N \cdot D$ coefficients.

Density matrix of a pure state

Given a generic pure state $|\Psi\rangle = \sum_{i=1}^{D^N} d_i |i\rangle \in \mathcal{H}^{D^N}$, the relative density matrix is defined as follows:

$$\rho = |\Psi\rangle\langle\Psi| = \sum_{i,j=1}^{D^N} d_j^* d_i |i\rangle\langle j| \tag{5}$$

Reduced Density matrix

Given a generic density matrix ρ in $\mathcal{H}^{D^N}$, the reduced density matrix relative to the $k-$ subsystem is:

$$\rho_k = Tr_1 ... Tr_{k-1} Tr_{k+1} ... Tr_N \rho \tag{6}$$

$$Tr_j \rho = \sum_{j=1}^{D} \langle j|\rho|j\rangle$$

Code Development

The following types turn to be useful.

Listing 1: Fortran code: Derived types: one for the states and one for the matrices.

```
        type state
!defining the type:
!separable pure state for nn subsystems
!which wavefunction belongs to H^D, i.e.
!i.e. D dimensional Hilbert space

integer :: nsub
!number of subsystems

integer :: ndim
!dimension Hilbert space single psi

!the coefficients of the total wave function
double complex, dimension (:), allocatable :: coeff

logical :: sep
!if .true. it is separable, if .false., no

logical :: pur
!if .true. it is pure, if .false., no

integer :: len_state
!length of the state

end type state


type dcm !defining the type: doublecomplex matrix
integer :: nr !number of rows
integer :: nc !number of columns
!the actual matrix
double complex, dimension (: , :), allocatable :: elem
```

```
!eigenvalues
double precision, dimension (:), allocatable :: eval
!trace
double complex :: m_trace
!determinant
double complex :: m_det
end type dcm
```

Wave function of a general pure state/Wave function of a separable pure state

For a general pure state a N^D vector is initialized with random numbers between 0 and 1 (both real and imaginary part). The first entry is real and the norm is set to one.
For a separable pure state a $N \cdot D$ vector is initialized with random numbers between 0 and 1 (both real and imaginary part). The first D entries are dedicated to the first body, the second to the second and so on. Each entry of the type $1+(jj-1)D$, with $jj \in \{1, ..., N\}$, is real and the norm is set to one. Every group of D coefficients related to one body is normalized to one.

Listing 2: Fortran code: Initialization of a type(state)

```
subroutine init_state (psi, nn, dd, debug, sepp, purr)
!this subroutine initializes a type state
!with random numbers between 0 and 1
!(both real and imaginary part)
!If sepp.eqv .true. then the state is separable
!hence a nn*dd vector
!the first coefficient of each wavefunction
!of one subsystem is set to real
!each single subsystem wavefunction
!is normalized to one
!If sepp.eqv .false. then the state is not separable
!hence a dd**nn vector
!the first coefficient of the wavefunction
!of one subsystem is set to real
!the wavefunction is normalized to one

integer :: ii, jj, kk
integer :: nn, dd
type(state) :: psi
integer :: my_stat
character (256) :: my_msg
logical :: debug, sepp, purr
double precision :: xx, yy
double precision :: summ, tempp
character(:), allocatable :: namefile
```

```
psi%nsub= nn
psi%ndim= dd

psi%sep=sepp


!SEPARABLE
if(psi%sep .eqv. .true.) then

psi%len_state=nn*dd

!WARNING IF NOT POSITIVE
if(debug.eqv..TRUE.) then
if(psi%nsub <= 0) then
print*, "nn␣<=␣0"
print*, "The␣actual␣value␣is", nn
end if

if(psi%ndim <= 0) then
print*, "dd␣<=␣0"
print*, "The␣actual␣value␣is", dd
end if
end if


!ALLOCATION VECTOR
allocate (psi%coeff(nn*dd),
$                        stat=my_stat, errmsg=my_msg)
!error handling allocation
if(my_stat /= 0) then
print*, 'Failed␣to␣allocate␣psi%coeff␣with␣stat␣=␣'
$                , my_stat, '␣and␣msg␣=␣'//trim(my_msg)
end if

!INITIALIZATION COEFFICIENTS
!subsystem by subsystem
do ii= 1, psi%nsub

do jj= 1, psi%ndim
call random_number(xx)

if (jj==1) then
psi%coeff(jj+(ii-1)*psi%ndim)=cmplx(xx,0)
!setting the global phase
```

```
!setting the first coeff real
else
call random_number(yy)
psi%coeff(jj+(ii-1)*psi%ndim)=cmplx(xx,yy)
end if
end do

summ=0.0
!norm computation for each subsystem
do kk= 1, psi%ndim

tempp= abs(psi%coeff(kk+(ii-1)*psi%ndim))**2
summ=summ + tempp
end do
summ=sqrt(summ)

!normalizing the coefficients
!dedicated to each subsystem
!in such a way that each subsystem is normalized
!to 1/sqrt(number of subsystems)
!Hence the total wave function is normalized to 1
do kk= 1, psi%ndim

psi%coeff(kk+(ii-1)*psi%ndim)=
$                        psi%coeff(kk+(ii-1)*psi%ndim)/(summ)
end do

if(debug.eqv..TRUE.) then

!check on norm
print*, "␣"
print*, "NORM␣of␣subsystem␣number:␣", ii
summ=0.0
!norm computation
do kk= 1, psi%ndim*psi%nsub
tempp= abs(psi%coeff(kk+(ii-1)*psi%ndim))**2
summ=summ + tempp
end do
summ=sqrt(summ)
print*, summ
print*, "␣"

end if
end do
```

```
if(debug.eqv..TRUE.) then!check
namefile="sep_purestate.txt"
call print_vector(psi%coeff, nn*dd, namefile, 80)
print*, "all␣subsystems"
do kk= 1, psi%ndim*psi%nsub
print*, psi%coeff(kk)
end do
print*, "␣"

end if

!------------
else !GENERAL
!------------

psi%len_state=dd**nn

!ALLOCATION VECTOR
allocate (psi%coeff(dd**nn),
$                        stat=my_stat, errmsg=my_msg)
!error handling allocation
if(my_stat /= 0) then
print*, 'Failed␣to␣allocate␣psi%coeff␣with␣stat␣=␣'
$                , my_stat, '␣and␣msg␣=␣'//trim(my_msg)
end if

!INITIALIZATION COEFFICIENTS
do ii= 1, psi%ndim**psi%nsub

call random_number(xx)

if (ii==1) then
psi%coeff(ii)=cmplx(xx,0)
!setting the global phase
!setting the first coeff real
else
call random_number(yy)
psi%coeff(ii)=cmplx(xx,yy)
end if
end do

summ=0.0
!norm computation
do kk= 1, psi%ndim**psi%nsub
tempp= abs(psi%coeff(kk))**2
```

```
summ=summ + tempp
end do
summ=sqrt(summ)

!normalizing the coefficients
do kk= 1, psi%ndim**psi%nsub
psi%coeff(kk)= psi%coeff(kk)/summ
end do

if(debug.eqv..TRUE.) then!check
namefile="gen_purestate.txt"
call print_vector(psi%coeff, dd**nn, namefile, 30)

!check on norm
print*, "GENERAL PURE STATE"
print*, "NORM of general pure state"
summ=0.0
!norm computation
do kk= 1, psi%ndim**psi%nsub
tempp= abs(psi%coeff(kk))**2
summ=summ + tempp
end do
summ=sqrt(summ)
print*, summ
print*, " "

do kk= 1, psi%ndim*psi%nsub
print*, psi%coeff(kk)
end do


end if

end if

end subroutine init_state
```

Density matrix of a pure state

The subroutine $denmat\_pure$ computes the density matrix of a general pure state of N subsystems, each one belonging to a D-dimensional Hilbert space. Hence the density matrix is $D^N \times D^N$.
If the the pure state is a separable one (which information is stored in a $N \cdot D$ vector), the subroutine $fromsep\_togen$ takes a separable state and returns the D^N vector. Given a $i-$ index of the D^N vector, the N coefficients $\alpha_1, ..., \alpha_N$ of the formula (3) are computed. Then,

the $i-th$ entry of the D^N vector is computed as the product of N coefficients stored in the $N \cdot D$ vector. These last ones are the entries (α_1), $(\alpha_2 + 1 \cdot D)$, $(\alpha_3 + 2 \cdot D)$,..., $(\alpha_N + (N-1) \cdot D)$ of the $N \cdot D$ vector.

Listing 3: Fortran code: Computation of the density matrix of a pure state, in the subroutine denmat_pure. In case of the separable pure state, the passage from the $N \cdot D$ vector to the D^N vector is performed in the second subroutine fromsep_togen

```
subroutine denmat_pure(psi, denmat, debug)
!this subroutine computes the
!density matrix of a general pure state
!of N subystems, each one belonging
!to a D-dimensional Hilbert space
!hence the density matrix is D**N x D**N
type(state) :: psi
type(state) :: psi_large
type(dcm) :: denmat
integer :: ii,jj, ll, tt , ss
character(:), allocatable :: namefile
logical :: debug, sepp, purr
integer:: nr_denmat, nc_denmat
integer :: accum

nr_denmat= psi%len_state
nc_denmat= psi%len_state

call init_dcm_mat (denmat, nr_denmat, nc_denmat,debug)

!-------------------------
if(psi%sep .eqv. .true.) then
!if the psi_is separable, written like a nn*dd vector
!this subroutine gives the relative dd**nn vector
!in the multi-state basis
call fromsep_togen(psi,psi_large)
!-------------------------

!COMPUTATION DENSITY MATRIX SEPARABLE STATE
do ii= 1, psi_large%len_state

do jj= 1, psi_large%len_state

denmat%elem(ii,jj)=
$                 conjg(psi_large%coeff(ii))*psi_large%coeff(
   jj)
```

```
end do
end do

if(debug.eqv..TRUE.) then!check

print*, "␣"
print*, "DENSITY␣MATRIX"

!computation of TRACE
!if the psi is a pure state
!then it should be one
print*, "␣"

do ii= 1, psi_large%len_state
denmat%m_trace=
$                        denmat%m_trace +denmat%elem(ii,ii)
end do

print*, "TRACE:␣", denmat%m_trace
print*, "␣"

!check of hermitianity
accum=0
do tt=1,psi%len_state
do ss=1,psi%len_state

if(abs(conjg(denmat%elem(tt,ss))-denmat%elem(ss,tt))
$                        /abs(denmat%elem(ss,tt))> 10E-10)
   then
accum=accum+1
end if

end do
end do

if(accum>0) then
print*, "Hermitianity␣violated␣in"
$                        , accum, "different␣entries"
else
print*, "Hermitianity␣checked"
end if
namefile="sep_denmat.txt"
call print_matrices(denmat, namefile, 60)

end if
```

```
!-------------------------
else
!-------------------------


!COMPUTATION DENSITY MATRIX GENERAL STATE
do ii= 1, psi%len_state

do jj= 1, psi%len_state

denmat%elem(ii,jj)=
$                 conjg(psi%coeff(ii))*psi%coeff(jj)

end do
end do

if(debug.eqv..TRUE.) then!check

print*, "␣"
print*, "DENSITY␣MATRIX"
!computation of TRACE
!if the psi is a pure state
!then it should be one
print*, "␣"

do ii= 1, psi%len_state
denmat%m_trace=
$                       denmat%m_trace +denmat%elem(ii,ii)
end do

print*, "TRACE:␣", denmat%m_trace
print*, "␣"
!check of hermitianity
accum=0
do tt=1,psi%len_state
do ss=1,psi%len_state

if(abs(conjg(denmat%elem(tt,ss))-denmat%elem(ss,tt))
$                       /abs(denmat%elem(ss,tt))> 10E-8) then
accum=accum+1
print*, accum
end if

end do
```

```
end do

if(accum>0) then
print*, "Hermitianity␣violated␣in"
$                        , accum, "different␣entries"
else
print*, "Hermitianity␣checked"
end if

namefile="gen_denmat.txt"
call print_matrices(denmat, namefile, 50)

end if

end if

end subroutine denmat_pure

subroutine fromsep_togen(psi,psi_large)
!this subroutine takes a separable state
!which information is stored in nn*dd vector
!and gives the dd**nn vector in the multi-state basis

type(state) :: psi
type(state) :: psi_large
logical sepp, purr, debug
integer :: ii, jj, ll
integer :: cc, kk
integer, dimension(:), allocatable :: vec_alpha
double precision :: summ, tempp
double complex :: temp

sepp=.false.
purr=.true.

debug=.false.

call init_state(psi_large, psi%nsub, psi%ndim, debug, sepp,
   purr)

allocate(vec_alpha(psi%nsub))



!this cycle allows to find the coefficients
```

```
!of the multistate basis as a function of the
!coefficients of the one-state basis
do ii= 1, psi_large%len_state

kk=ii-1
!this cycle finds which indexes (alpha) of the vector nn*dd
!are associate to an index of the vector dd**nn
!they are stored in a vector vec_alpha
do jj= 1, psi%nsub

cc=psi%nsub-jj

vec_alpha(cc+1)=int((kk-mod(kk,psi%ndim**cc))
$                                        /psi%ndim**cc +0.1)
   +1

kk=kk-(vec_alpha(cc+1)-1)*(psi%ndim**cc)


end do

temp=(1.,0.)

do jj= 1, psi%nsub

temp= temp*
$                                psi%coeff(vec_alpha(jj)+(jj
   -1)*psi%ndim)

end do
psi_large%coeff(ii)=temp

end do

if(debug.eqv..TRUE.) then!check

print*, "Enlarged psi"
do ii= 1, psi_large%len_state
print*, psi_large%coeff(ii)
end do

!norm computation
summ=0.
do kk= 1, psi_large%len_state
tempp= abs(psi_large%coeff(kk))**2
```

```
summ=summ + tempp
end do
print*, "NORM enlarged psi"
summ=sqrt(summ)
print*, summ
print*, " "



end if

end subroutine fromsep_togen
```

Reduced Density matrix

The subroutine red_denmat is helpful in order to compute the reduced density matrix of a general density matrix ρ in a D^N-dimensional Hilbert space. Given a $k-th$ sub-system, the trace is performed on ρ on that sub-system via the following subroutine. Hence the reduced density matrix given as output is in a D^{N-1}-dimensional Hilbert space. The link between the reduced matrix and the initial one is the following.
The initial matrix is:

$$\rho = \sum_{i,j=1}^{D^N} \rho_{ij} |i\rangle\langle j| \tag{7}$$

$$\rho = \sum_{\alpha_1,...,\alpha_N} \sum_{\alpha_1^{'},...,\alpha_N^{'}} C_{\alpha_1,...,\alpha_N} C^*_{\alpha_1^{'},...,\alpha_N^{'}} |\psi_{\alpha_1}\rangle \otimes ... \otimes |\psi_{\alpha_N}\rangle\langle\psi_{\alpha_1^{'}}| \otimes ... \otimes \langle\psi_{\alpha_N^{'}}|$$

$$\Rightarrow \rho = \sum_{\alpha_1,...,\alpha_N} \sum_{\alpha_1^{'},...,\alpha_N^{'}} C^{\alpha_1^{'},...,\alpha_N^{'}}_{\alpha_1,...,\alpha_N} |\psi_{\alpha_1}\rangle \otimes ... \otimes |\psi_{\alpha_N}\rangle\langle\psi_{\alpha_1^{'}}| \otimes ... \otimes \langle\psi_{\alpha_N^{'}}|$$

The reduced one $\rho^{(k)}$, obtained tracing off the $k-th$ subsystem, is:

$$\rho^{(k)} = \sum_{\substack{\alpha_1,..,\alpha_{k-1},\alpha_{k+1},..,\alpha_N \\ \alpha_1^{'},..,\alpha_{k-1}^{'},\alpha_{k+1}^{'},..,\alpha_N^{'}}} \left(\sum_{\alpha_k} C^{\alpha_1^{'},..,\alpha_k,..,\alpha_N^{'}}_{\alpha_1,..,\alpha_k,..,\alpha_N} \right) |\psi_{\alpha_1...\alpha_{k-1},\alpha_{k+1},...\alpha_N}\rangle\langle\psi_{\alpha_1^{'}...\alpha_{k-1}^{'},\alpha_{k+1}^{'},...\alpha_N^{'}}|$$

Where

$$|\psi_{\alpha_1...\alpha_{k-1},\alpha_{k+1},...\alpha_N}\rangle = |\psi_{\alpha_1}\rangle...|\psi_{\alpha_{k-1}}\rangle|\psi_{\alpha_{k+1}}\rangle...|\psi_{\alpha_N}\rangle$$
$$\langle\psi_{\alpha_1^{'}...\alpha_{k-1}^{'},\alpha_{k+1}^{'},...\alpha_N^{'}}| = \langle\psi_{\alpha_1^{'}}|....\langle\psi_{\alpha_{k-1}^{'}}|\langle\psi_{\alpha_{k+1}^{'}}|...\langle\psi_{\alpha_N^{'}}|$$

We have obtained the following relation between coefficients of the imput and output density matrix:

$$\rho^{(k)}_{\substack{\alpha_1,..,\alpha_{k-1},\alpha_{k+1},..,\alpha_N \\ \alpha_1^{'},..,\alpha_{k-1}^{'},\alpha_{k+1}^{'},..,\alpha_N^{'}}} = \sum_{\alpha_k} C^{\alpha_1^{'},..,\alpha_k,..,\alpha_N^{'}}_{\alpha_1,..,\alpha_k,..,\alpha_N} \tag{8}$$

Which can be rewritten as:

$$\rho_{rs}^{(k)} = \sum_{\alpha_k=1}^{D} \rho_{r(\alpha_k),s(\alpha_k)} \tag{9}$$

$$r(\alpha_k) = (r-1)\%D^{k-1} + (\alpha_k - 1)D^{k-1} + (r - 1 - (r-1)\%D^{k-1})D + 1$$
$$s(\alpha_k) = (s-1)\%D^{k-1} + (\alpha_k - 1)D^{k-1} + (s - 1 - (s-1)\%D^{k-1})D + 1$$

In this way it is possible to compute the reduced matrix coefficients from the initial density matrix coefficients.
Applying this subroutine $N-1$ times on the initial ρ is possible to compute the density matrix dedicated to only one sub-system.

Listing 4: Fortran code: Computation of the reduced density matrix obtained tracing off the k-th sub-system.

```
      subroutine red_denmat(denmat_tot,
      $                        denmat_red, dd, nn, kk, debug
        )
      !this subroutine computes the reduced
      !density matrix of a general density matrix
      !in a dd**nn-dimensional Hilbert space
      !The trace is performed on the kk-th subsystem
      !Hence the reduced density matrix
      !is in a dd**(nn-1)-dimensional Hilbert space

      type(dcm) :: denmat_tot
      type(dcm) :: denmat_red
      integer :: ii, jj, kk
      integer :: aa ,bb
      character(:), allocatable :: namefile
      integer :: nr_red, nc_red
      logical :: debug
      integer :: nn, dd
      integer :: ind_c, ind_r, ind_kk
      integer :: tt, ss, accum

      nr_red= dd**(nn-1)
      nc_red= dd**(nn-1)


      call init_dcm_mat (denmat_red, nr_red, nc_red,debug)


      !COMPUTATION REDUCED DENSITY MATRIX
      do aa= 1, nr_red
```

```
do bb= 1, nc_red

denmat_red%elem(aa,bb)=(0.0, 0.0)
do ind_kk= 1, dd

ind_r = mod(aa-1,(dd**(kk-1))) + (ind_kk-1)*(dd**(kk
   -1))
$             +(aa-1-mod(aa-1,(dd**(kk-1))))*(dd)+1



ind_c = mod(bb-1,(dd**(kk-1))) + (ind_kk-1)*(dd**(kk
   -1))
$             +(bb-1-mod(bb-1,(dd**(kk-1))))*(dd)+1
denmat_red%elem(aa,bb)= denmat_red%elem(aa,bb)+
$                       denmat_tot%elem(ind_r,ind_c)


end do

end do

end do

if(debug.eqv..TRUE.) then!check

print*, "␣"
print*, "DENSITY␣MATRIX"
!computation of TRACE
!if the psi is a pure state
!then it should be one
print*, "␣"

do ii= 1, dd**nn
denmat_red%m_trace=
$                       denmat_red%m_trace +
   denmat_red%elem(ii,ii)
end do

print*, "TRACE:␣", denmat_red%m_trace
print*, "␣"
!check of hermitianity
accum=0
do tt=1,dd**(nn-1)
do ss=1,dd**(nn-1)
```

```
        if(abs(conjg(denmat_red%elem(tt,ss))-denmat_red%elem(
           ss,tt))
        $                          /abs(denmat_red%elem(ss,tt))>
            10E-8) then
        accum=accum+1
        print*, accum
        end if

        end do
        end do

        if(accum>0) then
        print*, "Hermitianity violated in"
        $                         , accum, "different entries"
        else
        print*, "Hermitianity checked"
        end if

        end if

        if(debug.eqv..TRUE.) then!check
        namefile="denmat_red.txt"
        call print_matrices(denmat_red, namefile, 90)
        end if

        end subroutine red_denmat
```

Results, on 2 qubits

The code is tested on a N=2, D=2 system.

General pure state

The generated 2^2 random vector storing the information of the general pure state is the following:
(0.29564688802054850,0.0000000000000000)
(0.52088000320975480,0.58088547769063004)
(0.41083932083430852,6.29247396325845199E-002)
(0.33612657971304960,0.13460852007611654)

The first entry is real and the norm is one.

The **density matrix** computed from that general pure state is the following. It is noticeable that the trace is one, as it should be. A check confirmed that it is hermitian.

NUMBER OF ROWS: 4
NUMBER OF COLUMNS: 4
ELEMENTS:
FIRST ROW
(8.74070823962347504E-002,0.0000000000000000)
(0.1539965519810973 2,0.17173698377556451)
(0.12146336668113901,1.86035034518768870E-002)
(9.93747772731539414E-002,3.97965900615553786E-002)
SECOND ROW
(0.1539965519810973 2,-0.17173698377556451)
(0.60874391593566557,0.0000000000000000)
(0.25055005419490051,-0.20587435655513769)
(0.25327374840546091,-0.12513616245182452)
THIRD ROW
(0.12146336668113901,-1.86035034518768870E-002)
(0.25055005419490051,0.20587435655513769)
(0.17274847040142444,0.0000000000000000)
(0.14656422180178552,3.41517954545482866E-002)
FOURTH ROW
(9.93747772731539414E-002,-3.97965900615553786E-002) (0.25327374840546091,0.12513616245182452
(0.14656422180178552,-3.41517954545482866E-002) (0.13110053126667534,0.0000000000000000)
TRACE: (1.0000000000000000,0.0000000000000000)

The **reduced density matrix** computed from the initial density matrix, tracing off the second sub-system, is the following. It is hermitian and with trace one, as it should be.

NUMBER OF ROWS: 2
NUMBER OF COLUMNS: 2
ELEMENTS:
FIRST ROW:
(0.26015555279765917,0.0000000000000000)
(0.30056077378288282,0.20588877923011278)
SECOND ROW:
(0.14495682555673411,0.11184405979423383)
(0.25208417656547361,0.0000000000000000)
TRACE: (1.0000000000000000,0.0000000000000000)

Separable pure state

The generated $2 \cdot 2$ random vector storing the information of the separable pure state is the following:
(0.23895527483715603,0.0000000000000000)
(0.95207849628463093,0.19091074757565418)
(0.82491507559912247,0.0000000000000000)

(0.33589264535671831,0.45463309255328066)

The first entries are real and the norm of the coefficients dedicated to each subsystem is one.

The **density matrix** computed from that general pure state is the following. It is noticeable that the trace is close to one, as it should be. A check confirmed that it is hermitian.

NUMBER OF ROWS: 4
NUMBER OF COLUMNS: 4
ELEMENTS:
FIRST ROW
(3.88554304700659547E-002,0.0000000000000000)
(0.15481315421742758,3.10431283991087041E-002)
(1.58213296290961557E-002,2.14142825602566186E-002)
(4.59288083830653587E-002,9.79620132766160362E-002)
SECOND ROW
(0.15481315421742758,-3.10431283991087041E-002)
(0.64162945148063988,0.0000000000000000)
(8.01462299918658805E-002,7.26814508852648561E-002)
(0.26126157730853028,0.35361940935319569)
THIRD ROW
(1.58213296290961557E-002,-2.14142825602566186E-002)
(8.01462299918658805E-002,-7.26814508852648561E-002)
(1.82441929024348103E-002,0.0000000000000000)
(7.26910245288126255E-002,1.45760017571986630E-002)
FOURTH ROW
(4.59288083830653587E-002,-9.79620132766160362E-002)
(0.26126157730853028,-0.35361940935319569)
(7.26910245288126255E-002,-1.45760017571986630E-002)
(0.30127092514685916,0.0000000000000000)
TRACE: (0.99999999999999989,0.0000000000000000)

The **reduced density matrix** computed from the initial density matrix, tracing off the second sub-system, is the following. It is hermitian and with trace close to one, as it should be.

NUMBER OF ROWS: 2
NUMBER OF COLUMNS: 2
ELEMENTS:
FIRST ROW:
(5.70996233725007685E-002,0.0000000000000000)
(0.22750417874624021,4.56191301563073637E-002)
SECOND ROW:
(0.22750417874624021,-4.56191301563073637E-002)

(0.94290037662749904,0.0000000000000000)
TRACE: (0.99999999999999978,1.4821969375237396E-323)

Self-Evaluation

Efficiency in storing the wavefunctions

The minimum number of coefficents is stored for both the general and separable pure state. Hence D^N for the general and $N \cdot D$ for the separable. They are stored in two vectors, so we don't have to manage tensors of higher rank. The meaningless degrees of freedom are gauged: norm one for the D^N vector, norm one for each subsystem contained in the $N \cdot D$ vector, first entry real in the D^N vector and first entries real for each subsystem contained in the $N \cdot D$ vector. If we want to enlarge the $N \cdot D$ vector in a D^N vector, the apposite subroutine $fromsep\_togen$ does the work. A critic is that the code is not generalized to mixed state, so generalizing the initialization including also mixed state would cost some effort. Another critic can be the fact that for very long vectors it should be better to split the vector in different parts. In fact, for very high N, D^N becomes very large, also respect to $N \cdot D$. As a consequence, the computational coste required in storing a general pure state becomes very important, while for a separable pure state it remains reasonable.

The main goals achieved:

- Dealing with pure states, general and separable
- Computing and dealing density matrices and reduced density matrices.

What can be done next:

- Generalize the algorithm, dealing also with mixed states
- Adding the check on the positiveness of the density matrix.

Documentation

Listing 5: Fortran code: Documentation

```
C  ================================================C
   ================================================
C
C       subroutine init_state (psi, nn, dd, debug,
   sepp, purr)
C
C       .. Scalar Arguments ..
C       integer           nn
C       integer           dd
```

```
C       logical           debug
C       logical           sepp
C       logical           purr
C       ..
C       .. Array Arguments ..
C       type(state)     psi

C
C
C         Purpose
C         =======
C
C       \details \b Purpose:
C       \verbatim
C
C       !This subroutine initializes a type state
!with random real numbers between 0 and 1
!If sepp.eqv .true. then the state is separable
!hence a nn*dd vector is initialized
!the first coefficient of each wavefunction
!of one subsystem is set to real
!each single subsystem wavefunction
!is normalized to one
!If sepp.eqv .false. then the state is not separable
!hence a dd**n vector is initialized
!the first coefficient of the wavefunction
!of one subsystem is set to real
!the wavefunction is normalized to one
!The state and the norms are printed on terminal
!if .debug. is .true.
C
C        Arguments:
C         ==========
C

C        \param[out] psi
C        \verbatim
C          psi is type(state)
C                 the state
C        \endverbatimm
C
C        \param[in] nn
C        \verbatim
C          nn is integer
C                number subsystems
```

```
C         \endverbatimm
C
C         \param[in] dd
C         \verbatim
C           dd is integer
C                  number dimension of Hilbert space
C         \endverbatim
C
C         \param[in] debug
C         \verbatim
C           debug is logical
C         \endverbatim
C
C         \param[in] sepp
C         \verbatim
C           sepp is logical
C                if .true. it is separable
C         \endverbatim
C
C         \param[in] purr
C         \verbatim
C           purr is logical
C                if .true. it is pure
C         \endverbatim
C
C  ================================================C
   ================================================
C
C       subroutine denmat_pure(psi, denmat, debug)
C
C       .. Scalar Arguments ..
C       logical           debug
C       ..
C       .. Array Arguments ..
C       type(state)     psi
C       type(dcm)       denmat
C
C
C
C         Purpose
C         =======
C
C       \details \b Purpose:
C       \verbatim
C
```

```
      !this subroutine computes the
      !density matrix of a general pure state
      !of N subystems, each one belonging
      !to a D-dimensional Hilbert space
      !hence the density matrix is D**N x D**N
      !If the psi is separable, then the subroutine
      !fromsep_togen is called, in order to obtain
      !a D**N vector
      C
      C        Arguments:
      C         ==========
      C


      C        \param[in] psi
      C        \verbatim
      C          psi is type(state)
      C                the state
      C        \endverbatimm
      C
      C        \param[out] denmat
      C        \verbatim
      C          denmat type(dcm)
      C                 the computed density matrix
      C        \endverbatimm
      C
      C        \param[in] debug
      C        \verbatim
      C          debug is logical
      C        \endverbatim
      C
      C
      C  ================================================C
         ================================================
      C      subroutine      fromsep_togen(psi,psi_large)
      C
      C      .. Array Arguments ..
      C      type(state)     psi
      C      type(state)     psi_large
      C
      C
      C
      C        Purpose
      C        =======
      C
      C      \details \b Purpose:
```

```
C       \verbatim
C
!this subroutine takes a separable state
!which information is stored in nn*dd vector psi
!and gives the psi_large dd**nn vector in the multi-
   state basis
C
C        Arguments:
C         ==========
C

C        \param[in] psi
C        \verbatim
C          psi is type(state)
C                the separable state
C        \endverbatimm
C
C        \param[out] psi_large
C        \verbatim
C          psi is type(state)
C                the enlarged vector
C        \endverbatimm
C
C
C
C  ================================================C
   ================================================
C
C     subroutine red_denmat(denmat_tot, denmat_red,
   dd, nn, kk, debug)
C
C       .. Scalar Arguments ..
C       integer           dd
C       integer           nn
C       integer           kk
C       logical           debug
C       ..
C       .. Array Arguments ..
C       type(dcm)       denmat_tot
C       type(dcm)       denmat_red
C
C
C         Purpose
C         =======
C
```

```
C       \details \b Purpose:
C       \verbatim
C
!This subroutine computes the reduced
!density matrix of a general density matrix
!in a dd**nn-dimensional Hilbert space
!The trace is performed on the kk-th subsystem
!Hence the reduced density matrix
!is in a dd**(nn-1)-dimensional Hilbert space
C
C        Arguments:
C         ==========
C

C        \param[in] denmat_tot
C        \verbatim
C          denmat_tot type(dcm)
C                  the initial density matrix
C        \endverbatimm
C
C        \param[out] denmat_red
C        \verbatim
C          denmat_ed type(dcm)
C                  the reduced density matrix
C        \endverbatimm
C
C        \param[in] dd
C        \verbatim
C          dd is integer
C                  number dimension of Hilbert space
C        \endverbatim
C
C        \param[in] nn
C        \verbatim
C          nn is integer
C                  number of sub-systems
C        \endverbatim
C
C        \param[in] kk
C        \verbatim
C          kk is integer
C                  number the traced off sub-system
C        \endverbatim
C
C        \param[in] debug
```

```
C         \verbatim
C           debug is logical
C         \endverbatim
C
C
C  ================================================C
   ================================================
```