3_WernerPPT

May 31, 2021

1 Werner state separability

Initialize the RepLAB toolbox (be in the /replab directory or use `run path/replab/replab_init.m`)

[1]:
```
replab_init
replab.globals.useReconstruction(1); % use new algorithms for decomposition
```

```
Adding RepLAB to the path
Initializing dependency vpi
Initializing dependency YALMIP
Initializing dependency sdpt3
Adding embedded SDPT3 solver to the path
Initializing dependency MOcov
Initializing dependency MOxUnit
Initializing dependency cyclolab
```

Declare the symmetry group: $G = \mathcal{U}(2)$ acting on each subsystem. We do not implement the permutation of subsystems as it complicates the PPT constraint.

[2]:
```
G = replab.U(2);
```

This is the representation leaving the Werner state invariant.

[3]:
```
rep = kron(G.definingRep, G.definingRep);
```

Now, the partially transposed state has a "transpose" on the second subsystem; i.e. we take the conjugate representation.

[4]:
```
repT = kron(G.definingRep, conj(G.definingRep));
```

We define spaces of Hermitian matrices invariant under the representations. Those are spaces of matrices that transform as $X \to \rho(g) X \rho(g)^\dagger$ for each of the representations we defined.

[5]:
```
H = rep.hermitianInvariant
HT = repT.hermitianInvariant
```

```
H =

4 x 4 matrices representing an equivariant Hermitian form over C
```

```
  field: 'C'
  group: Unitary group U(2)
     nC: 4
     nR: 4
   repC: Unitary tensor representation
   repR: Unitary derived representation (conjugate) (inverse) (transpose)
special: 'hermitian'

HT =

4 x 4 matrices representing an equivariant Hermitian form over C
  field: 'C'
  group: Unitary group U(2)
     nC: 4
     nR: 4
   repC: Unitary tensor representation
   repR: Unitary derived representation (conjugate) (inverse) (transpose)
special: 'hermitian'
```

We define the partial transpose linear map. We could have used http://www.qetlab.com/PartialTranspose as well. Note that we permute the indices 1,3 as Matlab's reshape does not follow the kron convention (long story).

[6]:
```
ptFun = @(X) reshape(permute(reshape(X, [2 2 2 2]), [3 2 1 4]), [4 4]);
```

We tell RepLAB that the partial transpose is a super operator from the space H to the space HT; by doing so, RepLAB knows that the operator is compatible with the action of the group (`H` and `HT` must have been defined using representations of the same group to use this syntax).

[7]:
```
pt = replab.equiop.generic(H, HT, ptFun)
```

```
pt =

replab.evar.equiop_generic
              f: @(X) reshape (permute (reshape (X, [2, 2, 2, 2]), [3, 2, 1, …
          group: Unitary group U(2)
         source: 4 x 4 matrices representing an equivariant Hermitian form ov…
sourceInjection: replab.mrp.Identity
 supportsSparse: false
         target: 4 x 4 matrices representing an equivariant Hermitian form ov…
targetInjection: replab.mrp.Identity
```

Define the singlet state as an invariant matrix (equivariant variable in the math jargon).

[8]:
```
singlet = replab.equivar(H, 'value', [0 0 0 0; 0 1 -1 0; 0 -1 1 0; 0 0 0 0]/2)
```

```
singlet =
```

```
replab.equivar
     blocks: 2 x 2 cell
equivariant: 4 x 4 matrices representing an equivariant Hermitian form over C
```

Same for the noise

[9]:
```
noise = replab.equivar(H, 'value', eye(4)/4)
```

```
noise =

replab.equivar
     blocks: {0.25, 1 x 1 x 0 empty double array; 1 x 1 x 0 empty double arra…
equivariant: 4 x 4 matrices representing an equivariant Hermitian form over C
```

The visibility is a standard sdpvar from YALMIP.

[10]:
```
t = sdpvar;
```

Below, the syntax `equivar*sdpvar` works, but not `sdpvar*equivar`, which is why the scalar is on the right (as `sdpvar` would provide the `*` operator and doesn't handle equivar)

[11]:
```
rho = singlet*t + noise*(1-t);
```

We use the syntax `sdp(X)` to define a semidefinite positive constraint, where `X` is an equivar.

[12]:
```
C = [issdp(rho)
     issdp(pt(rho))]
```

```
+-----------------------------------------------------------+
|   ID|                    Constraint|   Coefficient range|
+-----------------------------------------------------------+
|   #1|   Element-wise inequality 1x1|        0.25 to 0.75|
|   #2|   Element-wise inequality 1x1|        0.25 to 0.25|
|   #3|   Element-wise inequality 1x1|        0.25 to 0.75|
|   #4|   Element-wise inequality 1x1|        0.25 to 0.25|
+-----------------------------------------------------------+
```

note that this is a linear program now

[13]:
```
optimize(C, -t, sdpsettings('solver', 'sdpt3')) % force SDPT3 the default␣
↪Octave solver has problems
```

```
 num. of constraints =  1
 dim. of linear var  =  4
*****************************************************************
   SDPT3: Infeasible path-following algorithms
*****************************************************************
```

```
 version  predcorr  gam  expon  scale_data
   HKM      1      0.000   1        0
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj    cputime
-------------------------------------------------------------------
 0|0.000|0.000|5.0e-01|1.3e+01|4.0e+02| 1.000000e+01  0.000000e+00| 0:0:00| chol
1  1
 1|1.000|1.000|1.2e-08|1.0e-01|1.2e+01| 9.385230e+00  5.200026e-02| 0:0:00| chol
1  1
 2|0.987|1.000|6.1e-08|1.0e-02|5.7e-01| 6.327589e-01  7.917588e-02| 0:0:00| chol
1  1
 3|1.000|0.902|2.3e-08|1.9e-03|1.1e-01| 4.276703e-01  3.241188e-01| 0:0:00| chol
1  1
 4|0.987|0.987|6.1e-09|1.2e-04|1.4e-03| 3.345748e-01  3.333157e-01| 0:0:00| chol
1  1
 5|0.989|0.989|1.1e-10|1.1e-05|1.5e-05| 3.333470e-01  3.333430e-01| 0:0:00| chol
1  1
 6|0.989|0.989|8.2e-12|1.2e-07|1.7e-07| 3.333335e-01  3.333334e-01| 0:0:00| chol
1  1
 7|0.996|1.000|8.2e-14|1.6e-12|2.4e-09| 3.333333e-01  3.333333e-01| 0:0:00|
  stop: max(relative gap, infeasibilities) < 1.00e-07
-------------------------------------------------------------------
 number of iterations   =  7
 primal objective value =  3.33333335e-01
 dual   objective value =  3.33333333e-01
 gap := trace(XZ)       = 2.38e-09
 relative gap           = 1.43e-09
 actual relative gap    = 1.43e-09
 rel. primal infeas (scaled problem)   = 8.19e-14
 rel. dual     "        "       "      = 1.64e-12
 rel. primal infeas (unscaled problem) = 0.00e+00
 rel. dual     "        "       "      = 0.00e+00
 norm(X), norm(y), norm(Z) = 1.3e+00, 3.3e-01, 6.2e-01
 norm(A), norm(b), norm(C) = 2.1e+00, 2.0e+00, 1.5e+00
 Total CPU time (secs)  = 0.06
 CPU time per iteration = 0.01
 termination code       =  0
 DIMACS: 8.2e-14  0.0e+00  2.0e-12  0.0e+00  1.4e-09  1.4e-09
-------------------------------------------------------------------
ans =

  scalar structure containing the fields:

    yalmipversion = 20200930
    matlabversion = 6.2.0
    yalmiptime = 0.090993
    solvertime = 0.4646
    info = Successfully solved (SDPT3-4)
    problem = 0
```

The separability threshold is:

[14]:
```
double(t)
```

```
ans = 0.3333
```