1_Intro

May 31, 2021

1 Introduction: groups and representations

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

[1]:
```
replab_init
```

```
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
```

Construct the symmetric group.

[2]:
```
S3 = replab.S(3)
```

```
S3 =

Symmetric group acting on 3 elements
    domainSize: 3
generatorNames: {'x1', 'x2'}
      identity: [1, 2, 3]
          type: Symmetric group acting on 3 elements
  generator(1): [2, 3, 1]
  generator(2): [2, 1, 3]
     recognize: AtlasResult (Dihedral group of order 6)
```

Let's take a random element from the group; it's a permutation row vector

[3]:
```
S3.sample
```

```
ans =

   3   2   1
```

Construct the representation that acts on $\vec{P} = (P(1), P(2), P(3))$ in two different ways: the first one is a shortcut, the second one uses explicit images. We check the soundness of our construction.

[4]:
```
rho = S3.naturalRep;
rho = S3.repByImages('R', 3, 'preimages', {[2 3 1] [2 1 3]}, 'images', {[0 0 1;
↪1 0 0; 0 1 0] [0 1 0; 1 0 0; 0 0 1]})
rho.check % Run automated tests

% this one is wrong
% rho = S3.repByImages('R', 3, 'preimages', {[2 3 1] [2 1 3]}, 'images', {[0 1
↪0; 0 0 1; 1 0 0] [0 1 0; 1 0 0; 0 0 1]})
% rho.check
```

```
rho =

Orthogonal representation
           dimension: 3
divisionAlgebraName: []
               field: 'R'
               group: Symmetric group acting on 3 elements
   imagesErrorBound: [0, 0]
          isUnitary: true
           morphism: replab.mrp.PermToFiniteGroup
       preimages{1}: [2, 3, 1]
          images{1}: [0, 0, 1; 1, 0, 0; 0, 1, 0]
       preimages{2}: [2, 1, 3]
          images{2}: [0, 1, 0; 1, 0, 0; 0, 0, 1]

Checking commutes with commutant algebra…
Checking composition…
Checking identity…
Checking matrixColAction…
Checking matrixRowAction…
Checking respects division algebra…
Checking unitary…
Checking withTorusImage->torusImage…
```

Let's take the image of a group element. It's the corresponding permutation matrix.

[5]:
```
g = [3 2 1];
rho.image(g)
```

```
ans =

   0   0   1
   0   1   0
   1   0   0
```

Finally, let us look at the invariant subspaces.

[6]:
```
dec = rho.decomposition
```

```
dec =

Orthogonal reducible representation
           dimension: 3
divisionAlgebraName: []
               field: 'R'
               group: Symmetric group acting on 3 elements
 injection_internal: 3 x 3 double
        isSimilarRep: true
           isUnitary: true
     mapsAreAdjoint: true
              parent: Orthogonal representation
projection_internal: 3 x 3 double
  basis(1,'double'): [0.57735; 0.57735; 0.57735]
  basis(2,'double'): [0.8165; -0.40825; -0.40825]
  basis(3,'double'): [-1.0084e-16; 0.70711; -0.70711]
        component(1): Isotypic component R(1) (trivial)
        component(2): Isotypic component R(2) (nontrivial)
```

The representation has an invariant subspace `[1,1,1]` and the subspace `[2 -1 -1; 0 1 -1]` is invariant as well.

1.1 Manipulating representations

Now, imagine the same group is acting on $P(a_1, a_2) \in \mathbb{R}^9$ which represents two successive outcomes of the box. We could construct `rho2` by computing the explicit images, but the tensor product of the representation works as well here.

[7]:
```
rho2 = kron(rho, rho) % tensor product
```

```
rho2 =

Orthogonal tensor representation
           dimension: 9
divisionAlgebraName: []
               field: 'R'
               group: Symmetric group acting on 3 elements
           isUnitary: true
           factor(1): Orthogonal representation
           factor(2): Orthogonal representation
```

[8]:
```
dec2 = rho2.decomposition
```

```
dec2 =

Orthogonal reducible representation
          dimension: 9
divisionAlgebraName: []
              field: 'R'
              group: Symmetric group acting on 3 elements
 injection_internal: 9 x 9 double
       isSimilarRep: true
          isUnitary: true
     mapsAreAdjoint: true
             parent: Orthogonal tensor representation
projection_internal: 9 x 9 double
  basis(1,'double'): [0.57735; 0; 0; 0; 0.57735; 0; 0; 0; 0.57735]
  basis(2,'double'): 9 x 1 double
  basis(3,'double'): 9 x 1 double
  basis(4,'double'): 9 x 1 double
  basis(5,'double'): 9 x 1 double
  basis(6,'double'): 9 x 1 double
  basis(7,'double'): 9 x 1 double
  basis(8,'double'): 9 x 1 double
  basis(9,'double'): 9 x 1 double
       component(1): Isotypic component I(2)xR(1) (trivial)
       component(2): Isotypic component R(1) (nontrivial)
       component(3): Isotypic component I(3)xR(2) (nontrivial)
```

Let's look at the discovered basis. The first two columns correspond to the two copies of the trivial representation. Those are invariant vectors. The third column is also an invariant subspace. The remaining columns decompose in a more complex way (three copies of an irreducible representation of dimension 2, the standard representation of $\mathcal{S}_3$).

[9]:
```
dec2.basis
```

```
ans =

 Columns 1 through 8:

   0.5774        0  -0.0000   0.0302  -0.0547   0.2007  -0.3635  -0.3385
        0   0.4082  -0.4082   0.3041   0.4888  -0.4336   0.2433  -0.2299
        0   0.4082   0.4082  -0.5757   0.0030   0.0251   0.4965  -0.0364
        0   0.4082   0.4082   0.2852  -0.5000  -0.4426  -0.2265  -0.2369
   0.5774        0   0.0000   0.0323   0.0535   0.2144   0.3555  -0.3617
        0   0.4082  -0.4082  -0.5753   0.0190   0.0061  -0.4971  -0.0477
        0   0.4082  -0.4082   0.2712  -0.5078   0.4274   0.2538   0.2776
        0   0.4082   0.4082   0.2905   0.4970   0.4174  -0.2700   0.2734
   0.5774        0  -0.0000  -0.0625   0.0012  -0.4151   0.0080   0.7001
```

```
Column 9:

  0.6131
  0.1878
  0.2946
 -0.1789
 -0.5996
 -0.2930
  0.1052
 -0.1158
 -0.0134
```

[ ]: