Functional framework for representing and transforming quantum channels

Jarosław Miszczak

Institute of Theoretical and Applied Informatics, Polish Academy of Sciences

ACA2013, Malaga, July 2013

Motivation

Initial problem

Problem: Distribution of *product* eigenvalues for random 2-qubit states.[1] Exact formula exists for the spectrum, but only numerical approximation is possible for the local spectrum.

Solution: *Mathematica* = functional-based programming language + powerful symbolic and numeric capabilities. Symbolic algebra for the eigenvalues and random states for the product eigenvalues.

[1] P. Gawron, Z. Puchała, J.A.M, Ł. Skowronek, K. Życzkowski, *Restricted numerical range: a versatile tool in the theory of quantum information*, J. Math. Phys., Vol. 51, No. 10 (2010), pp. 102204. arXiv:0905.3646

The framework (or package) should:

- provide a simplified method for the representation of quantum channels,
- allow the manipulation of quantum channels on non-qubit systems,
- provide the means for converting between different representations of quantum channels.

Implementation

Design assumptions

We aim to utilize the functional language offered by *Mathematica*, thus we follow some rules:

- main assumption: quantum channels are just functions and thus can be used as a first-class objects,
- use currying – `f[a][b]` instead of `f[a,b]` – to partially apply functions,
- use **`Map`** and **`Fold`** instead of looping and tables.

Implementation

Functions as arguments

Functions are first-class objects and can be used as function arguments.

Example: transposition map

It can be expressed in *Mathematica* as

```
trans = Function[x, Transpose[x]]
```

Such defined function can be mapped on a list of matrices as

```
Map[f, mtxs]
```

or nested k-times

```
Nest[f, mtx, k]
```

Problem with partial application

In *Mathematica* one has to explicitly use empty slots (`#`-signs) to define a partially applied function, *e.g.* if we have `f[a,b]`, then `f[a,#]` gives partial application of `f`.

For this reason in order to use the functional version of some procedures, it is necessary to provide a curried version of these functions.

Example: depolarizing channel

We define depolarizing channel for one qubit as `dep2[p][x]`, not as `dep2[p,x]`

Examples

Natural representation

Channels are linear mappings, thus it is possible to represent them by matrices. The natural representation is given by the unreshaping of the images of the map on the base matrices in $\mathbb{M}_{d^2}$.

```
NaturalRepresentation = Function[f, Function[d,
    With[{base=BaseMatrices[d^2]}, Map[Res[f[#]]&, base]]]];
```

Example: transposition

One can check if the natural representation of the one-qubit transposition channel `trans`

```
NaturalRepresentation[trans][2]
```

is equal to the SWAP gate.

`NaturalRepresentation` is occasionally called `Superoperator`.

Using the above notation we can easily use channels without taking care of the dimension.

Example: depolarizing channel

Depolarizing channel acts on a density matrix ρ as

$$D_{p,n}(\rho) = (1-p)\rho + p/n\mathbb{1}_n,$$

or in *Mathematica* as

```
dep[p][n][r] = Function[p, Function[n, Function[r,
    (1-p) r + p/n Id[n] ]]]
```

For a fixed dimension we get its matrix form as

```
NaturalRepresentation[trans][3]
```

Examples
Product channels

Tensor product of two operators has the natural representation given by

$$M_{\Phi\otimes\Psi} = M_{R^{-1}}(M_\Phi \otimes M_\Psi)M_R,$$

where M_R exchanges between the canonical and the tensor-product bases.

Example: partial transposition

If we take $\Psi = \mathbb{1}$ and $\Phi = \cdot^T$ we obtain the partial transposition

```
partTransA = ProductSuperoperator[
    NaturalRepresentation[trans][2], Id[4]]
```

Tensor product of two operators has the natural representation given by

$$M_{\Phi\otimes\Psi} = M_{R^{-1}}(M_\Phi \otimes M_\Psi)M_R,$$

where M_R exchanges between the canonical and the tensor-product bases.

Example: partial transposition

If we take $\Psi = \mathbb{1}$ and $\Phi = \cdot^T$ we obtain the partial transposition

```
partTransA =  ProductSuperoperator[
    NaturalRepresentation[trans][2], Id[4]]
```

Problem: calculation of M_R

Matrix M_R (on $n \times n$) can be calculated using n^4 matrix multiplications, which is very inefficient. There is much faster method based on index manipulation.

Examples

Conversion between representations

Quantum channels can be represented in different, equivalent forms: Kraus form, natural representation, dynamical matrix.

Example: dynamical matrix

Dynamical matrix is obtained by reshuffling the natural representation

```
dynTrans = Reshuffle[NaturalRepresentation[trans][2]]
```

Example: Kraus form

Kraus operators are formed by reshaping eigenvectors of the dynamical matrix

```
{val, vec} = Eigensystem[dynTrans];
krausTrans = Sqrt[val] (Unvec[#/Norm[#]] & /@ vec);
```

Examples

Summary

- Functional code
  - easy to write and maintain
  - seamless transition between representations of channels
- Implementation based on index manipulation is much faster
- *Mathematica*
  - great numerical and symbolical capabilities
  - QI package[2]:
    `http://zksi.iitis.pl/wiki/projects:mathematica-qi`

[2] JAM, *Singular value decomposition and matrix reorderings in quantum information theory*, Int. J. Mod. Phys. C, Vol. 22, No. 9 (2011), pp. 897-918.

Quantiki project

Wiki and portal of quantum information

- Portal and encyclopedia of quantum information
  - list of groups, job offers, conferences
  - catalog of quantum computing simulators `http://quantiki.org/wiki/List_of_QC_simulators`
  - video abstracts `http://quantiki.org/video_abstracts` and `http://youtube.com/user/QuantikiVideo`
  - community-edited wiki with encyclopedia `http://quantiki.org/wiki`
- Cooperation with National University of Singapore
- More at `http://quantiki.org/`

Thank you!

http://iitis.pl/~miszczak/talks/