Tomography Documentation

Release 0.1

Tjeerd Fokkens, Andreas Fognini, Val Zwiller

Sep 14, 2016

Contents:

CHAPTER

ONE

INTRODUCTION

This implementation of the density matrix reconstruction follows closely the method described in D. F. V. James et al. Phys. Rev. A, 64, 052312 (2001). We tried to keep the naming in the code as similar as possible to that reference.

Measurement

To reconstruct the density matrix of a two qubit photonic system we need to perform 16 coincidence measurements between the two qubits in 16 different polarization projections. A valid basis set for the reconstruction is for example:

```
basis =  ["HH","HV","VV","VH","HD","HR","VD","VR","DH","DV","DD","DR","RH","RV","RD",
↪"RR"]
```

In this example the first coincidence measurement is performed for the basis element "HH". This means that for both photons, the horizontal component of the polarization is measured.

In the second coincidence measurement, "HV", we measure the horizontal component of the first photon, and the vertical polarization component of the second one.

Usage of the library

Once all the coincidence counts have been collected the density matrix describing the quantum system can be reconstructed.

```
import numpy as np
from Tomography import DensityMatrix

round_digits = 2
dm = DensityMatrix(basis= ["HH","HV","VV","VH","RH","RV","DV","DH","DR","DD","RD","HD
↪","VD","VL","HL","RL"])
```

At first, the DensityMatrix class is initialized with the measurement basis. The measured correlation counts in this basis set is given to the function rho which calculates the density matrix.

```
#Compute the raw density matrix, data from: D. F. V. James et al. Phys. Rev. A, 64,␣
↪052312 (2001).
cnts = np.array([34749, 324, 35805, 444, 16324, 17521, 13441, 16901, 17932, 32028,␣
↪15132, 17238, 13171, 17170, 16722, 33586])
rho  = dm.rho(cnts)
```

However, due to measurement imperfections this matrix is not necessarily positive semidefinite. To circumvent this problem a maximum likelihood estimation as described in the aforementioned paper is implemented. Please note

that we initialize the t parameters in our implementation with only ones. In this way, the algorithm can also handle states like HH, since the Cholesky decomposition can effectively be computed. We get the reconstructed positive semidefinite density matrix by evoking:

```
rho_recon=dm.rho_max_likelihood(rho, cnts)
```

The library can also compute some quantum measures on the density matrix like the concurrence:

```
concurrence=dm.concurrence(rho_recon)
```

or the fidelity:

```
f=dm.fidelity_max(rho_recon)
```

The fidelity is calculated here to a maximally entangled state. We used the algorithm described in http://dx.doi.org/10.1103/PhysRevA.66.022307.

In quantum tomography not only the density matrix is of interest but also the pure state which most likely characterizes the system. This is only reasonable if the density matrix reconstructed is already close to a pure state, i.e. its fidelty or concurrence is close to unity.

The following code block is an example how to reconstruct a pure state with this library:

```
closest_state_basis =["HH","HV","VH","VV"]
closest_state = dm.find_closest_pure_state(rho_recon, basis=closest_state_basis)

s = str()
for i in range(3):
        s = s + "\t"+ str(closest_state[i]) + "\t|"+closest_state_basis[i] + "> + \n"

s = s + "\t" + str(closest_state[3]) + "\t|" + closest_state_basis[3] + ">"

print("Closest State: \n" + s + "\n")
```

The density matrix from any pure state can also easily be constructed. For example from the following Bell state: $\frac{1}{\sqrt{2}}(|HH\rangle + i|VV\rangle)$.

```
HH =dm.state("HH")
VV =dm.state("VV")

print(dm.rho_state(state=1/np.sqrt(2)*(HH+1j*VV)))
```

Error estimation

The error estimation is performed based on a Monte Carlo simulation. Each correlation count is assumed to be subjected to counting statistics. Thus, the measured number N of correlation counts will be replaced in each step of the simulation with a draw from a normal distribution with standard deviation $\sigma = \sqrt{N}$ and mean $\mu = N$. In each simulation step a new density matrix is calculated. Based on this set of simulated density matrices the standard deviation can be computed to estimate the error.

To get the error of the above examples do:

```
import numpy as np
from Tomography import Errorize
round_digits = 2
```

```
basis= ["HH", "HV","VV","VH","RH","RV","DV","DH","DR","DD","RD","HD","VD","VL","HL",
↪"RL"]
cnts = np.array([34749, 324, 35805, 444, 16324, 17521, 13441, 16901, 17932, 32028,␣
↪15132, 17238, 13171, 17170, 16722, 33586])
#Data from: D. F. V. James et al. Phys. Rev. A, 64, 052312 (2001).

err = Errorize(basis = basis, cnts = cnts)
err.multiprocessing_simulate(n_cycles_per_core  = 10, nbr_of_cores = 2)

rho_err = err.rho_max_likelihood()

print("Uncertainty of rho: \n" + str(np.around(rho_err, decimals =round_digits)) + "\n
↪")

#Uncertainty of fidelity and concurrence estimates
fid_err=err.fidelity_max()
con_err=err.concurrence()

print("fid_err: \n" + str(fid_err) + "\n")
print("con_err: \n" + str(con_err) + "\n")
```

CHAPTER

TWO

INSTALLATION

Before you start the installation of the library it is advised that you have installed numpy (numpy.org) and scipy (SciPy.org) packages already.

To install the library open a terminal and navigate into the Tomography folder. Then install it either by:

```
python setup.py install
```

Or by:

```
python3 setup.py install
```

Depending on your python installation. Please note that this library needs python3.x .

You can test if the installation worked by importing the library:

```
import Tomography
```

If you not getting an error message, everything works.

CHAPTER

THREE

MODULES DESCRIPTION

In the following the Tomography module is described. It consists of two classes:

```
* DensityMatrix
* Errorize
```

The DensityMatrix class is used to calculate the density matrix, some quantum measures, and the pure state closest to the found density matrix. The Errorize class is used to compute uncertainties of the values computed in the DensityMatrix class by means of a Monte Carlo simulation.

Tomography module

CHAPTER

FOUR

LICENSE

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search