偏光を用いた2量子もつれの量子状態トモグラフィによる評価

62200139 青木 陽

March 17, 2025

Contents

Chapter 1

原理

1　レーザ

WIP

2　ビームスプリッタ

WIP

3　非線型結晶

WIP

4　量子状態トモグラフィ

4.1　ベクトル空間・内積空間

4.2　SU(2)

4.3　密度行列

4.4　密度行列の再構成

Chapter 2

実験方法

Chapter 3

実験結果

Chapter 4

考察

付録 A

ソースコード

```
import matplotlib.pyplot as plt
import numpy as np
import mpl_toolkits
from matplotlib import cm
from matplotlib.colors import Normalize

class HVDR:
    def __init__(self, l):
        self.l = l
    def __mul__(self, other):
        ret = list()
        for x in self.l:
            tmp = list()
            for y in other.l:
                tmp.append(x * y)
            ret.append(tmp)
        return ret

def calc_sigma_pow():
    sigma_0 = HVDR([1, 1, 0, 0])
    sigma_1 = HVDR([-1, -1, 2, 0])
    sigma_2 = HVDR([-1, -1, 0, 2])
    sigma_3 = HVDR([1, -1, 0, 0])
    sigma = [sigma_0, sigma_1, sigma_2, sigma_3]
    sigma_pow = [[None for _ in range(4)] for _ in range(4)]
    for i in range(4):
        for j in range(4):
            sigma_pow[i][j] = sigma[i] * sigma[j]
    return sigma_pow
def load_data(path):
    polar_dict = {'H': 0, 'V': 1, 'D': 2, 'R': 3}
    raw_data = np.loadtxt(path, dtype = "str", delimiter = ',')
    data = [[None for _ in range(4)] for _ in range(4)]
    for i in range(1, len(raw_data)):
        data[polar_dict[raw_data[i][0]]][polar_dict[raw_data[i][1]]] = int(raw_data[i][2])
    return data
def calc_uij(sigma_pow, data):
    u = [[0 for _ in range(4)] for _ in range(4)]
    for i1 in range(4):
        for j1 in range(4):
            for i2 in range(4):
                for j2 in range(4):
```

```
                    u[i1][j1] += sigma_pow[i1][j1][i2][j2] * data[i2][j2]
    return u
def calc_sigma_matrix_tensor():
    sigma_num_0 = np.array([[1, 0], [0, 1]])
    sigma_num_1 = np.array([[0, 1], [1, 0]])
    sigma_num_2 = np.array([[0, -1j], [1j, 0]])
    sigma_num_3 = np.array([[1, 0], [0, -1]])
    sigma_num = [sigma_num_0, sigma_num_1, sigma_num_2, sigma_num_3]
    sigma_num_pow = [[None for _ in range(4)] for _ in range(4)]
    for i in range(4):
        for j in range(4):
            sigma_num_pow[i][j] = np.kron(sigma_num[i], sigma_num[j])
    return sigma_num_pow
def estimate_rho(u, sigma_num_pow):
    rho = [[0 for _ in range(4)] for _ in range(4)]
    rho_trace = 0
    for i1 in range(4):
        for j1 in range(4):
            for i2 in range(4):
                for j2 in range(4):
                    rho[i2][j2] += u[i1][j1] * sigma_num_pow[i1][j1][i2][j2]
    for i in range(4): rho_trace += rho[i][i]
    rho_norm = [[rho[i][j] / rho_trace for j in range(4)] for i in range(4)]
    rho_norm = np.array(rho_norm)
    return rho_norm
def make_graph(rho_norm, data_name):
    rho_real_imag = [rho_norm.real, rho_norm.imag]
    tmp_x = np.arange(4)
    tmp_y = np.arange(4)
    tmp_X, tmp_Y = np.meshgrid(tmp_x, tmp_y)
    label_bra = [
        r"$\left| \rm{HH} \right\rangle$",
        r"$\left| \rm{VH} \right\rangle$",
        r"$\left| \rm{HV} \right\rangle$",
        r"$\left| \rm{VV} \right\rangle$"
    ]
    label_ket = [
        r"$\left\langle \rm{HH} \right|$",
        r"$\left\langle \rm{VH} \right|$",
        r"$\left\langle \rm{HV} \right|$",
        r"$\left\langle \rm{VV} \right|$"
    ]
    x = tmp_X.ravel()
    y = tmp_Y.ravel()
    z = np.zeros_like(x)
    dx = dy = 0.5
    for i in range(2):
        fig = plt.figure()
        ax = fig.add_subplot(111, projection="3d")
        dz = rho_real_imag[i].ravel()
        norm = Normalize(vmin=-1, vmax=1)
        colors = cm.coolwarm_r(norm(dz))
        alpha = 0.7
        colors[:, 3] = alpha
        ax.bar3d(x, y, z, dx, dy, dz, color=colors, shade=True)
        ax.set_xticks(tmp_x + dx / 2)
        ax.set_xticklabels(label_bra, ha="center")
        ax.set_yticks(tmp_y + dy / 2)
```

```
        ax.set_yticklabels(label_ket, ha="center")
        ax.set_zlim(-0.75, 0.75)
        ax.zaxis.set_tick_params(labelleft=False, labelright=False, labeltop=False, labelbottom=False)
        mappable = cm.ScalarMappable(norm=norm, cmap="coolwarm_r")
        mappable.set_array(dz)
        fig.colorbar(mappable, ax=ax, shrink=0.6, aspect=10, label=r"$\hat{\rho}$")
        title = data_name + "_" + "riemaalg"[i:: 2]
        ax.set_title(title)
        title += ".pdf"
        plt.savefig(title, bbox_inches = "tight")
def main():
    sigma_pow = calc_sigma_pow()
    sigma_num_pow = calc_sigma_matrix_tensor()
    datas = ["data1", "data2"]
    for i in range(2):
        data = load_data(datas[i] + ".csv")
        u = calc_uij(sigma_pow, data)
        rho_norm = estimate_rho(u, sigma_num_pow)
        make_graph(rho_norm, datas[i])
if (__name__ == "__main__"):
    main()
```