In [1]:
```
import numpy as np
from scipy import linalg as sLA
from numpy import linalg as LA
import copy
import scipy
from scipy.stats import chi2
from matplotlib import pyplot as plt
from matplotlib import style
```

In [2]:
```
style.use('ggplot')
```

Лабораторная работа No3. Реконструкция матрицы плотности методом максимального правдоподобия, оценка адекватности и точности реконструкции

Пункт 1:

Рассмотрим протокол томогрфии квантового состояния, основанный на измерениях во взаимно-несмещённых базисах (MUB) для размености d = 4 и выполним симуляцию измерений:

In [3]:
```
# Массив унитарных матриц образующих MUB для d = 4

mub_array = 1/2*np.array([
    [[2,0,0,0],
     [0,2,0,0],
     [0,0,2,0],
     [0,0,0,2]],

    [[1,1,1,1],
     [1,1,-1,-1],
     [1,-1,-1,1],
     [1,-1,1,-1]],

    [[1,1,1,1],
     [-1,-1,1,1],
     [-1j,1j,1j,-1j],
     [-1j,1j,-1j,1j]],

    [[1,1,1,1],
     [-1j,-1j,1j,1j],
     [-1j,1j,1j,-1j],
     [-1,1,-1,1]],

    [[1,1,1,1],
     [-1j,-1j,1j,1j],
     [-1,1,-1,1],
     [-1j,1j,1j,-1j]],

])
```

In [4]:
```
# Получаем операторы P(j,k) и B из набора матриц, соответствующих заданному MUB
def build_P(mub_array):
    P = []
    B = []
    for mub in mub_array:
        P_part = []
        B_part = []
        for string in mub.T:
            op = np.outer(string, string.conj())
            P_part.append(op)
            B_part.append(np.ravel(op, order = 'C'))
        B.append(B_part)
        P.append(P_part)
    return np.array(P), np.vstack(np.array(B))

# Получаем операторы X из набора матриц, соответствующих заданному MUB
def build_X(mub_array):
    X = np.vstack(copy.deepcopy(mub_array).transpose((0,2,1)).conj())
    return X
```

In [5]:
```
# Составим матрицы согласно заданию
P, B = build_P(mub_array)
X = build_X(mub_array)
X,P,B
```

Out[5]:
```
(array([[ 1. -0.j ,  0. -0.j ,  0. -0.j ,  0. -0.j ],
        [ 0. -0.j ,  1. -0.j ,  0. -0.j ,  0. -0.j ],
        [ 0. -0.j ,  0. -0.j ,  1. -0.j ,  0. -0.j ],
        [ 0. -0.j ,  0. -0.j ,  0. -0.j ,  1. -0.j ],
```

```
        [ 0.5-0.j ,  0.5-0.j ,  0.5-0.j ,  0.5-0.j ],
        [ 0.5-0.j ,  0.5-0.j , -0.5-0.j , -0.5-0.j ],
        [ 0.5-0.j , -0.5-0.j , -0.5-0.j ,  0.5-0.j ],
        [ 0.5-0.j , -0.5-0.j ,  0.5-0.j , -0.5-0.j ],
        [ 0.5-0.j , -0.5-0.j ,  0. +0.5j,  0. +0.5j],
        [ 0.5-0.j , -0.5-0.j ,  0. -0.5j,  0. -0.5j],
        [ 0.5-0.j ,  0.5-0.j ,  0. -0.5j,  0. +0.5j],
        [ 0.5-0.j ,  0.5-0.j ,  0. +0.5j,  0. -0.5j],
        [ 0.5-0.j ,  0. +0.5j,  0. +0.5j, -0.5-0.j ],
        [ 0.5-0.j ,  0. +0.5j,  0. -0.5j,  0.5-0.j ],
        [ 0.5-0.j ,  0. -0.5j,  0. -0.5j, -0.5-0.j ],
        [ 0.5-0.j ,  0. -0.5j,  0. +0.5j,  0.5-0.j ],
        [ 0.5-0.j ,  0. +0.5j, -0.5-0.j ,  0. +0.5j],
        [ 0.5-0.j ,  0. +0.5j,  0.5-0.j ,  0. -0.5j],
        [ 0.5-0.j ,  0. -0.5j, -0.5-0.j ,  0. -0.5j],
        [ 0.5-0.j ,  0. -0.5j,  0.5-0.j ,  0. +0.5j]]),
array([[[[ 1.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ],
         [ 0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ],
         [ 0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ],
         [ 0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ]],

        [[ 0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ],
         [ 0.  +0.j  ,  1.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ],
         [ 0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ],
         [ 0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ]],

        [[ 0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ],
         [ 0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ],
         [ 0.  +0.j  ,  0.  +0.j  ,  1.  +0.j  ,  0.  +0.j  ],
         [ 0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ]],

        [[ 0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ],
         [ 0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ],
         [ 0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ],
         [ 0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  1.  +0.j  ]]],


       [[[ 0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  ],
         [ 0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  ],
         [ 0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  ],
         [ 0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  ]],

        [[ 0.25+0.j  ,  0.25+0.j  , -0.25-0.j  , -0.25-0.j  ],
         [ 0.25+0.j  ,  0.25+0.j  , -0.25-0.j  , -0.25-0.j  ],
         [-0.25+0.j  , -0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  ],
         [-0.25+0.j  , -0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  ]],

        [[ 0.25+0.j  , -0.25-0.j  , -0.25-0.j  ,  0.25+0.j  ],
         [-0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  , -0.25+0.j  ],
         [-0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  , -0.25+0.j  ],
         [ 0.25+0.j  , -0.25-0.j  , -0.25-0.j  ,  0.25+0.j  ]],

        [[ 0.25+0.j  , -0.25-0.j  ,  0.25+0.j  , -0.25-0.j  ],
         [-0.25+0.j  ,  0.25+0.j  , -0.25+0.j  ,  0.25+0.j  ],
         [ 0.25+0.j  , -0.25-0.j  ,  0.25+0.j  , -0.25-0.j  ],
         [-0.25+0.j  ,  0.25+0.j  , -0.25+0.j  ,  0.25+0.j  ]]],


       [[[ 0.25+0.j  , -0.25-0.j  ,  0.  +0.25j,  0.  +0.25j],
         [-0.25+0.j  ,  0.25+0.j  , -0.  -0.25j, -0.  -0.25j],
         [ 0.  -0.25j, -0.  +0.25j,  0.25+0.j  ,  0.25+0.j  ],
         [ 0.  -0.25j, -0.  +0.25j,  0.25+0.j  ,  0.25+0.j  ]],

        [[ 0.25+0.j  , -0.25-0.j  ,  0.  -0.25j,  0.  -0.25j],
         [-0.25+0.j  ,  0.25+0.j  ,  0.  +0.25j,  0.  +0.25j],
         [ 0.  +0.25j,  0.  -0.25j,  0.25+0.j  ,  0.25+0.j  ],
         [ 0.  +0.25j,  0.  -0.25j,  0.25+0.j  ,  0.25+0.j  ]],

        [[ 0.25+0.j  ,  0.25+0.j  ,  0.  -0.25j,  0.  +0.25j],
         [ 0.25+0.j  ,  0.25+0.j  ,  0.  -0.25j,  0.  +0.25j],
         [ 0.  +0.25j,  0.  +0.25j,  0.25+0.j  , -0.25+0.j  ],
         [ 0.  -0.25j,  0.  -0.25j, -0.25-0.j  ,  0.25+0.j  ]],

        [[ 0.25+0.j  ,  0.25+0.j  ,  0.  +0.25j,  0.  -0.25j],
         [ 0.25+0.j  ,  0.25+0.j  ,  0.  +0.25j,  0.  -0.25j],
         [ 0.  -0.25j,  0.  -0.25j,  0.25+0.j  , -0.25-0.j  ],
         [ 0.  +0.25j,  0.  +0.25j, -0.25+0.j  ,  0.25+0.j  ]]],


       [[[ 0.25+0.j  ,  0.  +0.25j,  0.  +0.25j, -0.25-0.j  ],
         [ 0.  -0.25j,  0.25+0.j  ,  0.25+0.j  , -0.  +0.25j],
         [ 0.  -0.25j,  0.25+0.j  ,  0.25+0.j  , -0.  +0.25j],
         [-0.25+0.j  , -0.  -0.25j, -0.  -0.25j,  0.25+0.j  ]],

        [[ 0.25+0.j  ,  0.  +0.25j,  0.  -0.25j,  0.25+0.j  ],
         [ 0.  -0.25j,  0.25+0.j  , -0.25-0.j  ,  0.  -0.25j],
         [ 0.  +0.25j, -0.25+0.j  ,  0.25+0.j  ,  0.  +0.25j],
         [ 0.25+0.j  ,  0.  +0.25j,  0.  -0.25j,  0.25+0.j  ]],

        [[ 0.25+0.j  ,  0.  -0.25j,  0.  -0.25j, -0.25-0.j  ],
         [ 0.  +0.25j,  0.25+0.j  ,  0.25+0.j  ,  0.  -0.25j],
```

```
          [ 0.  +0.25j,  0.25+0.j  ,  0.25+0.j  ,  0.  -0.25j],
          [-0.25+0.j  ,  0.  +0.25j,  0.  +0.25j,  0.25+0.j  ]],

         [[ 0.25+0.j  ,  0.  -0.25j,  0.  +0.25j,  0.25+0.j  ],
          [ 0.  +0.25j,  0.25+0.j  , -0.25+0.j  ,  0.  +0.25j],
          [ 0.  -0.25j, -0.25-0.j  ,  0.25+0.j  ,  0.  -0.25j],
          [ 0.25+0.j  ,  0.  -0.25j,  0.  +0.25j,  0.25+0.j  ]]],


        [[[ 0.25+0.j  ,  0.  +0.25j, -0.25-0.j  ,  0.  +0.25j],
          [ 0.  -0.25j,  0.25+0.j  , -0.  +0.25j,  0.25+0.j  ],
          [-0.25+0.j  , -0.  -0.25j,  0.25+0.j  , -0.  -0.25j],
          [ 0.  -0.25j,  0.25+0.j  , -0.  +0.25j,  0.25+0.j  ]],

         [[ 0.25+0.j  ,  0.  +0.25j,  0.25+0.j  ,  0.  -0.25j],
          [ 0.  -0.25j,  0.25+0.j  ,  0.  -0.25j, -0.25-0.j  ],
          [ 0.25+0.j  ,  0.  +0.25j,  0.25+0.j  ,  0.  -0.25j],
          [ 0.  +0.25j, -0.25+0.j  ,  0.  +0.25j,  0.25+0.j  ]],

         [[ 0.25+0.j  ,  0.  -0.25j, -0.25-0.j  ,  0.  -0.25j],
          [ 0.  +0.25j,  0.25+0.j  ,  0.  -0.25j,  0.25+0.j  ],
          [-0.25+0.j  ,  0.  +0.25j,  0.25+0.j  ,  0.  +0.25j],
          [ 0.  +0.25j,  0.25+0.j  ,  0.  -0.25j,  0.25+0.j  ]],

         [[ 0.25+0.j  ,  0.  -0.25j,  0.25+0.j  ,  0.  +0.25j],
          [ 0.  +0.25j,  0.25+0.j  ,  0.  +0.25j, -0.25+0.j  ],
          [ 0.25+0.j  ,  0.  -0.25j,  0.25+0.j  ,  0.  +0.25j],
          [ 0.  -0.25j, -0.25-0.j  ,  0.  -0.25j,  0.25+0.j  ]]]]),
 array([[[ 1.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,
           0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,
           0.  +0.j  ],
         [ 0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,
           1.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,
           0.  +0.j  ],
         [ 0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,
           0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,
           1.  +0.j  ],
         [ 0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,
           0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,  0.  +0.j  ,
           0.  +0.j  ],
         [ 0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  ,
           0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  ,
           0.25+0.j  ],
         [ 0.25+0.j  ,  0.25+0.j  , -0.25-0.j  , -0.25-0.j  ,  0.25+0.j  ,
           0.25+0.j  , -0.25-0.j  , -0.25-0.j  , -0.25+0.j  , -0.25+0.j  ,
           0.25+0.j  ,  0.25+0.j  , -0.25+0.j  , -0.25+0.j  ,  0.25+0.j  ,
           0.25+0.j  ],
         [ 0.25+0.j  , -0.25-0.j  , -0.25-0.j  ,  0.25+0.j  , -0.25+0.j  ,
           0.25+0.j  ,  0.25+0.j  , -0.25+0.j  , -0.25+0.j  ,  0.25+0.j  ,
           0.25+0.j  , -0.25+0.j  ,  0.25+0.j  , -0.25-0.j  , -0.25-0.j  ,
           0.25+0.j  ],
         [ 0.25+0.j  , -0.25-0.j  ,  0.25+0.j  , -0.25-0.j  , -0.25+0.j  ,
           0.25+0.j  , -0.25+0.j  ,  0.25+0.j  ,  0.25+0.j  , -0.25-0.j  ,
           0.25+0.j  , -0.25-0.j  , -0.25+0.j  ,  0.25+0.j  , -0.25+0.j  ,
           0.25+0.j  ],
         [ 0.25+0.j  , -0.25-0.j  ,  0.  +0.25j,  0.  +0.25j, -0.25+0.j  ,
           0.25+0.j  , -0.  -0.25j, -0.  -0.25j,  0.  -0.25j, -0.  +0.25j,
           0.25+0.j  ,  0.25+0.j  ,  0.  -0.25j, -0.  +0.25j,  0.25+0.j  ,
           0.25+0.j  ],
         [ 0.25+0.j  , -0.25-0.j  , -0.  +0.25j,  0.  -0.25j,  0.  -0.25j,
          -0.25+0.j  ,  0.25+0.j  ,  0.  -0.25j,  0.  -0.25j, -0.25-0.j  ,
           0.25+0.j  , -0.25+0.j  ,  0.  -0.25j,  0.  -0.25j, -0.25+0.j  ,
           0.25+0.j  ],
         [ 0.25+0.j  ,  0.  +0.25j,  0.  -0.25j,  0.25+0.j  ,  0.  -0.25j,
           0.25+0.j  , -0.25-0.j  ,  0.  -0.25j,  0.  +0.25j, -0.25+0.j  ,
           0.25+0.j  ,  0.  +0.25j,  0.25+0.j  ,  0.  +0.25j,  0.  -0.25j,
           0.25+0.j  ],
         [ 0.25+0.j  ,  0.  -0.25j,  0.  -0.25j, -0.25-0.j  ,  0.  +0.25j,
           0.25+0.j  ,  0.25+0.j  ,  0.  -0.25j,  0.  +0.25j,  0.25+0.j  ,
           0.25+0.j  ,  0.  -0.25j, -0.25+0.j  ,  0.  +0.25j,  0.  +0.25j,
           0.25+0.j  ],
         [ 0.25+0.j  ,  0.  -0.25j,  0.  +0.25j,  0.25+0.j  ,  0.  +0.25j,
           0.25+0.j  , -0.25+0.j  ,  0.  +0.25j,  0.  -0.25j, -0.25-0.j  ,
           0.25+0.j  ,  0.  -0.25j,  0.25+0.j  ,  0.  -0.25j,  0.  +0.25j,
```

```
  0.25+0.j  ],
[ 0.25+0.j  ,  0.  +0.25j, -0.25-0.j  ,  0.  +0.25j,  0.  -0.25j,
  0.25+0.j  , -0.  +0.25j,  0.25+0.j  , -0.25+0.j  , -0.  -0.25j,
  0.25+0.j  , -0.  -0.25j,  0.  -0.25j,  0.25+0.j  , -0.  +0.25j,
  0.25+0.j  ],
[ 0.25+0.j  ,  0.  +0.25j,  0.25+0.j  ,  0.  -0.25j,  0.  -0.25j,
  0.25+0.j  ,  0.  -0.25j, -0.25-0.j  ,  0.25+0.j  ,  0.  +0.25j,
  0.25+0.j  ,  0.  -0.25j,  0.  +0.25j, -0.25+0.j  ,  0.  +0.25j,
  0.25+0.j  ],
[ 0.25+0.j  ,  0.  -0.25j, -0.25-0.j  ,  0.  -0.25j,  0.  +0.25j,
  0.25+0.j  ,  0.  -0.25j,  0.25+0.j  , -0.25+0.j  ,  0.  +0.25j,
  0.25+0.j  ,  0.  +0.25j,  0.  +0.25j,  0.25+0.j  ,  0.  -0.25j,
  0.25+0.j  ],
[ 0.25+0.j  ,  0.  -0.25j,  0.25+0.j  ,  0.  +0.25j,  0.  +0.25j,
  0.25+0.j  ,  0.  +0.25j, -0.25+0.j  ,  0.25+0.j  ,  0.  -0.25j,
```

Сгенерируем случайное чистое состояние размерности d = 4, проверим правильность полученных матриц:

In [6]:

```
# Класс квантового состояния
class State():

    def __init__(self, d = 4):
        self.d = d
        self.phi = None
        self.rho = None

    def build_clear_state(self, random_state = 42):


        phi = np.random.randn(int(d)) + 1j*np.random.randn(int(d))
        self.phi = phi/ np.sqrt((sum(abs(phi) ** 2))) #np.sqrt(np.dot(phi, phi.conj()))
        self.rho = np.outer(self.phi, self.phi.conj())

    def set_phi(self, coefs):
        self.phi = coefs
        self.rho = np.outer(self.phi, self.phi.conj())

    def get_phi(self):
        return self.phi

    def get_rho(self):
        return self.rho

#Вычисление Фиделити для матриц плотности
def Fidelity(rho1, rho2):
    return np.abs(np.trace(sLA.sqrtm(sLA.sqrtm(rho1) @ rho2 @ sLA.sqrtm(rho1))))**2

#Вычисление Фиделити для векторов чистых состояний
def Fidelity_pure(vec0, vec1):
    return np.abs(np.dot(vec0, vec1.conjugate())) ** 2

#Вычисление\норму Фробениуса
def Frobenius_norm(matrix):
    return np.sqrt(np.sum(list(map(lambda x: np.abs(x) ** 2,matrix))))
```

In [7]:

```
# Создадим случайное чистое состояние размерности 4
d = 4

state = State(d)
state.build_clear_state()
phi = state.get_phi()
rho = state.get_rho()

#Проверка
(np.abs(phi)**2).sum(),phi, np.trace(rho), np.trace(np.dot(rho, rho))
```

Out[7]:

```
(0.9999999999999998,
 array([-0.20583501-0.60322185j,  0.53290457+0.34194131j,
         0.04553771-0.25306455j, -0.12842804+0.33201671j]),
 (0.9999999999999998+0j),
 (0.9999999999999993+0j))
```

In [8]:
```
# Получение вероятностей по правилу Борна
def apply_P(rho,P):
    p = np.zeros((P.shape[0], P.shape[1]))
    j = 0
    for P_j in P:
        k = 0
        for P_jk in P_j:
            p[j][k] = np.trace(np.dot(P_jk, rho))
            k += 1
        j += 1

    return p

# Сравним распределения вероятностей
def compare_distr(m1,m2):
    return np.sum(np.abs(m1 - m2))
```

In [9]:
```
# Получим вероятности измерений тремя способами
p = apply_P(rho,P)
p_X = np.abs(np.dot(X,phi))**2
p_B = np.dot(B, np.ravel(rho, order = 'F'))
```

```
/home/stas/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8: ComplexWarning: Casting complex values to
real discards the imaginary part
```

In [10]:
```
# Сравним распределения вероятностей, полученные разными способами
print('Сумма модулей разностей вероятностей полученных по правилу Борна и с помощбю матрицы X: ', compare_distr(p_X.
print('Сумма модулей разностей вероятностей полученных по правилу Борна и с помощбю матрицы B: ', compare_distr(p_B.
```

```
Сумма модулей разностей вероятностей полученных по правилу Борна и с помощбю матрицы X:  6.071532165918825e-16
Сумма модулей разностей вероятностей полученных по правилу Борна и с помощбю матрицы B:  3.3998342026444346e-16
```

Ниже представлены ф-ии для реконструкции матрицы плотности состояния из лабороторной работы №2

In [11]:

```
# Моделирует серию измерений:

def estimate_probs(rho, n_shots = 100, d = 4):
    p_B = np.dot(B, np.ravel(rho, order = 'F'))
    rng = np.random.default_rng()
    p_B_matrix = p_B.reshape((5,d))
    prob_res = np.ravel(np.array([rng.multinomial(n_shots, x.astype(dtype = float)) for x in p_B_matrix])/n_shots, o
    return prob_res



# Корректирует СЗ восстановленной матрицы, проектируя её на
# множество матриц плотности
def correct_eigvals(v):

    vals = sorted(v)[::-1]
    inds = np.arange(len(vals))
    w_list = np.abs(np.cumsum(vals)-1)/(inds + 1)

    j = 0
    for val, w in zip(vals,w_list):
        if (val - w) <0:
            break
        else:
            j += 1

    vals_correct = copy.deepcopy(v)

    if  j <= (len(vals) - 1):
        vals_correct = vals_correct - w_list[j-1]
        vals_correct[vals_correct<0] = 0

    return vals_correct

def project_rho(rho):

    vals, vecs = LA.eig(rho.copy())
    vals_new = correct_eigvals(vals)
    psi = np.dot(vecs, np.diag(np.sqrt(vals_new)))
    rho_new = np.dot(psi, psi.conj().T)
    return rho_new



# реализут восстановление матрицы плотности
def recover_rho(B, probs, correct_rho = True):
    u, s, vh = LA.linalg.svd(B)
    q = np.dot(u.conj().T, probs)

    tail_num = B.shape[0] - B.shape[1]
    f = q[:-tail_num]/s
    rho_new = np.reshape(np.dot(vh.conj().T, f),(4,4)).conj()

    vals, vecs = LA.eig(rho_new)

    delta = np.abs(np.sum(vals[vals<0]))

    rho_new_correct = rho_new.copy()

    if correct_rho == True:
        vals_new = correct_eigvals(vals)
        rho_new_correct = vecs @ np.diag(vals_new) @ vecs.conj().T

    return rho_new_correct, delta
```

In [12]:

```
def purify_rho_to_psi(rho, rang = 1):


    vals, vecs = LA.eig(rho.copy())


    indices = vals.argsort()
    vals_pure = vals[sorted(indices[-rang:])]
    vecs_pure = vecs.T[sorted(indices[-rang:])].T

#     print(vals_pure)

    psi = np.dot(vecs_pure, np.diag(np.sqrt(vals_pure)))

    return psi
```

In [168…

```
d = 4
n = 100 # умножаем на 4, тк имеется 4 оператора измерения

state = State(d)
state.build_clear_state()
rho = state.get_rho()


prob_res = estimate_probs(rho, n_shots = n, d = d)
rho_new, delta = recover_rho(B, prob_res, correct_rho = True)
```

```
/home/stas/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:7: ComplexWarning: Casting complex values to
real discards the imaginary part
  import sys
```

In [169…

```
# Подадим сначала исходную матрицу плотности чистого состояния
psi_init_list = []
for r in range(1,5):
    psi = purify_rho_to_psi(rho, rang = r)
    psi_init_list.append(psi)

    print(f'Fidelity очищенной матрицы плотности ранга {r}: {Fidelity(np.dot(psi, psi.conj().T), rho)}')
```

```
Fidelity очищенной матрицы плотности ранга 1: 1.0000000077599411
Fidelity очищенной матрицы плотности ранга 2: 1.0000000121232806
Fidelity очищенной матрицы плотности ранга 3: 1.0000000247701382
Fidelity очищенной матрицы плотности ранга 4: 1.0000000232467723
```

Fidelity равна 1 для всех рангов очищения, тк на вход матрица плотности чистого состояния

In [170…

```
psi_list = []
for r in range(1,5):
    psi = purify_rho_to_psi(rho_new, rang = r)
    psi_list.append(psi)

    print(f'Fidelity очищенной матрицы плотности ранга {r}: {Fidelity(np.dot(psi, psi.conj().T), rho_new)}')
```

```
Fidelity очищенной матрицы плотности ранга 1: 0.8959456436901126
Fidelity очищенной матрицы плотности ранга 2: 1.0000000119975363
Fidelity очищенной матрицы плотности ранга 3: 1.0000000259802773
Fidelity очищенной матрицы плотности ранга 4: 1.0000000140433727
```

In [171…

```
P.shape, prob_res.shape
```

Out[171…

```
((5, 4, 4, 4), (20,))
```

In [172…

```
prob_res
```

Out[172…

```
array([0.  , 0.36, 0.34, 0.3 , 0.33, 0.46, 0.21, 0.  , 0.1 , 0.51, 0.24,
       0.15, 0.59, 0.06, 0.01, 0.34, 0.27, 0.05, 0.04, 0.64])
```

In [173…

```
(np.reshape(P, (20,4,4)).T * prob_res).T
```

Out[173…

```
array([[[ 0.    +0.j    ,  0.    +0.j    ,  0.    +0.j    ,
          0.    +0.j    ],
        [ 0.    +0.j    ,  0.    +0.j    ,  0.    +0.j    ,
          0.    +0.j    ],
        [ 0.    +0.j    ,  0.    +0.j    ,  0.    +0.j    ,
          0.    +0.j    ],
        [ 0.    +0.j    ,  0.    +0.j    ,  0.    +0.j    ,
          0.    +0.j    ]],

       [[ 0.    +0.j    ,  0.    +0.j    ,  0.    +0.j    ,
          0.    +0.j    ],
        [ 0.    +0.j    ,  0.36  +0.j    ,  0.    +0.j    ,
          0.    +0.j    ],
        [ 0.    +0.j    ,  0.    +0.j    ,  0.    +0.j    ,
          0.    +0.j    ],
        [ 0.    +0.j    ,  0.    +0.j    ,  0.    +0.j    ,
          0.    +0.j    ]],

       [[ 0.    +0.j    ,  0.    +0.j    ,  0.    +0.j    ,
          0.    +0.j    ],
        [ 0.    +0.j    ,  0.    +0.j    ,  0.    +0.j    ,
          0.    +0.j    ],
        [ 0.    +0.j    ,  0.    +0.j    ,  0.34  +0.j    ,
          0.    +0.j    ],
        [ 0.    +0.j    ,  0.    +0.j    ,  0.    +0.j    ,
          0.    +0.j    ]],

       [[ 0.    +0.j    ,  0.    +0.j    ,  0.    +0.j    ,
          0.    +0.j    ],
        [ 0.    +0.j    ,  0.    +0.j    ,  0.    +0.j    ,
          0.    +0.j    ],
        [ 0.    +0.j    ,  0.    +0.j    ,  0.    +0.j    ,
          0.    +0.j    ],
```

```
 [ 0.    +0.j    ,  0.    +0.j    ,  0.    +0.j    ,
   0.3   +0.j    ]],

[[ 0.0825+0.j    ,  0.0825+0.j    ,  0.0825+0.j    ,
   0.0825+0.j    ],
 [ 0.0825+0.j    ,  0.0825+0.j    ,  0.0825+0.j    ,
   0.0825+0.j    ],
 [ 0.0825+0.j    ,  0.0825+0.j    ,  0.0825+0.j    ,
   0.0825+0.j    ],
 [ 0.0825+0.j    ,  0.0825+0.j    ,  0.0825+0.j    ,
   0.0825+0.j    ]],

[[ 0.115 +0.j    ,  0.115 +0.j    , -0.115 -0.j    ,
  -0.115 -0.j    ],
 [ 0.115 +0.j    ,  0.115 +0.j    , -0.115 -0.j    ,
  -0.115 -0.j    ],
 [-0.115 +0.j    , -0.115 +0.j    ,  0.115 +0.j    ,
   0.115 +0.j    ],
 [-0.115 +0.j    , -0.115 +0.j    ,  0.115 +0.j    ,
   0.115 +0.j    ]],

[[ 0.0525+0.j    , -0.0525-0.j    , -0.0525-0.j    ,
   0.0525+0.j    ],
 [-0.0525+0.j    ,  0.0525+0.j    ,  0.0525+0.j    ,
  -0.0525+0.j    ],
 [-0.0525+0.j    ,  0.0525+0.j    ,  0.0525+0.j    ,
  -0.0525+0.j    ],
 [ 0.0525+0.j    , -0.0525-0.j    , -0.0525-0.j    ,
   0.0525+0.j    ]],

[[ 0.    +0.j    ,  0.    -0.j    ,  0.    +0.j    ,
   0.    -0.j    ],
 [-0.    +0.j    ,  0.    +0.j    , -0.    +0.j    ,
   0.    +0.j    ],
 [ 0.    +0.j    ,  0.    -0.j    ,  0.    +0.j    ,
   0.    -0.j    ],
 [-0.    +0.j    ,  0.    +0.j    , -0.    +0.j    ,
   0.    +0.j    ]],

[[ 0.025 +0.j    , -0.025 -0.j    ,  0.    +0.025j ,
   0.    +0.025j ],
 [-0.025 +0.j    ,  0.025 +0.j    ,  0.    -0.025j ,
   0.    -0.025j ],
 [ 0.    -0.025j , -0.    +0.025j ,  0.025 +0.j    ,
   0.025 +0.j    ],
 [ 0.    -0.025j , -0.    +0.025j ,  0.025 +0.j    ,
   0.025 +0.j    ]],

[[ 0.1275+0.j    , -0.1275-0.j    ,  0.    -0.1275j,
   0.    -0.1275j],
 [-0.1275+0.j    ,  0.1275+0.j    ,  0.    +0.1275j,
   0.    +0.1275j],
 [ 0.    +0.1275j,  0.    -0.1275j,  0.1275+0.j    ,
   0.1275+0.j    ],
 [ 0.    +0.1275j,  0.    -0.1275j,  0.1275+0.j    ,
   0.1275+0.j    ]],

[[ 0.06  +0.j    ,  0.06  +0.j    ,  0.    -0.06j  ,
   0.    +0.06j  ],
 [ 0.06  +0.j    ,  0.06  +0.j    ,  0.    -0.06j  ,
   0.    +0.06j  ],
 [ 0.    +0.06j  ,  0.    +0.06j  ,  0.06  +0.j    ,
  -0.06  +0.j    ],
 [ 0.    -0.06j  ,  0.    -0.06j  , -0.06  -0.j    ,
   0.06  +0.j    ]],

[[ 0.0375+0.j    ,  0.0375+0.j    ,  0.    +0.0375j,
   0.    -0.0375j],
 [ 0.0375+0.j    ,  0.0375+0.j    ,  0.    +0.0375j,
   0.    -0.0375j],
 [ 0.    -0.0375j,  0.    -0.0375j,  0.0375+0.j    ,
  -0.0375-0.j    ],
 [ 0.    +0.0375j,  0.    +0.0375j, -0.0375+0.j    ,
   0.0375+0.j    ]],

[[ 0.1475+0.j    ,  0.    +0.1475j,  0.    +0.1475j,
  -0.1475-0.j    ],
 [ 0.    -0.1475j,  0.1475+0.j    ,  0.1475+0.j    ,
  -0.    +0.1475j],
 [ 0.    -0.1475j,  0.1475+0.j    ,  0.1475+0.j    ,
  -0.    +0.1475j],
 [-0.1475+0.j    ,  0.    -0.1475j,  0.    -0.1475j,
   0.1475+0.j    ]],

[[ 0.015 +0.j    ,  0.    +0.015j ,  0.    -0.015j ,
   0.015 +0.j    ],
 [ 0.    -0.015j ,  0.015 +0.j    , -0.015 -0.j    ,
   0.    -0.015j ],
 [ 0.    +0.015j , -0.015 +0.j    ,  0.015 +0.j    ,
   0.    +0.015j ],
 [ 0.015 +0.j    ,  0.    +0.015j ,  0.    -0.015j ,
```

```
   0.015 +0.j    ]],

[[ 0.0025+0.j    ,  0.    -0.0025j,  0.    -0.0025j,
  -0.0025-0.j    ],
 [ 0.    +0.0025j,  0.0025+0.j    ,  0.0025+0.j    ,
   0.    -0.0025j],
 [ 0.    +0.0025j,  0.0025+0.j    ,  0.0025+0.j    ,
   0.    -0.0025j],
 [-0.0025+0.j    ,  0.    +0.0025j,  0.    +0.0025j,
   0.0025+0.j    ]],

[[ 0.085 +0.j    ,  0.    -0.085j ,  0.    +0.085j ,
   0.085 +0.j    ],
 [ 0.    +0.085j ,  0.085 +0.j    , -0.085 +0.j    ,
   0.    +0.085j ],
 [ 0.    -0.085j , -0.085 -0.j    ,  0.085 +0.j    ,
   0.    -0.085j ],
 [ 0.085 +0.j    ,  0.    -0.085j ,  0.    +0.085j ,
   0.085 +0.j    ]],

[[ 0.0675+0.j    ,  0.    +0.0675j, -0.0675-0.j    ,
   0.    +0.0675j],
 [ 0.    -0.0675j,  0.0675+0.j    , -0.    +0.0675j,
   0.0675+0.j    ],
 [-0.0675+0.j    ,  0.    -0.0675j,  0.0675+0.j    ,
   0.    -0.0675j],
 [ 0.    -0.0675j,  0.0675+0.j    , -0.    +0.0675j,
   0.0675+0.j    ]],

[[ 0.0125+0.j    ,  0.    +0.0125j,  0.0125+0.j    ,
   0.    -0.0125j],
 [ 0.    -0.0125j,  0.0125+0.j    ,  0.    -0.0125j,
  -0.0125-0.j    ],
 [ 0.0125+0.j    ,  0.    +0.0125j,  0.0125+0.j    ,
   0.    -0.0125j],
 [ 0.    +0.0125j, -0.0125+0.j    ,  0.    +0.0125j,
   0.0125+0.j    ]],

[[ 0.01  +0.j    ,  0.    -0.01j  , -0.01  -0.j    ,
   0.    -0.01j  ],
 [ 0.    +0.01j  ,  0.01  +0.j    ,  0.    -0.01j  ,
   0.01  +0.j    ],
 [-0.01  +0.j    ,  0.    +0.01j  ,  0.01  +0.j    ,
   0.    +0.01j  ],
 [ 0.    +0.01j  ,  0.01  +0.j    ,  0.    -0.01j  ,
   0.01  +0.j    ]],

[[ 0.16  +0.j    ,  0.    -0.16j  ,  0.16  +0.j    ,
   0.    +0.16j  ],
 [ 0.    +0.16j  ,  0.16  +0.j    ,  0.    +0.16j  ,
  -0.16  +0.j    ],
 [ 0.16  +0.j    ,  0.    -0.16j  ,  0.16  +0.j    ,
   0.    +0.16j  ],
 [ 0.    -0.16j  , -0.16  -0.j    ,  0.    -0.16j
```

In [194…

```python
def psi_prob(psi, P_matrix):
    prob = np.trace(psi @ psi.conj().T @ P_matrix)
    return prob

def J_operator(psi, k_list, P_list):
    prob_list = np.array([psi_prob(psi, P_matrix) for P_matrix in P_list])
    J = np.sum((P_list.T*k_list/prob_list).T, axis = 0)
    return J

def I_operator(P_list, n_shots = 100):

    n_shots = copy.copy(n_shots)/4 # тк имеется 4 измерения в каждом базисе

    I = np.sum(P_list*n_shots, axis = 0)
    return I

def update_psi(psi, k_list, P_list, mu = 0.5, n_shots = 100):
    psi_new = (1 - mu)*np.linalg.inv(I_operator(P_list, n_shots = n_shots))@J_operator(psi, k_list, P_list)@psi.copy
    return psi_new

def find_MML_psi(psi0, k_list, P_list, mu = 0.5, eps = 1e-8, n_shots = 100, verbose = False):
    psi = psi0.copy()
    for i in range(5000):
        psi_new = update_psi(psi.copy(), k_list, P_list, mu = 0.5, n_shots = n_shots)
        err = abs(Frobenius_norm(psi @ psi.T.conj() - psi_new @ psi_new.T.conj()))
        if verbose:
            print(err)
        if err < eps:
            break
        psi = psi_new
    return psi_new
```

In [195…

```
rho_MML_list = []
Fidelity_list = []
for psi in psi_list:
    psi_MML = find_MML_psi(psi, prob_res*n, np.reshape(P, (20,4,4)), mu = 0.5, eps = 1e-8,  n_shots = n, verbose = T
    rho_MML = np.dot(psi_MML, psi_MML.conj().T)
    rho_MML = project_rho(rho_MML)
    rho_MML_list.append(rho_MML)
    print()
```

```
5.309187286308206
2.087818825817047
0.20836615391848937
0.01550504144197077
0.009301653553821837
0.005653102803858354
0.0034330275938580855
0.002084151256038093
0.0012651776331754443
0.0007680841654652312
0.00046638375382856094
0.0002832601792658725
0.00017209116103534178
0.00010458803778599079
6.358770095426135e-05
3.867656973061305e-05
2.3535424393687877e-05
1.432883279709145e-05
8.728304132408141e-06
5.319796861190802e-06
3.244311252475048e-06
1.979833758556355e-06
1.209011381500014e-06
7.38829715174303e-07
4.5184293331998327e-07
2.7655184716007875e-07
1.6940570496269624e-07
1.0386229331749424e-07
6.373552285897395e-08
3.914835236780922e-08
2.406950678438718e-08
1.4813405338156633e-08
9.126147012301558e-09

5.200218578592196
2.028664153158164
0.198939035228468
0.014226185757437334
0.008759657850677196
0.0055336996122305184
0.003616079774108793
0.002482527759068118
0.0018107497500151229
0.0014034495712503842
0.001143562906669088
0.0009653820583800471
0.000833887332893377
0.0007308152771820408
0.0006464728093890135
0.0005754445104405243
0.0005144804547187407
0.0004614764027079247
0.0004149736329926744
0.0003739033578836116
0.0003374484318726709
0.0003049636087702272
0.0002759266424972602
0.0002499066311686421
0.0002265425709095705
0.00020552826955961515
0.0001866013952176475
0.00016953531078571374
0.00015413284364219413
0.00014022143560425327
0.00012764930104322564
0.00011628233718414554
0.00010600160642688736
9.670126110173697e-05
8.828681560049548e-05
8.06736948334588e-05
7.378600508249673e-05
6.75554857170952e-05
6.192060944653302e-05
5.68258056988705e-05
5.222078704140869e-05
4.805996269067804e-05
4.430192638351564e-05
4.0909008448729927e-05
3.784688390182061e-05
3.508422990868563e-05
3.259242711691926e-05
```

```
3.0345300119325582e-05
2.8318892823620668e-05
2.6491274746461666e-05
2.484237435748081e-05
2.3353835575999713e-05
2.2008893538076754e-05
2.0792265810097844e-05
1.969005541440382e-05
1.8689662378897506e-05
1.7779701037443035e-05
1.6949920849882495e-05
1.6191129264365547e-05
1.5495115674655936e-05
1.4854576188288817e-05
1.4263039294043756e-05
1.3714792926116721e-05
1.3204813496139142e-05
1.2728697618754367e-05
1.2282597158802831e-05
1.1863158144947383e-05
1.1467463968810663e-05
1.109298311466037e-05
1.0737521515474866e-05
1.0399179531263857e-05
1.0076313401063518e-05
9.767500963822193e-06
9.471511362548195e-06
9.187278441693686e-06
8.913877488106114e-06
8.65050501587119e-06
8.396461240041722e-06
8.151134958049116e-06
7.913990543430845e-06
7.684556789913555e-06
7.462417361555729e-06
7.247202648800337e-06
7.038582824546484e-06
6.8362619513532795e-06
6.639972962017187e-06
6.4494734304565705e-06
6.264541964937829e-06
6.084975187823592e-06
5.910585164330223e-06
5.741197243756273e-06
5.57664824711004e-06
5.416784921218533e-06
5.261462666126269e-06
5.11054443898801e-06
4.963899853144356e-06
4.821404409536456e-06
4.682938854453643e-06
4.548388643758794e-06
4.417643485005251e-06
4.2905969605823665e-06
4.167146203816119e-06
4.0471916272452e-06
3.930636698399747e-06
3.81738773761491e-06
3.7073537606828113e-06
3.6004463338674406e-06
3.4965794454396067e-06
3.395669413387499e-06
3.2976347801758114e-06
3.2023962407130594e-06
3.1098765654798138e-06
3.0200005366419596e-06
2.93269489017356e-06
2.8478882638474977e-06
2.7655111385423493e-06
2.6854958099666373e-06
2.607776323950271e-06
2.5322884497912798e-06
2.4589696318464304e-06
2.387758958073075e-06
2.3185971187401345e-06
2.2514263704785396e-06
2.186190500407182e-06
2.122834795458871e-06
2.0613060037161197e-06
2.00155230403269e-06
1.943523272873367e-06
1.8871698532585603e-06
1.832444315198666e-06
1.7793002358000317e-06
1.7276924631504217e-06
1.6775770811723477e-06
1.6289113877742793e-06
1.581653859691759e-06
1.535764124349072e-06
1.491202933280633e-06
1.4479321314843818e-06
```

```
1.4059146304279489e-06
1.3651143816276498e-06
1.3254963490025638e-06
1.28702648247253e-06
1.2496716908245164e-06
1.2133998226662675e-06
1.1781796308174895e-06
1.143980761213866e-06
1.1107737172417263e-06
1.0785298434812463e-06
1.0472212989914771e-06
1.0168210422472737e-06
9.873027987680317e-07
9.58641049087752e-07
9.308110027990003e-07
9.037885810882481e-07
8.775503968438602e-07
8.520737308869186e-07
8.273365240947146e-07
8.03317344436572e-07
7.799953817884686e-07
7.573504210630595e-07
7.353628368484272e-07
7.140135646882888e-07
6.932840898223398e-07
6.731564350584672e-07
6.53613139783608e-07
6.346372527128761e-07
6.162123075397086e-07
5.983223148775485e-07
5.809517533540114e-07
5.640855426523007e-07
5.477090481118329e-07
5.318080485636543e-07
5.163687472251473e-07
5.013777360094626e-07
4.868219998992409e-07
4.726889041757756e-07
4.5896617464227044e-07
4.456418992168826e-07
4.3270450483393346e-07
4.201427592480063e-07
4.079457549894788e-07
3.9610289838003084e-07
3.8460390724377833e-07
3.734387963385544e-07
3.625978666195022e-07
3.520717071723285e-07
3.418511757483492e-07
3.319273964439039e-07
3.222917515333685e-07
3.1293587371708825e-07
3.0385163796762754e-07
2.950311562527954e-07
2.864667675041635e-07
2.7815103650818264e-07
2.7007674044048326e-07
2.6223686963674216e-07
2.546246140286132e-07
2.4723336517846217e-07
2.400567047542128e-07
2.3308840113115456e-07
2.2632240446612196e-07
2.1975283850672538e-07
2.1337399896348019e-07
2.0718034873391966e-07
2.0116651026250908e-07
1.953272585075832e-07
1.8965752972274548e-07
1.841523958923627e-07
1.788070820496329e-07
1.7361694308890592e-07
1.68577476340494e-07
1.636843071361982e-07
1.5893318431333897e-07
1.5431998470515238e-07
1.4984070565572128e-07
1.4549145737376343e-07
1.4126846285778005e-07
1.3716805780090353e-07
1.3318668405460404e-07
1.2932088233981336e-07
1.2556729989366772e-07
1.219226779207354e-07
1.1838385233736662e-07
1.1494775171691532e-07
1.1161139366745381e-07
1.0837188253395208e-07
1.0522640542246494e-07
1.0217223560052058e-07
9.920671887401312e-08
```

```
9.632728160427921e-08
9.353142673302699e-08
9.081672659729694e-08
8.818082615758786e-08
8.562143518612175e-08
8.313633599167008e-08
8.072336907952288e-08
7.838044204694208e-08
7.610552093551464e-08
7.389663221665973e-08
7.175185789686583e-08
6.966933738122227e-08
6.764726397589273e-08
6.568388192734673e-08
6.377748936750669e-08
6.192642899227e-08
6.012909626500327e-08
5.838393254503822e-08
5.668942193696725e-08
5.50440951157056e-08
5.3446523155532625e-08
5.1895319832133745e-08
5.0389141841548353e-08
4.892667926199236e-08
4.750666360134843e-08
4.612786373198523e-08
4.478908355859233e-08
4.348915992037365e-08
4.222696569838387e-08
4.100140609653698e-08
3.9811417100615645e-08
3.865596672536857e-08
3.7534052040666456e-08
3.644470085566648e-08
3.5386966240220284e-08
3.435993143166336e-08
3.336270511547136e-08
3.2394421651059603e-08
3.1454241503044105e-08
3.054134957957726e-08
2.9654952921026306e-08
2.8794282938998408e-08
2.7958591137282694e-08
2.714715571688011e-08
2.6359271426497154e-08
2.5594252417499752e-08
2.4851438743647397e-08
2.4130182694822166e-08
2.342986062027143e-08
2.2749863838853704e-08
2.2089602513711182e-08
2.144850478992734e-08
2.0826013041059636e-08
2.0221588204187594e-08
1.963470659239786e-08
1.9064856819545417e-08
1.85115468411066e-08
1.7974294013887824e-08
1.745263644772804e-08
1.6946116760602687e-08
1.645429880909399e-08
1.5976753744214624e-08
1.551307014391968e-08
1.5062842152958626e-08
1.4625682990714374e-08
1.4201210243789904e-08
1.378905629144369e-08
1.3388864888675158e-08
1.3000288622759305e-08
1.2622989147321047e-08
1.2256640554089768e-08
1.1900924298277197e-08
1.15555311925306e-08
1.122016160679274e-08
1.0894528122699281e-08
1.0578342289841035e-08
1.0271336253768364e-08
9.973237379702434e-09

5.200218578592196
2.028664153158164
0.19893903522846804
0.014226185757437379
0.008759657850676995
0.005533699612230414
0.00361607977410907
0.0024825277590687237
0.0018107497500145678
0.001403449571250562
0.0011435629066691684
0.0009653820583797197
```

```
0.0008338873328932015
0.0007308152771819894
0.000646472809389116
0.000575444510440487
0.0005144804547187828
0.0004614764027079693
0.0004149736329925372
0.0003739033578835534
0.00033744843187264524
0.00030496360877050005
0.00027592664249696043
0.000249906631169046
0.00022654257090953 64
0.00020552826955948196
0.00018660139521766046
0.00016953531078590893
0.00015413284364215
0.00014022143560416 43
0.00012764930104327939
0.00011628233718425207
0.00010600160642661729
9.67012611017673e-05
8.828681560060648e-05
8.067369483354223e-05
7.37860050823563e-05
6.755548571701029e-05
6.192060944663541e-05
5.682580569906079e-05
5.222078704131504e-05
4.805996269054005e-05
4.4301926383393886e-05
4.0909008448966317e-05
3.784688390170942e-05
3.508422990873213e-05
3.259242711668197e-05
3.0345300119779338e-05
2.83188928234208e-05
2.6491274746388252e-05
2.484237435743551e-05
2.3353835575914976e-05
2.2008893538302163e-05
2.0792265810368244e-05
1.969005541375143e-05
1.86896623793239e-05
1.7779701037858098e-05
1.694992084890873e-05
1.619112926431226e-05
1.5495115675328413e-05
1.4854576188079676e-05
1.4263039294319313e-05
1.3714792925601727e-05
1.320481349589273e-05
1.2728697619055856e-05
1.2282597158868183e-05
1.1863158144574203e-05
1.1467463969315615e-05
1.1092983114603507e-05
1.0737521515498642e-05
1.0399179531353823e-05
1.0076313400903187e-05
9.767500963456719e-06
9.471511363092558e-06
9.187278441589942e-06
8.913877488224114e-06
8.650505015126402e-06
8.396461240428694e-06
8.151134958241048e-06
7.913990543714823e-06
7.684556789913896e-06
7.462417361184107e-06
7.247202648919883e-06
7.038582824686081e-06
6.836261951109042e-06
6.639972963115282e-06
6.449473429942619e-06
6.264541964139204e-06
6.084975188181733e-06
5.910585164408827e-06
5.74119724432536e-06
5.576648246394386e-06
5.4167849214855915e-06
5.2614626662189185e-06
5.110544438660729e-06
4.963899853520024e-06
4.821404409491345e-06
4.682938854334178e-06
4.548388643653084e-06
4.417643484815138e-06
4.290596961603413e-06
4.167146203080771e-06
4.047191627556768e-06
```

```
3.930636698100793e-06
3.817387737171951e-06
3.7073537608510744e-06
3.600446333735998e-06
3.4965794459355475e-06
3.395669412900808e-06
3.29763478027632e-06
3.202396241243269e-06
3.1098765656548468e-06
3.0200005370227865e-06
2.9326948902797832e-06
2.8478882621191672e-06
2.7655111392200777e-06
2.6854958102549254e-06
2.607776323863352e-06
2.5322884494027093e-06
2.4589696317923618e-06
2.387758957752785e-06
2.318597119446205e-06
2.2514263701156813e-06
2.186190500815012e-06
2.1228347948854455e-06
2.0613060034784287e-06
2.0015523045312356e-06
1.943523273047441e-06
1.8871698535140579e-06
1.8324443146085657e-06
1.7793002358592202e-06
1.7276924628181688e-06
1.6775770816235717e-06
1.628911387824733e-06
1.5816538598409714e-06
1.535764124387142e-06
1.4912029333727588e-06
1.447932131531402e-06
1.4059146303927282e-06
1.3651143815772274e-06
1.3254963495174919e-06
1.287026481661932e-06
1.2496716905964227e-06
1.2133998224710214e-06
1.1781796313158674e-06
1.1439807623988878e-06
1.1107737179306888e-06
1.0785298416728862e-06
1.0472212996732681e-06
1.0168210417049503e-06
9.873027988197674e-07
9.58641048797217e-07
9.308110027665224e-07
9.037885811729183e-07
8.775503963498288e-07
8.52073731567231e-07
8.273365246577744e-07
8.033173442255597e-07
7.799953807718271e-07
7.573504215893773e-07
7.35362837145384e-07
7.140135642911925e-07
6.932840896978454e-07
6.731564350151265e-07
6.536131401481799e-07
6.346372528629419e-07
6.162123069209539e-07
5.983223149595821e-07
5.809517526962603e-07
5.640855440606543e-07
5.477090469724777e-07
5.318080494175919e-07
5.16368747267122e-07
5.01377735796917e-07
4.868219997114932e-07
4.7268890451841315e-07
4.5896617500345704e-07
4.4564189886777765e-07
4.3270450475251935e-07
4.201427594038054e-07
4.0794575508520135e-07
3.961028985987127e-07
3.846039072907158e-07
3.7343879580499514e-07
3.6259786635079093e-07
3.520717081511113e-07
3.418511754656849e-07
3.319273958894402e-07
3.222917512066865e-07
3.129358740702061e-07
3.0385163793406084e-07
2.9503115618974e-07
2.8646676777662627e-07
2.781510361663089e-07
```

```
2.70076740451814e-07
2.622368695326096e-07
2.5462461415206427e-07
2.472333652455045e-07
2.400567046917878e-07
2.3308840156254132e-07
2.2632240452989808e-07
2.1975283823041863e-07
2.1337399883452704e-07
2.0718034837368793e-07
2.011665103587033e-07
1.9532725877345818e-07
1.8965752984124724e-07
1.8415239599314867e-07
1.788070818481491e-07
1.7361694336510796e-07
1.685774764083546e-07
1.6368430674393945e-07
1.5893318394698806e-07
1.543199853190814e-07
1.498407058402969e-07
1.4549145713058228e-07
1.412684621153264e-07
1.3716805893729077e-07
1.3318668379918406e-07
1.2932088232448902e-07
1.2556729969467284e-07
1.2192267791152764e-07
1.1838385227010311e-07
1.1494775142967791e-07
1.1161139356771179e-07
1.0837188231419789e-07
1.0522640595098097e-07
1.0217223586103988e-07
9.920671912759591e-08
9.632728139517307e-08
9.35314265838943e-08
9.081672672355329e-08
8.818082575821464e-08
8.562143558805506e-08
8.313633599314963e-08
8.072336865410534e-08
7.838044230261262e-08
7.610552097807802e-08
7.389663201867616e-08
7.175185779883926e-08
6.966933769883326e-08
6.764726429596577e-08
6.568388180265411e-08
6.377748926972174e-08
6.19264284411838e-08
6.01290967824522e-08
5.8383932476676586e-08
5.668942219779073e-08
5.504409500963728e-08
5.3446522748194144e-08
5.189532123886751e-08
5.038914142169731e-08
4.8926679014280485e-08
4.750666305112768e-08
4.6127864461962986e-08
4.478908321791555e-08
4.34891598719878e-08
4.2226965061138584e-08
4.1001406277431654e-08
3.981141754239368e-08
3.8655966274774366e-08
3.7534052880917e-08
3.6444700733031746e-08
3.538696555235938e-08
3.435993193499305e-08
3.336270376842525e-08
3.2394422321662536e-08
3.1454241762042706e-08
3.054134929197657e-08
2.965495256320994e-08
2.879428312733562e-08
2.795859249533781e-08
2.714715511104883e-08
2.635927140876993e-08
2.559425223203229e-08
2.4851439154939137e-08
2.4130182199990564e-08
2.342986061950978e-08
2.274986357867333e-08
2.2089601887973993e-08
2.1448504693236275e-08
2.0826013781897134e-08
2.022158880639386e-08
1.9634706828239535e-08
1.9064857013874962e-08
```

```
1.8511546448702633e-08
1.797429395980665e-08
1.745263690133716e-08
1.6946116242100908e-08
1.6454298836731788e-08
1.5976753546015565e-08
1.5513070382173067e-08
1.5062843143761718e-08
1.4625681403564173e-08
1.4201209931225681e-08
1.3789057050493194e-08
1.3388864984604734e-08
1.3000288164442697e-08
1.2622989463363262e-08
1.225664056445896e-08
1.1900924095828172e-08
1.155553097106008e-08
1.122016214141667e-08
1.0894528117733544e-08
1.057834294411883e-08
1.0271335508249936e-08
9.973237687596647e-09

5.200218578592196
2.028664153158164
0.19893903522846834
0.014226185757436985
0.008759657850677302
0.0055336996122301915
0.003616079774108753
0.0024825277590684552
0.0018107497500143095
0.0014034495712503085
0.0011435629066695759
0.0009653820583798657
0.0008338873328933516
0.0007308152771821391
0.0006464728093893737
0.0005754445104402395
0.0005144804547186228
0.00046147640270802155
0.00041497363299266253
0.0003739033578834796
0.0003374484318726486
0.0003049636087703905
0.00027592664249713553
0.00024990663116884136
0.00022654257090947468
0.000205528269559682
0.0001866013952176633
0.00016953531078559808
0.00015413284364217638
0.00014022143560439533
0.00012764930104319075
0.00011628233718431764
0.0001060016064266013
9.67012611018106e-05
8.828681560066157e-05
8.067369483339277e-05
7.378600508262826e-05
6.755548571671794e-05
6.192060944668462e-05
5.6825805698998294e-05
5.222078704101817e-05
4.805996269097222e-05
4.430192638309031e-05
4.090900844900479e-05
3.784688390190186e-05
3.508422990865693e-05
3.259242711701397e-05
3.0345300119253947e-05
2.831889282358558e-05
2.649127474625281e-05
2.4842374357858637e-05
2.335383557609826e-05
2.20088935378366e-05
2.079226581069475e-05
1.9690055413746728e-05
1.8689662379263808e-05
1.7779701037740852e-05
1.6949920848995983e-05
1.6191129264249337e-05
1.5495115675437412e-05
1.4854576187564131e-05
1.4263039294268604e-05
1.371479292599381e-05
1.3204813496548632e-05
1.2728697618651591e-05
1.2282597159209626e-05
1.1863158144472162e-05
1.1467463969043578e-05
```

```
1.1092983114475126e-05
1.073752151585926e-05
1.0399179531092205e-05
1.0076313400799524e-05
9.767500963950695e-06
9.471511362258591e-06
9.187278441635795e-06
8.91387748908273e-06
8.650505015548494e-06
8.396461239316772e-06
8.151134958267569e-06
7.913990543807495e-06
7.684556789953015e-06
7.462417361257722e-06
7.247202648721721e-06
7.0385828248980415e-06
6.836261951241164e-06
6.639972962918926e-06
6.449473430378557e-06
6.264541964281365e-06
6.084975187615258e-06
5.910585164370487e-06
5.741197244855621e-06
5.576648245663632e-06
5.416784921646297e-06
5.261462666147943e-06
5.110544439377271e-06
4.963899852722766e-06
4.821404409579715e-06
4.682938854455649e-06
4.548388643791112e-06
4.417643484346948e-06
4.2905969621058374e-06
4.1671462030880926e-06
4.047191627399794e-06
3.9306366980751835e-06
3.817387736790326e-06
3.707353761292413e-06
3.6004463339105203e-06
3.49657944560673e-06
3.3956694132129926e-06
3.2976347801492887e-06
3.2023962410765157e-06
3.109876565684834e-06
3.0200005358846787e-06
2.9326948913716998e-06
2.847888262742695e-06
2.76551113917289e-06
2.685495809852739e-06
2.607776323349803e-06
2.532288449921663e-06
2.4589696316954286e-06
2.387758958385522e-06
2.318597119171727e-06
2.251426370173408e-06
2.186190500011401e-06
2.1228347952107654e-06
2.0613060041691807e-06
2.0015523039098603e-06
1.943523272826105e-06
1.8871698536742365e-06
1.8324443145802108e-06
1.7793002356611964e-06
1.7276924637417631e-06
1.6775770810748161e-06
1.6289113880559175e-06
1.5816538593883742e-06
1.5357641245448125e-06
1.4912029329871469e-06
1.4479321316549944e-06
1.4059146302919389e-06
1.3651143818037786e-06
1.3254963495500997e-06
1.2870264816790762e-06
1.2496716910648139e-06
1.2133998224454628e-06
1.1781796310740037e-06
1.1439807619791432e-06
1.1107737169190515e-06
1.0785298428626253e-06
1.0472212999588358e-06
1.01682104163953e-06
9.873027990460853e-07
9.58641048560358e-07
9.30811002933433e-07
9.037885805578367e-07
8.775503968863831e-07
8.520737313697682e-07
8.273365243308744e-07
8.03317344555596e-07
7.7999538037077e-07
```

```
7.573504217251703e-07
7.353628366977378e-07
7.140135650040611e-07
6.932840897184329e-07
6.731564350266272e-07
6.536131400907869e-07
6.346372525972303e-07
6.162123068223761e-07
5.983223160246665e-07
5.809517526240713e-07
5.640855435663508e-07
5.477090466956049e-07
5.318080494686789e-07
5.163687472513935e-07
5.013777357793488e-07
4.868219999435086e-07
4.7268890446351266e-07
4.5896617483239487e-07
4.456418989852225e-07
4.327045048605484e-07
4.2014275929188285e-07
4.0794575462741424e-07
3.9610289906206495e-07
3.8460390708461756e-07
3.734387956184849e-07
3.625978668656947e-07
3.520717070708287e-07
3.418511757419808e-07
3.319273964710707e-07
3.222917510771598e-07
3.129358739830031e-07
3.0385163802804306e-07
2.9503115695649414e-07
2.864667672176098e-07
2.7815103634277405e-07
2.7007674065163105e-07
2.6223686920170694e-07
2.5462461429367987e-07
2.472333649713018e-07
2.4005670518670544e-07
2.3308840143780645e-07
2.263224042690917e-07
2.1975283867385925e-07
2.1337399854326905e-07
2.071803484958868e-07
2.0116650963048483e-07
1.953272593119434e-07
1.89657529875634e-07
1.8415239588119699e-07
1.788070814200524e-07
1.73616944380364e-07
1.6857747638150493e-07
1.6368430552777235e-07
1.5893318500543186e-07
1.5431998484006982e-07
1.498407050700982e-07
1.4549145731538767e-07
1.4126846262432824e-07
1.3716805891586155e-07
1.3318668368742126e-07
1.2932088226223806e-07
1.255673001181769e-07
1.2192267800941866e-07
1.1838385225808532e-07
1.1494775132393715e-07
1.1161139362182857e-07
1.0837188259601546e-07
1.0522640566786046e-07
1.0217223640574401e-07
9.92067179520922e-08
9.632728195757768e-08
9.35314267170815e-08
9.081672695848722e-08
8.818082554894536e-08
8.56214351882779e-08
8.313633609387698e-08
8.072336879339412e-08
7.838044168330474e-08
7.610552187322883e-08
7.389663152411876e-08
7.175185803175305e-08
6.966933799202943e-08
6.764726368113023e-08
6.5683881895248e-08
6.377748938938282e-08
6.192642870420851e-08
6.012909631309701e-08
5.838393268776611e-08
5.668942281879931e-08
5.5044094486299194e-08
5.344652260759427e-08
```

```
5.1895320474194434e-08
5.038914169094206e-08
4.8926679376807615e-08
4.750666358458151e-08
4.612786416491359e-08
4.478908289263182e-08
4.3489159971910294e-08
4.222696536949687e-08
4.100140596058925e-08
3.981141707172112e-08
3.8655966989497584e-08
3.7534052854749716e-08
3.644470070543066e-08
3.538696531932791e-08
3.43599319901563e-08
3.336270391157322e-08
3.2394422548710234e-08
3.145424158132106e-08
3.0541349163861386e-08
2.9654952811467494e-08
2.879428345842079e-08
2.7958591109932056e-08
2.7147156248925237e-08
2.635927112452553e-08
2.5594252215069952e-08
2.485143882217193e-08
2.4130182292410378e-08
2.342986127474889e-08
2.2749863285628114e-08
2.208960307759386e-08
2.1448504710291693e-08
2.0826013379685356e-08
2.0221588330039607e-08
1.9634706055513175e-08
1.906485705604505e-08
1.851154670162735e-08
1.7974293939741094e-08
1.74526359377455e-08
1.6946117240368743e-08
1.6454298870004688e-08
1.597675394334731e-08
1.55130700494449e-08
1.5062842304072993e-08
1.4625682726743612e-08
1.4201210231177265e-08
1.3789056064523994e-08
1.3388865098743456e-08
1.3000288274137358e-08
1.2622989323245568e-08
1.2256640588266934e-08
```

In [196…

```
print(f'Fidelity до применения MLE: {Fidelity(rho, rho_new)}')
print()
for r in range(4):
    print(f'Fidelity с применением MLE для МП ранка r = {r+1}: {Fidelity(rho, rho_MML_list[r])}')
    print()
```

```
Fidelity до применения MLE: 0.9813651142845828

Fidelity с применением MLE для МП ранка r = 1: 0.9907177797808446

Fidelity с применением MLE для МП ранка r = 2: 0.9912926831690085

Fidelity с применением MLE для МП ранка r = 3: 0.9912926975942743

Fidelity с применением MLE для МП ранка r = 4: 0.9912926831634363
```

In [197…

```
# rho, rho_new, rho_MML_list[3]
```

In [198…

```
def p_val(rho, k_list, P_list, n_shots, r, verbose = False):
    prob_list = np.array([np.trace(rho@P) for P in P_list])

    if verbose:
        print((prob_list*n_shots).astype('int'))
        print(k_list.astype('int'))

#     xi2, p_value = scipy.stats.chisquare(prob_list, f_exp=k_list/n_shots, ddof=nu, axis=0)

    xi2 = np.sum((k_list - prob_list*n_shots)**2/(prob_list*n_shots))
#     print(xi2.real, calc_nu(r))
    p_value = 1 - chi2.cdf(xi2.real, calc_nu(r))

    return p_value
```

In [199…

```
def calc_nu(r, j = 5, d = 4):

    def calc_nu_p(d, r):
        return (2*d - r)*r - 1

    return d*j - j - calc_nu_p(d, r)

r = 3
nu = calc_nu(r)

nu, p_val(rho_MML_list[r-1], prob_res*n, np.reshape(P, (20,4,4)), n, r, verbose = True)
```

```
[43 16  6 33 17 33 48  0 31  4 60  3  3  7 21 67 31  6 20 41]
[41 20  3 36 17 30 53  0 30  3 62  5  2  6 23 69 36  6 18 40]
/home/stas/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: ComplexWarning: Casting complex values to
real discards the imaginary part
  """
```

Out[199…

```
(1, 0.005194867749863996)
```

Пункт 4:

Сгенерируем статистические данные

In [211…

```
%%time
d = 4
n = 100 # умножаем на 4, тк имеется 4 оператора измерения
N = 1000

p_val_res = []
Fidelity_MML_res = []
Fidelity_res = []

for j in range(N):

    state = State(d)
    state.build_clear_state()
    rho = state.get_rho()


    prob_res = estimate_probs(rho, n_shots = n, d = d)
    rho_new, _ = recover_rho(B, prob_res, correct_rho = True)

    psi_list = []
    for r in range(1,5):
        psi = purify_rho_to_psi(rho_new, rang = r)
        psi_list.append(psi)

    Fidelity_MML_list = []
    Fidelity_list = []
    p_value_list = []

    for r in range(4):
        psi = psi_list[r]
        psi_MML = find_MML_psi(psi, prob_res*n, np.reshape(P, (20,4,4)), mu = 0.5, eps = 1e-8,  n_shots = n)
        rho_MML = np.dot(psi_MML, psi_MML.conj().T)
        rho_MML = project_rho(rho_MML)

        Fidelity_MML_list.append(Fidelity(rho, rho_MML))
        Fidelity_list.append(Fidelity(rho, rho_new))

        p_value = p_val(rho_MML, prob_res*n, np.reshape(P, (20,4,4)), n, r+1)
        p_value_list.append(p_value)

    p_val_res.append(p_value_list)
    Fidelity_res.append(Fidelity_list)
    Fidelity_MML_res.append(Fidelity_MML_list)

p_val_res = np.array(p_val_res).T
Fidelity_MML_res = np.array(Fidelity_MML_res).T
Fidelity_res = np.array(Fidelity_res).T
```

```
/home/stas/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:7: ComplexWarning: Casting complex values to
real discards the imaginary part
  import sys
CPU times: user 33min 8s, sys: 25min 51s, total: 59min
Wall time: 16min 42s
```

In [212…

```
np.isnan(p_val_res[3][0])
```

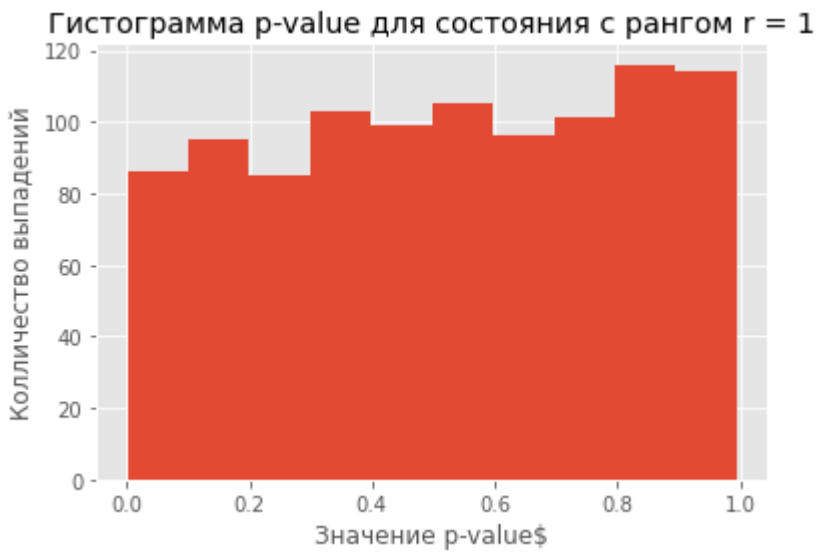
Out[212…

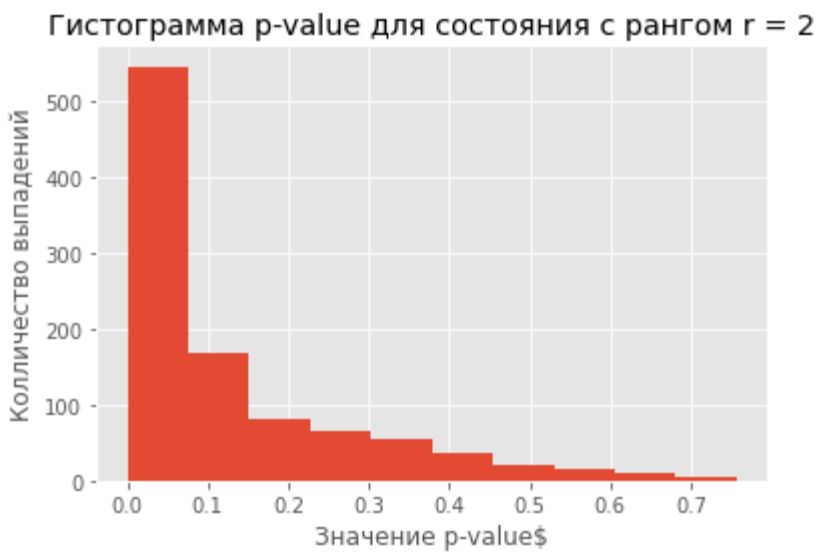
```
True
```

Построим распределение p-value для различных рангов r

In [213…

```
for r in range(4):
    plt.hist([x for x in p_val_res[r] if not np.isnan(x)], bins = 10)
    plt.title(f'Гистограмма p-value для состояния с рангом r = {r+1}')
    plt.xlabel(r'Значение p-value$')
    plt.ylabel(r'Колличество выпадений')
    plt.show()
```



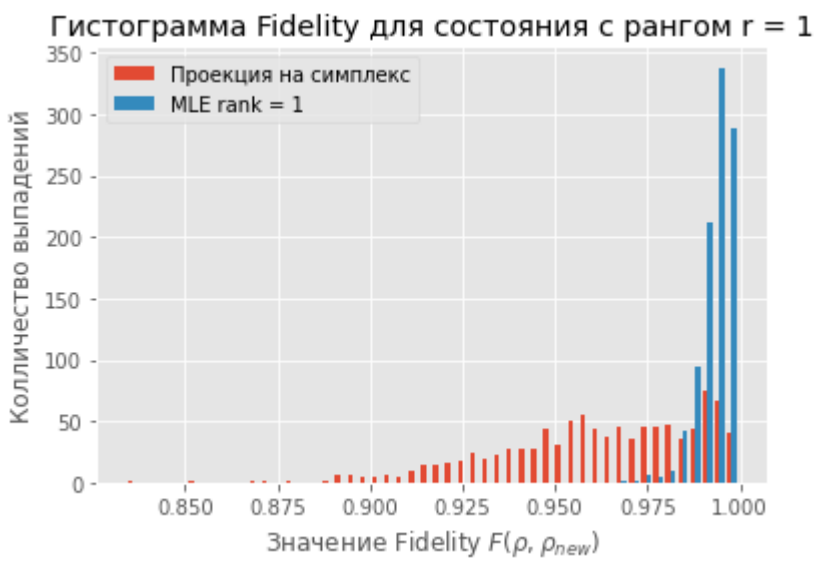


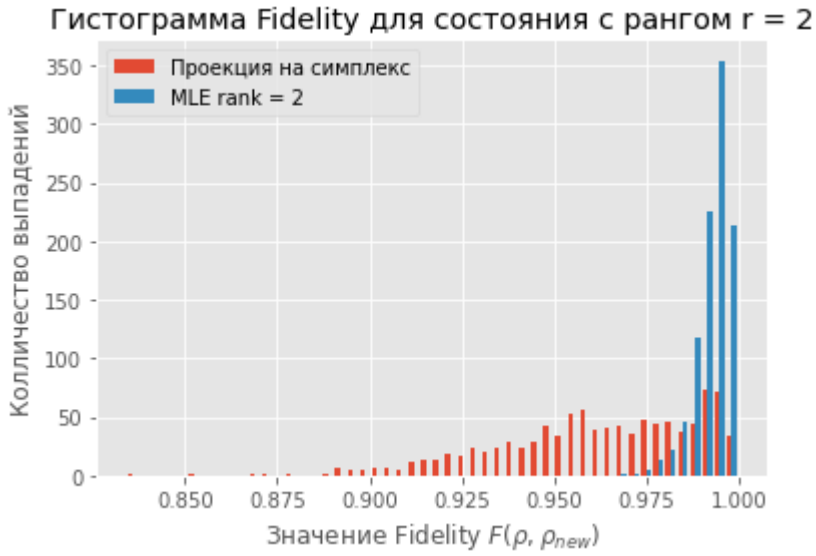
Адекватные результаты (равномерное p-value) наблюдается только для ранга r=1. Возможно такие результаты связаны с тем, что восстанавливаемое состояние чистое. Таким образом точность при оптимизации с начльным приближением большего ранга не даёт дополнительного прироста точности. В то же время количество степеней свободы возрастает с увеличением ранга, что влияет на теоретическое распределение χ^2 (увеличивается теоретическая ожидаемая точность)

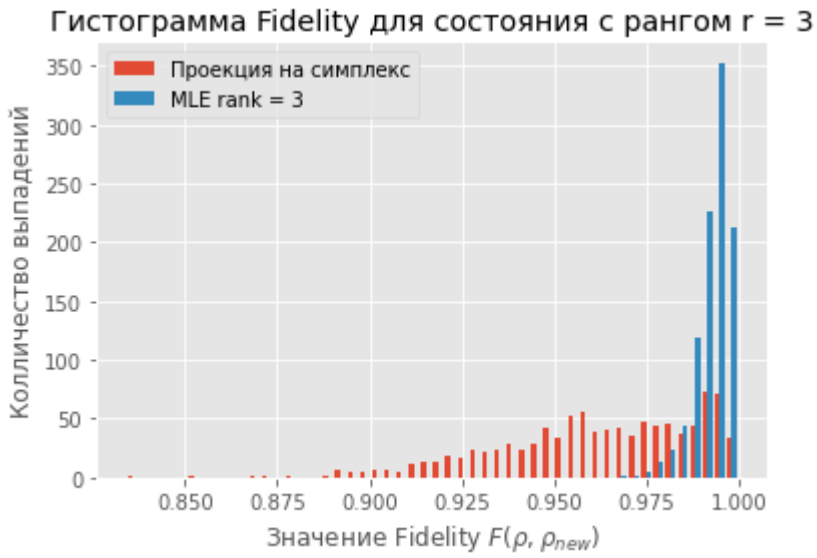
Сравним распределения фиделити для восстановленных состояний с применением MLE для каждого ранга с распределением фиделити до применения MLE.

In [214…

```
for r in range(4):
    plt.hist([Fidelity_res[r] ,Fidelity_MML_res[r] ],bins = 50,label = ['Проекция на симплекс',f'MLE rank = {r+1}'])
    plt.title(f'Гистограмма Fidelity для состояния с рангом r = {r+1}')
    plt.xlabel(r'Значение Fidelity $F(\rho, \rho_{new})$')
    plt.ylabel(r'Колличество выпадений')
    plt.legend(loc='upper left')
    plt.show()
```







Видно, что MLE заметно улучшает качество восстановленного состояния

Пункт 5:

Сравним ожидаемые средние потери точности и их дисперсии с выборочными значениями

перейдём в Евклидово пространство удвоенной размерности

In [215…
```
def double_psi(psi):
    return np.vstack([psi.real,psi.imag])

def double_P(P):
     return np.block([[P.real, -P.imag],[P.imag, P.real]])
```

In [216…
```
def build_H(psi, P_list, prob_list, n_shots = 100):

    n_shots = copy.copy(n_shots)/4 # тк имеется 4 измерения в каждом базисе

    psi = double_psi(psi)
    P_list = [double_P(P) for P in P_list]

    H = np.array([P@psi@(P@psi).T for P in P_list])
    H = np.sum((H.T/prob_list*n_shots).T, axis = 0)

    return H
```

In [217…
```
def calc_M_D(H, r = 1):
    vals = LA.eigvals(H)
    vals = np.array(sorted(vals)[1:-1])
    d = 1/(2*vals)

    mean_err = np.sum(d)
    std_err = 2*np.sum(d**2)
    return mean_err, std_err
```

Получим статистическую оценку M(1 - F) и D(1-F)

In [218…
```
F_list = Fidelity_res[0]

err_list = 1- F_list

mean_err_stat = np.mean(err_list)
std_err_stat = np.std(err_list)**2
```

Получим статистическую оценку M(1 - F) и D(1-F)

In [219…
```
prob_res
```

Out[219…
```
array([0.4 , 0.19, 0.17, 0.24, 0.47, 0.49, 0.  , 0.04, 0.27, 0.14, 0.21,
       0.38, 0.69, 0.02, 0.01, 0.28, 0.4 , 0.01, 0.02, 0.57])
```

In [222…
```
# d = 4

state = State(d)
state.build_clear_state()
rho = state.get_rho()

prob_res = [0]
while min(prob_res) <= 0:
    prob_res = estimate_probs(rho, n_shots = 100, d = d)

H = build_H(psi, np.reshape(P, (20,4,4)), prob_res)
mean_err_theor, std_err_theor = calc_M_D(H)
```

```
/home/stas/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:7: ComplexWarning: Casting complex values to
real discards the imaginary part
  import sys
```

Сравним результаты

In [223…
```
print(f'Теоретическое среднее ошибки: {mean_err_theor}')
print(f'Статистическая оценка среднего ошибки: {mean_err_stat}')
print()
print(f'Теоретическая дисперсия ошибки: {std_err_theor}')
print(f'Статистическая оценка дисперсии ошибки: {std_err_stat}')
```

```
Теоретическое среднее ошибки: 0.039085030846526835
Статистическая оценка среднего ошибки: 0.03838641137955782

Теоретическая дисперсия ошибки: 0.0006828578931005285
Статистическая оценка дисперсии ошибки: 0.0007128801814030991
```

Теоритические величины совпадают со статистическими значениями с высокой точностью

In [ ]: