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
# from functools import lru_cache
```

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

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]:

```
(0.9999999999999997,
 array([-0.44086378+0.16794189j,  0.65688276-0.15644722j,
        -0.21792252-0.23601523j, -0.46592776+0.03437833j]),
 (0.9999999999999998+0j),
 (0.9999999999999996+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:  1.2021633688519273e-15
Сумма модулей разностей вероятностей полученных по правилу Борна и с помощбю матрицы B:  3.4086001816658927e-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


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

    return psi
```

In [13]:
```
d = 4
n = 100

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


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

```
/home/stas/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:6: ComplexWarning: Casting complex values to
real discards the imaginary part
```

In [14]:
```
prob_res[:4]
```

Out[14]:
```
array([0.33, 0.55, 0.1 , 0.02])
```

In [15]:
```
# 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: 1.0000000079543585
Fidelity очищенной матрицы плотности ранга 2: 1.0000000122479662
Fidelity очищенной матрицы плотности ранга 3: 1.0000000180072446
Fidelity очищенной матрицы плотности ранга 4: 1.000000009422724
```

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

In [16]:
```
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: 0.9491798961697282
Fidelity очищенной матрицы плотности ранга 2: 1.0000000164783027
Fidelity очищенной матрицы плотности ранга 3: 1.0000000165603065
Fidelity очищенной матрицы плотности ранга 4: 1.0000000156129283
```

In [17]:
```
P.shape, prob_res.shape
```

Out[17]:
```
((5, 4, 4, 4), (20,))
```

In [18]:
```
prob_res
```

Out[18]:
```
array([0.33, 0.55, 0.1 , 0.02, 0.05, 0.33, 0.09, 0.53, 0.13, 0.4 , 0.34,
       0.13, 0.39, 0.46, 0.06, 0.09, 0.64, 0.34, 0.  , 0.02])
```

In [19]:
```
(np.reshape(P, (20,4,4)).T).T.sum(axis =0)
```

Out[19]:
```
array([[5.+0.j, 0.+0.j, 0.+0.j, 0.+0.j],
       [0.+0.j, 5.+0.j, 0.+0.j, 0.+0.j],
       [0.+0.j, 0.+0.j, 5.+0.j, 0.+0.j],
       [0.+0.j, 0.+0.j, 0.+0.j, 5.+0.j]])
```

In [20]:
```
(np.reshape(P, (20,4,4)).T * prob_res).T.sum(axis = 0)
```

Out[20]:
```
array([[ 1.33 +0.j   , -0.075+0.415j, -0.03 -0.13j , -0.155+0.065j],
       [-0.075-0.415j,  1.55 +0.j   , -0.205+0.095j,  0.11 +0.11j ],
       [-0.03 +0.13j , -0.205-0.095j,  1.1  +0.j   , -0.045-0.065j],
       [-0.155-0.065j,  0.11 -0.11j , -0.045+0.065j,  1.02 +0.j   ]])
```

In [21]:

```
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

I_ = None
def I_inv(P_list, n_shots = 100):
    global I_

    if I_ is None:
        I = np.sum(P_list*n_shots, axis = 0)
        I_ = np.linalg.inv(I)
        return I_

    else:
        return I_



def update_psi(psi, k_list, P_list, mu = 0.5, n_shots = 100):
    psi_new = (1 - mu)*I_inv(P_list, n_shots = n_shots)@J_operator(psi, k_list, P_list)@psi.copy() + mu*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 = mu, 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 [22]:

```
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()
```

```
0.05293661858932734
0.01767387697480867
0.0077037696070332735
0.0037005780355113646
0.0019398121057031096
0.0010811509191944733
0.0006251005050536292
0.00036950479836694356
0.00022186868567103266
0.00013503619548584528
8.327627701161263e-05
5.203885331709722e-05
3.294500716238022e-05
2.111676856005912e-05
1.3688815779715029e-05
8.961774012444412e-06
5.91621386818947e-06
3.932420995580151e-06
2.6281300966854726e-06
1.763977559427168e-06
1.1878931717596513e-06
8.019795680748543e-07
5.424850314199254e-07
3.674923653008727e-07
2.492245711810868e-07
1.6916044345167437e-07
1.1489057504336188e-07
7.806953741464205e-08
5.306898412942521e-08
3.6084816924022805e-08
2.4541694843114094e-08
1.6693964635492546e-08
1.1357245407477996e-08
7.727386275211004e-09

0.03478643165053966
0.0171766071341766
0.008908287720896883
0.004895804980415512
0.0028899833523878733
```

```
0.0018726067459913526
0.0013614491667520289
0.0011048048811456018
0.0009657863065685285
0.0008765762041452852
0.000807989776328002
0.0007488346437099678
0.0006951943813811751
0.0006458065695355748
0.0006002656868746576
0.0005583797353738592
0.0005199677869702864
0.0004848153181855085
0.0004526794172144655
0.00042330487217938974
0.0003964385676529105
0.000371839304586019
0.0003492833693782335
0.0003285669713493932
0.00030950663584046315
0.0002919383874912479
0.00027571629791181345
0.00026071076530059947
0.00024680674677766376
0.0002339020665765039
0.00022190586151756876
0.00021073718780841383
0.0002003237918059808
0.0001906010359209642
0.00018151096546217298
0.00017300150036725923
0.00016502573588907064
0.00015754133739332986
0.00015051001599091738
0.00014389707336916841
0.00013767100579242854
0.00013180315870691344
0.00012626742466657402
0.00012103997842019852
0.00011609904394226559
0.00011142468900077758
0.00010699864353141
0.00010280413864179858
9.882576356553258e-05
9.504933826431676e-05
9.146179971485022e-05
8.805110020665539e-05
8.480611619149359e-05
8.171656645242036e-05
7.877293849247876e-05
7.596642222366933e-05
7.328885012806199e-05
7.073264317844797e-05
6.829076190265086e-05
6.595666203173461e-05
6.372425425811409e-05
6.158786768064629e-05
5.954221654245378e-05
5.7582369958555475e-05
5.5703724305861584e-05
5.3901978036001265e-05
5.217310866341042e-05
5.0513351727819974e-05
4.8919181533908664e-05
4.738729351350376e-05
4.591458805127745e-05
4.449815563985736e-05
4.313526324783445e-05
4.182334178898895e-05
4.055997459056107e-05
3.9342886781025364e-05
3.8169935508402556e-05
3.7039100922022064e-05
3.594847784799842e-05
3.489626810812257e-05
3.388077341120316e-05
3.2900388789554364e-05
3.1953596517735256e-05
3.1038960485102e-05
3.0155120976756904e-05
2.930078983843192e-05
2.8474745988040087e-05
2.7675831244641623e-05
2.6902946460692017e-05
2.6155047918093467e-05
2.5431143981923478e-05
2.4730291980492002e-05
2.405159530477205e-05
2.339420069980258e-05
2.275729574193355e-05
2.214010648472653e-05
```

```
2.1541895260867328e-05
2.0961958628585504e-05
2.0399625450608946e-05
1.985425510263255e-05
1.9325235791030048e-05
1.881198298112818e-05
1.831393792432145e-05
1.7830566277201414e-05
1.7361356808444328e-05
1.690582018025807e-05
1.646348781283534e-05
1.603391081037864e-05
1.5616658954516994e-05
1.5211319759273097e-05
1.4817497580351898e-05
1.4434812777936615e-05
1.4062900925779178e-05
1.3701412070755908e-05
1.3350010030915166e-05
1.3008371734664369e-05
1.2676186598011447e-05
1.2353155936589987e-05
1.2038992409666327e-05
1.1733419495428557e-05
1.1436170994569907e-05
1.114699056215247e-05
1.0865631261798356e-05
1.0591855148459915e-05
1.0325432866674812e-05
1.0066143278035147e-05
9.81377309983599e-06
9.568116569801286e-06
9.328975122330492e-06
9.096157082997332e-06
8.869477384920727e-06
8.648757282929282e-06
8.43382410392053e-06
8.224510990021199e-06
8.02065666629644e-06
7.822105215601874e-06
7.628705862555159e-06
7.4403127728982995e-06
7.256784857144711e-06
7.0779855864339975e-06
6.903782816740195e-06
6.7340486191497195e-06
6.5686591228600544e-06
6.407494357216071e-06
6.250438110423063e-06
6.097377786688219e-06
5.948204273989146e-06
5.802811817644152e-06
5.661097895556254e-06
5.522963105847955e-06
5.388311050953594e-06
5.257048234067699e-06
5.1290839556628525e-06
5.004330214691121e-06
4.882701617435527e-06
4.764115284200758e-06
4.648490766250374e-06
4.535749960864393e-06
4.425817033243747e-06
4.318618339460171e-06
4.2140823553796e-06
4.112139603029737e-06
4.012722586504277e-06
3.915765725615461e-06
3.821205293681027e-06
3.7289793579844198e-06
3.639027722461274e-06
3.5512918703622304e-06
3.4657149151506212e-06
3.382241543949198e-06
3.30081797171025e-06
3.221391893515295e-06
3.1439124363794164e-06
3.0683301174614587e-06
2.9945968009228453e-06
2.922665655547734e-06
2.8524911161807765e-06
2.784028847516601e-06
2.7172357018785974e-06
2.652069691390942e-06
2.588489943871351e-06
2.5264566806800885e-06
2.4659311753657387e-06
2.4068757281555913e-06
2.34925363412142e-06
2.293029154278408e-06
2.238167489062288e-06
```

```
2.1846347495836925e-06
2.132397931776926e-06
2.081424893227991e-06
2.03168432536484e-06
1.9831457337851664e-06
1.9357794109837714e-06
1.889556417792502e-06
1.844448561361823e-06
1.8004283719537206e-06
1.7574690852056984e-06
1.7155446233167483e-06
1.6746295730394582e-06
1.6346991705452063e-06
1.5957292819380384e-06
1.5576963862154527e-06
1.5205775600594325e-06
1.4843504591571088e-06
1.4489933050453417e-06
1.4144848684810968e-06
1.380804455029527e-06
1.347931890501741e-06
1.3158475077717182e-06
1.2845321334321009e-06
1.2539670721109723e-06
1.2241340987875461e-06
1.1950154407858017e-06
1.1665937702773125e-06
1.1388521899389893e-06
1.1117742238873598e-06
1.0853438038335768e-06
1.059545260993253e-06
1.034363314823422e-06
1.009783062717074e-06
9.857899700117168e-07
9.623698616124933e-07
9.395089127761495e-07
9.171936359474444e-07
8.954108798949675e-07
8.741478143423495e-07
8.533919236783709e-07
8.331310009849506e-07
8.133531374469835e-07
7.940467167461531e-07
7.752004061722376e-07
7.568031499382792e-07
7.388441635346037e-07
7.213129238042009e-07
7.041991663374073e-07
6.874928752205964e-07
6.711842790581039e-07
6.552638444055659e-07
6.397222692952414e-07
6.24550477749e-07
6.097396145431364e-07
5.952810387514124e-07
5.811663196861332e-07
5.673872303594617e-07
5.539357431175274e-07
5.408040251972519e-07
5.279844320252188e-07
5.154695052940139e-07
5.032519662708984e-07
4.913247111646104e-07
4.796808095233896e-07
4.683134959753224e-07
4.5721616966644265e-07
4.463823887891779e-07
4.3580586561822634e-07
4.254804647707485e-07
4.154001983609667e-07
4.0555922260570693e-07
3.9595183324872797e-07
3.86572464769048e-07
3.7741568387810006e-07
3.6847618812073276e-07
3.5974880385708794e-07
3.5122847866432417e-07
3.429102839816721e-07
3.3478940819859e-07
3.2686115478906276e-07
3.191209403579027e-07
3.115642911883049e-07
3.0418684061801864e-07
2.9698432646412993e-07
2.8995258824416126e-07
2.8308756596688047e-07
2.763852951971808e-07
2.6984190804057497e-07
2.634536275077805e-07
2.5721676782740925e-07
2.5112773104203314e-07
```

```
2.451830047569013e-07
2.3937916113101e-07
2.3371285341761304e-07
2.2818081496787097e-07
2.227798581120532e-07
2.1750686905622008e-07
2.1235881006441807e-07
2.073327153646448e-07
2.0242568957421425e-07
1.9763490647062957e-07
1.92957607490145e-07
1.8839109921122627e-07
1.8393275320738666e-07
1.7958000250953916e-07
1.7533034204024832e-07
1.7118132655977426e-07
1.6713056835678334e-07
1.6317573672619745e-07
1.593145569618626e-07
1.555448080254093e-07
1.5186432130644756e-07
1.4827098059616664e-07
1.447627195344654e-07
1.4133752104897905e-07
1.3799341619760184e-07
1.34728481805796e-07
1.3154084193385693e-07
1.2842866462517209e-07
1.253901606904877e-07
1.224235843223944e-07
1.1952723133649133e-07
1.1669943704065009e-07
1.1393857711789706e-07
1.1124306584190644e-07
1.0861135441936207e-07
1.0604193182496228e-07
1.0353332169833142e-07
1.010840837280696e-07
9.869281180350874e-08
9.635813209055117e-08
9.407870488413666e-08
9.185322112146414e-08
8.968040267335635e-08
8.755900314152949e-08
8.548780431561917e-08
8.346561688705683e-08
8.149128131134369e-08
7.95636634531549e-08
7.768165774456474e-08
7.584418351456762e-08
7.405018692191122e-08
7.22986381664999e-08
7.058853238382661e-08
6.89188885521549e-08
6.728874851973308e-08
6.569717701988271e-08
6.414326124783882e-08
6.262610992790951e-08
6.114485196875706e-08
5.969863894789338e-08
5.828663995439802e-08
5.6908046598158035e-08
5.5562066742192575e-08
5.424792933627295e-08
5.296488093955722e-08
5.171218501557501e-08
5.048912317578653e-08
4.929499490776549e-08
4.8129114956049e-08
4.699081447270616e-08
4.5879441426500433e-08
4.479435805734093e-08
4.373494273390413e-08
4.270058752816093e-08
4.169069962323995e-08
4.070470040047958e-08
3.9742024145313366e-08
3.880211896722633e-08
3.7884446317919656e-08
3.698848011244508e-08
3.6113706760366165e-08
3.525962432932786e-08
3.4425744254798354e-08
3.361158747743088e-08
3.281668829261508e-08
3.2040590145152e-08
3.128284886363421e-08
3.0543030236397665e-08
2.9820709720712835e-08
2.911547379884034e-08
2.8426917934607452e-08
```

```
2.7754647826741826e-08
2.709827803853559e-08
2.6457432188546244e-08
2.5831743522357092e-08
2.522085329624698e-08
2.462441096034384e-08
2.4042075683443494e-08
2.3473512878130666e-08
2.2918396951718333e-08
2.237641050563047e-08
2.184724146155542e-08
2.1330588609700733e-08
2.0826154005808264e-08
2.0333649898252096e-08
1.9852793462616103e-08
1.938330931903222e-08
1.892492861962495e-08
1.847738887522986e-08
1.804043288611147e-08
1.7613811245292735e-08
1.71972787339166e-08
1.679059746635192e-08
1.6393533449974823e-08
1.6005860445350303e-08
1.562735549473052e-08
1.5257801438733142e-08
1.4896987714030034e-08
1.454470686705633e-08
1.4200756890062847e-08
1.3864941237175487e-08
1.3537066980740576e-08
1.3216946972085612e-08
1.2904397317003646e-08
1.2599238940119129e-08
1.2301297099892378e-08
1.2010401936731083e-08
1.1726385225224283e-08
1.1449085486366908e-08
1.1178343456863644e-08
1.0914004139036172e-08
1.0655915687896535e-08
1.0403931202655916e-08
1.015790490246022e-08
9.917697860754331e-09

0.03478643165053965
0.017176607134176607
0.008908287720896883
0.004895804980415425
0.0028899833523878134
0.0018726067459912588
0.00136144916675219
0.0011048048811456532
0.0009657863065685456
0.0008765762041453133
0.0008079897763279567
0.0007488346437099137
0.0006951943813813406
0.0006458065695354663
0.0006002656868746894
0.0005583797353738409
0.0005199677869702587
0.0004848153181854864
6
0.0004526794172145047
0.0004233048721792382
0.00039643856765297203
0.00037183930458586825
0.0003492833693784712
0.0003285669713494044
0.00030950663584062595
0.0002919383874912889
0.00027571629791155476
0.0002607107653005862
0.00024680674677768 5
0.00023390206657655004
0.00022190586151758034
0.0002107371871878383 2
0.0002003237918059784
0.00019060103592101178
0.0001815109654618753
0.0001730015003674648
0.00016502573588905657
0.00015754133739340345
0.00015051001599090586
0.00014389707336923417
0.0001376710057923264
0.00013180315870691786
0.0001262674246665773
0.0001210399784202142
0.00011609904394221289
0.00011142468900101048
```

```
0.00010699864353129782
0.0001028041386414985
9.882576356571997e-05
9.504933826442082e-05
9.146179971484218e-05
8.805110020667003e-05
8.480611619164375e-05
8.171656645220323e-05
7.877293849250785e-05
7.596642222372881e-05
7.32888501279225e-05
7.073264317846958e-05
6.829076190275828e-05
6.595666203144207e-05
6.372425425866221e-05
6.158786768029006e-05
5.9542216542465674e-05
5.7582369958707866e-05
5.570372430568423e-05
5.390197803584028e-05
5.217310866359802e-05
5.051335172783339e-05
4.8919181533906726e-05
4.7387293513525346e-05
4.591458805127311e-05
4.4498155639641366e-05
4.31352632480676e-05
4.182334178902399e-05
4.0559974590540956e-05
3.934288678091387e-05
3.816993550859809e-05
3.703910092181128e-05
3.594847784807584e-05
3.489626810797917e-05
3.388077341132546e-05
3.290038878959767e-05
3.195359651771003e-05
3.1038960485113356e-05
3.0155120976879385e-05
2.9300789838498273e-05
2.8474745987948337e-05
2.7675831244553528e-05
2.69029464606964e-05
2.615504791804439e-05
2.5431143981949885e-05
2.473029198035822e-05
2.4051595304822196e-05
2.3394200700006924e-05
2.2757295741829256e-05
2.214010648474008e-05
2.1541895260906156e-05
2.096195862845791e-05
2.039962545072851e-05
1.985425510261835e-05
1.93252357910417e-05
1.8811982980986715e-05
1.8313937924362104e-05
1.7830566277348415e-05
1.7361356808268426e-05
1.6905820180266492e-05
1.6463487812945536e-05
1.6033910810506183e-05
1.5616658954217266e-05
1.5211319759392093e-05
1.4817497580335947e-05
1.4434812777916576e-05
1.406290092574536e-05
1.370141207070821e-05
1.335001003131859e-05
1.3008371734347207e-05
1.2676186598016832e-05
1.2353155936614048e-05
1.2038992409755528e-05
1.1733419495298486e-05
1.1436170994631883e-05
1.1146990561826782e-05
1.0865631262250287e-05
1.0591855148132642e-05
1.0325432866846695e-05
1.0066143278065732e-05
9.813773099884575e-06
9.568116569773173e-06
9.328975122132102e-06
9.096157083428217e-06
8.869477384642543e-06
8.64875728294385e-06
8.43382410390004e-06
8.22451098980006e-06
8.020656666635706e-06
7.822105215267516e-06
7.628705862751623e-06
```

```
7.4403127728998386e-06
7.256784857157869e-06
7.077985586503039e-06
6.903782816590587e-06
6.734048619377334e-06
6.568659122638702e-06
6.407494357234531e-06
6.250438110558761e-06
6.0973777866304195e-06
5.948204274011455e-06
5.802811817637872e-06
5.66109789551683e-06
5.522963105930792e-06
5.388311050903446e-06
5.257048234108537e-06
5.129083955689023e-06
5.0043302145158975e-06
4.882701617494453e-06
4.764115284247592e-06
4.648490766458214e-06
4.535749960626685e-06
4.425817033367227e-06
4.3186183395516965e-06
4.214082355177687e-06
4.112139603046097e-06
4.012722586522102e-06
3.915765725631377e-06
3.821205293676998e-06
3.728979357979007e-06
3.6390277223623524e-06
3.5512918706513316e-06
3.4657149149234584e-06
3.382241543946381e-06
3.300817971584364e-06
3.2213918935501207e-06
3.1439124363484946e-06
3.068330117501682e-06
2.9945968009341735e-06
2.922665655499593e-06
2.8524911162757395e-06
2.784028847544352e-06
2.7172357021469646e-06
2.652069691153089e-06
2.588489943884342e-06
2.5264566806721853e-06
2.4659311754158797e-06
2.4068757280846133e-06
2.3492536341572528e-06
2.2930291542588736e-06
2.2381674890118895e-06
2.184634749543028e-06
2.132397932049172e-06
2.08142489302297e-06
2.0316843253878845e-06
1.9831457338983266e-06
1.9357794106234554e-06
1.8895564180056272e-06
1.8444485614337746e-06
1.800428371798085e-06
1.75746908535165e-06
1.7155446232623205e-06
1.6746295728265002e-06
1.634699170742826e-06
1.5957292819955795e-06
1.5576963862109639e-06
1.520577560078175e-06
1.4843504589574572e-06
1.4489933053774535e-06
1.414484868313055e-06
1.3808044551119165e-06
1.3479318904717202e-06
1.3158475078549328e-06
1.2845321332504608e-06
1.2539670721492796e-06
1.224134098733553e-06
1.1950154406927928e-06
1.1665937702984716e-06
1.1388521902097213e-06
1.111774223647153e-06
1.085343803870968e-06
1.0595452611022739e-06
1.0343633146607381e-06
1.0097830627812875e-06
9.85789969849341e-07
9.623698618854202e-07
9.39508912727405e-07
9.171936358632373e-07
8.954108800740239e-07
8.741478142046385e-07
8.533919235645373e-07
8.331310010646924e-07
```

```
8.133531375067297e-07
7.940467167473867e-07
7.752004061826224e-07
7.568031497792576e-07
7.388441636149168e-07
7.213129239006855e-07
7.041991664388643e-07
6.874928751080569e-07
6.711842790583587e-07
6.552638443207615e-07
6.397222693365223e-07
6.245504778208417e-07
6.097396144027523e-07
5.952810388648841e-07
5.811663196482541e-07
5.673872303653624e-07
5.539357430966925e-07
5.408040251151133e-07
5.279844321911253e-07
5.154695051319725e-07
5.032519663724142e-07
4.913247110785464e-07
4.796808095000435e-07
4.683134960623101e-07
4.572161697408046e-07
4.463823887458267e-07
4.358058655657035e-07
4.254804647275596e-07
4.154001985915499e-07
4.0555922248473374e-07
3.959518331977583e-07
3.8657246454229214e-07
3.774156840298862e-07
3.684761882512136e-07
3.597488036641904e-07
3.5122847890050305e-07
3.429102839323257e-07
3.3478940813172963e-07
3.268611548015285e-07
3.191209403719595e-07
3.1156429125278626e-07
3.041868407038501e-07
2.969843262919291e-07
2.8995258829920697e-07
2.830875660549447e-07
2.763852950721303e-07
2.6984190805679565e-07
2.6345362745199237e-07
2.572167678728186e-07
2.511277310040009e-07
2.4518300477258253e-07
2.3937916121825153e-07
2.3371285335538662e-07
2.281808151613806e-07
2.2277985808397697e-07
2.1750686901147498e-07
2.1235880994365018e-07
2.0733271538591956e-07
2.0242568959789324e-07
1.97634906448368e-07
1.92957607561081e-07
1.8839109921133355e-07
1.8393275323007986e-07
1.7958000240514973e-07
1.7533034214437647e-07
1.7118132646419523e-07
1.67130568381983e-07
1.6317573672561022e-07
1.5931455707280055e-07
1.5554480784013182e-07
1.518643214466531e-07
1.4827098051143283e-07
1.447627195158667e-07
1.4133752123581651e-07
1.3799341598702982e-07
1.3472848175582268e-07
1.3154084223117592e-07
1.2842866447807875e-07
1.253901606619413e-07
1.2242358442083346e-07
1.1952723116763372e-07
1.166994371271124e-07
1.1393857731837103e-07
1.1124306560466603e-07
1.0861135442552607e-07
1.0604193189384767e-07
1.0353332171309604e-07
1.0108408366808532e-07
9.869281177055312e-08
9.635813209810158e-08
9.407870489083153e-08
```

```
9.185322106598022e-08
8.968040268922033e-08
8.755900310960849e-08
8.548780434367213e-08
8.346561691134068e-08
8.149128138884442e-08
7.956366349532743e-08
7.768165765036712e-08
7.584418352630056e-08
7.405018677651362e-08
7.229863830293427e-08
7.058853236218383e-08
6.891888852832459e-08
6.728874843818531e-08
6.569717712418709e-08
6.414326133866858e-08
6.262610979437303e-08
6.114485202278148e-08
5.969863883268627e-08
5.828664006221931e-08
5.690804656787026e-08
5.55620667647134e-08
5.424792952536083e-08
5.296488086638942e-08
5.17121848533713e-08
5.048912333652166e-08
4.929499489992143e-08
4.81291148248489e-08
4.69908145162924e-08
4.587944122365756e-08
4.479435839803106e-08
4.373494260226308e-08
4.270058752446036e-08
4.169069974534595e-08
4.0704700203528474e-08
3.9742024002046386e-08
3.88021192344365e-08
3.7884446453854805e-08
3.698847985352475e-08
3.6113706714230484e-08
3.5259624673845036e-08
3.442574403414642e-08
3.361158761114079e-08
3.2816688045816345e-08
3.2040590229948326e-08
3.128284888521785e-08
3.054303024084361e-08
2.982070972671412e-08
2.911547397366853e-08
2.8426917712743742e-08
2.7754647830518233e-08
2.709827803310527e-08
2.6457432126379726e-08
2.5831743580832555e-08
2.5220853113719653e-08
2.4624411254667088e-08
2.404207565913181e-08
2.3473512702389147e-08
2.291839708378408e-08
2.237641031513366e-08
2.1847241734145764e-08
2.133058851067316e-08
2.082615397335663e-08
2.0333649706416375e-08
1.985279368976866e-08
1.9383309348707134e-08
1.892492861288687e-08
1.8477388838153433e-08
1.804043290212577e-08
1.761381126421899e-08
1.719727857968419e-08
1.6790597546551634e-08
1.639353354370977e-08
1.6005860549886923e-08
1.562735525586375e-08
1.52578014055265e-08
1.4896988217460045e-08
1.4544706441126885e-08
1.4200756939548522e-08
1.3864941251601634e-08
1.353706692761788e-08
1.3216946882020746e-08
1.2904397469244739e-08
1.259923893262848e-08
1.2301297218278755e-08
1.2010401589181631e-08
1.172638538273301e-08
1.1449085617552891e-08
1.1178343452800033e-08
1.091400404523621e-08
1.0655915770818267e-08
```

```
1.0403930972780374e-08
1.015790507810518e-08
9.91769779978657e-09

0.03478643165053965
0.017176607134176642
0.008908287720896737
0.004895804980415416
0.0028899833523879323
0.0018726067459914494
0.0013614491667520684
0.001104804881145592
0.0009657863065685706
0.0008765762041453425
0.0008079897763278004
0.0007488346437098814
0.0006951943813812279
0.0006458065695356919
0.0006002656868746771
0.00055837935735373842
0.0005199677869702755
0.0004848153181854769
0.0004526794172144722
0.0004233048721795122
0.0003964385676527968
0.00037183930458603055
0.00034928336937824813
0.0003285669713494072
0.0003095066358404662
0.00029193838749134895
0.0002757162979116769
0.0002607107653005607
0.00024680674677772963
0.00023390206657645444
0.00022190586151764083
0.00021073718780840157
0.00020032379180596013
0.00019060103592100625
0.00018151096546184327
0.00017300150036767773
0.00016502573588886887
0.00015754133739330693
0.0001505100159910492
0.00014389707336907862
0.00013767100579243304
0.0001318031587069122
0.0001262674246665699
0.00012103997842031016
0.0001160990439420222
0.00011142468900099697
0.00010699864353147725
0.00010280413864154643
9.882576356570442e-05
9.504933826429933e-05
9.146179971488056e-05
8.805110020669224e-05
8.480611619165726e-05
8.171656645211326e-05
7.877293849263061e-05
7.59664222236472e-05
7.328885012778352e-05
7.073264317862005e-05
6.829076190278796e-05
6.595666203140438e-05
6.372425425860633e-05
6.158786768034984e-05
5.954221654248014e-05
5.758236995865522e-05
5.5703724305730836e-05
5.390197803597782e-05
5.217310866345057e-05
5.0513351727928666e-05
4.891918153379885e-05
4.738729351346793e-05
4.5914588051293255e-05
4.449815563970486e-05
4.3135263248062386e-05
4.1823341788996664e-05
4.055997459051857e-05
3.934288678099527e-05
3.8169935508565864e-05
3.703910092184628e-05
3.594847784802681e-05
3.489626810817879e-05
3.3880773411152776e-05
3.2900388789574564e-05
3.195359651764573e-05
3.1038960485231554e-05
3.015512097679673e-05
2.930078983851417e-05
2.8474745987967768e-05
```

```
2.7675831244551563e-05
2.6902946460694375e-05
2.6155047918009685e-05
2.5431143981959186e-05
2.4730291980393458e-05
2.405159530495493e-05
2.3394200699764045e-05
2.27572957419168e-05
2.214010648471702e-05
2.1541895260868778e-05
2.0961958628357415e-05
2.0399625450736957e-05
1.985425510281936e-05
1.9325235790962946e-05
1.8811982981127407e-05
1.831393792430597e-05
1.7830566277192476e-05
1.7361356808296158e-05
1.690582018023035e-05
1.646348781323346e-05
1.6033910810269488e-05
1.561665895437216e-05
1.521131975930345e-05
1.481749758037296e-05
1.4434812777892462e-05
1.4062900925630548e-05
1.3701412070842658e-05
1.3350010031174368e-05
1.300837173437445e-05
1.2676186598171671e-05
1.2353155936555601e-05
1.2038992409653388e-05
1.173341949540791e-05
1.1436170994553595e-05
1.1146990562039674e-05
1.0865631262060301e-05
1.0591855148284979e-05
1.0325432866772266e-05
1.006614327806253e-05
9.813773099791988e-06
9.56811656978057e-06
9.328975122323448e-06
9.096157083257635e-06
8.869477384587841e-06
8.648757282936536e-06
8.433824103880229e-06
8.224510989974587e-06
8.020656666582484e-06
7.822105215292458e-06
7.62870586268508e-06
7.440312772954639e-06
7.256784857141099e-06
7.077985586481497e-06
6.90378281671768e-06
6.7340486191503235e-06
6.568659122942759e-06
6.407494357117076e-06
6.250438110577766e-06
6.097377786463482e-06
5.948204274268717e-06
5.8028118174219526e-06
5.661097895750585e-06
5.522963105569442e-06
5.388311051154215e-06
5.257048234265377e-06
5.129083955343618e-06
5.004330214786061e-06
4.88270161712162e-06
4.7641152843514965e-06
4.64849076638896e-06
4.535749960909688e-06
4.42581703305788e-06
4.318618339775483e-06
4.214082355222757e-06
4.112139603105315e-06
4.012722586586499e-06
3.91576572534089e-06
3.821205293787892e-06
3.728979357813839e-06
3.639027722602708e-06
3.55129187048752e-06
3.4657149149338917e-06
3.3822415440323075e-06
3.3008179715853166e-06
3.2213918935372543e-06
3.143912436339049e-06
3.06833011747393e-06
2.9945968011679737e-06
2.9226656553342803e-06
2.852491116280268e-06
2.784028847384192e-06
```

```
2.7172357021408935e-06
2.6520696913066715e-06
2.5884899439221863e-06
2.5264566806255477e-06
2.465931175407587e-06
2.406875727935491e-06
2.349253634240353e-06
2.2930291544283026e-06
2.2381674888921877e-06
2.1846347495471134e-06
2.1323979321549594e-06
2.081424892914552e-06
2.0316843255209318e-06
1.9831457337762654e-06
1.9357794107245183e-06
1.8895564179372456e-06
1.844448561287506e-06
1.8004283719640747e-06
1.7574690853504873e-06
1.7155446232586215e-06
1.674629572907126e-06
1.6346991705425237e-06
1.5957292820040835e-06
1.5576963862644536e-06
1.5205775600878492e-06
1.4843504589237995e-06
1.4489933055158888e-06
1.4144848683835144e-06
1.3808044548912458e-06
1.3479318905116855e-06
1.3158475078528201e-06
1.2845321332964286e-06
1.2539670721427928e-06
1.2241340986182124e-06
1.1950154406643626e-06
1.166593770380546e-06
1.138852190220026e-06
1.1117742237501547e-06
1.0853438038241717e-06
1.059545260991267e-06
1.0343633147462988e-06
1.009783062597877e-06
9.857899700439157e-07
9.623698618822038e-07
9.395089125954701e-07
9.171936360295695e-07
8.95410879948276e-07
8.74147814270975e-07
8.533919237827553e-07
8.331310007558993e-07
8.133531377817651e-07
7.940467167128279e-07
7.752004059595497e-07
7.568031498704387e-07
7.388441636868884e-07
7.213129238888979e-07
7.041991661862749e-07
6.874928752735953e-07
6.711842790433749e-07
6.552638443562153e-07
6.397222693409715e-07
6.245504777455707e-07
6.097396144395851e-07
5.952810387763661e-07
5.81166319805423e-07
5.673872303513348e-07
5.539357432411265e-07
5.408040249691801e-07
5.27984432070035e-07
5.15469505282168e-07
5.032519661996633e-07
4.913247114943467e-07
4.796808093333226e-07
4.683134959955254e-07
4.5721616952230226e-07
4.463823889136079e-07
4.3580586560901157e-07
4.254804646612036e-07
4.154001985236442e-07
4.0555922253552284e-07
3.959518332381088e-07
3.8657246477071045e-07
3.774156837324457e-07
3.6847618828987814e-07
3.597488037979844e-07
3.512284787297743e-07
3.4291028395780533e-07
3.347894082566168e-07
3.2686115472269556e-07
3.191209403255356e-07
3.115642912539636e-07
```

```
3.041868405995698e-07
2.96984326441726e-07
2.899525882490748e-07
2.8308756613517236e-07
2.7638529516490904e-07
2.6984190771183253e-07
2.6345362777402985e-07
2.5721676776595315e-07
2.5112773102579737e-07
2.4518300476576423e-07
2.3937916116159016e-07
2.337128532903711e-07
2.281808151662626e-07
2.2277985810564814e-07
2.175068688919961e-07
2.1235881005229113e-07
2.0733271541489248e-07
2.0242568959503945e-07
1.9763490646310757e-07
1.9295760750612376e-07
1.883910992102046e-07
1.8393275331348468e-07
1.7958000243120463e-07
1.7533034203077426e-07
1.7118132650079089e-07
1.6713056845566406e-07
1.6317573665944032e-07
1.5931455691347649e-07
1.5554480803450616e-07
1.518643214099081e-07
1.48270980649598e-07
1.4476271938598563e-07
1.4133752116868984e-07
1.3799341614608042e-07
1.347284817192795e-07
1.3154084220427564e-07
1.2842866432338573e-07
1.253901608428157e-07
1.2242358421001095e-07
1.1952723135229408e-07
1.1669943703558592e-07
1.1393857749507506e-07
1.1124306553264467e-07
1.0861135454001637e-07
1.0604193160461755e-07
1.0353332183879752e-07
1.0108408366502062e-07
9.869281176813201e-08
9.635813218630102e-08
9.407870490871402e-08
9.185322103997154e-08
8.968040265272875e-08
8.755900306610595e-08
8.548780428003537e-08
8.34656170761671e-08
8.149128131316793e-08
7.956366337393502e-08
7.768165791947849e-08
7.58441834246033e-08
7.40501867756147e-08
7.229863819102798e-08
7.058853224120115e-08
6.891888870724616e-08
6.728874851682629e-08
6.569717707580721e-08
6.414326126495749e-08
6.262610986582545e-08
6.114485201277891e-08
5.96986389168721e-08
5.828663998920236e-08
5.690804660007949e-08
5.5562066750901245e-08
5.424792931392222e-08
5.296488096246214e-08
5.171218510937592e-08
5.0489123196816773e-08
4.9294994963341824e-08
4.812911465890336e-08
4.6990814657631424e-08
4.587944140844398e-08
4.479435809645711e-08
4.3734942655398826e-08
4.2700587508148555e-08
4.169069964578615e-08
4.0704700370283864e-08
3.9742024127828985e-08
3.880211901818243e-08
3.788444629232807e-08
3.698848015041588e-08
3.611370669465051e-08
3.525962458993692e-08
```

```
3.442574404251154e-08
3.3611587652211387e-08
3.281668807995498e-08
3.204059020034263e-08
3.1282848886916944e-08
3.05430300122151e-08
2.9820710079131545e-08
2.9115473765655668e-08
2.8426917697590246e-08
2.7754647870848244e-08
2.7098278046179354e-08
2.645743220226046e-08
2.5831743565668087e-08
2.5220853142017608e-08
2.4624411076340566e-08
2.4042075645854485e-08
2.3473513073552363e-08
2.2918396692484538e-08
2.2376410443552085e-08
2.184724168197456e-08
2.1330588408574427e-08
2.0826154127050346e-08
2.0333649666000016e-08
1.985279374884295e-08
1.9383309217337433e-08
1.8924928676884915e-08
1.8477388741289535e-08
1.8040432895664907e-08
1.761381138245835e-08
1.7197278644178613e-08
1.6790597583504945e-08
1.6393533481160753e-08
1.6005860321187756e-08
1.5627355516572994e-08
1.5257801477268923e-08
1.4896987750898349e-08
1.454470682609392e-08
1.420075691172202e-08
1.3864941222385577e-08
1.3537066961045255e-08
1.3216946863693211e-08
1.2904397398195539e-08
1.2599238987585426e-08
```

In [23]:

```
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: 0.9649318027813342

Fidelity с применением MLE для МП ранка r = 1: 0.9894559236130683

Fidelity с применением MLE для МП ранка r = 2: 0.9812096940590345

Fidelity с применением MLE для МП ранка r = 3: 0.981209688987215

Fidelity с применением MLE для МП ранка r = 4: 0.9812097038154293
```

In [24]:

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

In [25]:

```
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 [26]:
```
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)
```

```
[34 52  9  3  7 33 12 46 11 44 33 10 35 50  5  9 62 34  0  2]
[33 55 10  2  5 33  9 53 13 40 34 13 39 46  6  9 64 34  0  2]
/home/stas/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: ComplexWarning: Casting complex values to real discards the imaginary part
  """
```

Out[26]:
```
(1, 0.009549385070038663)
```

Пункт 4:

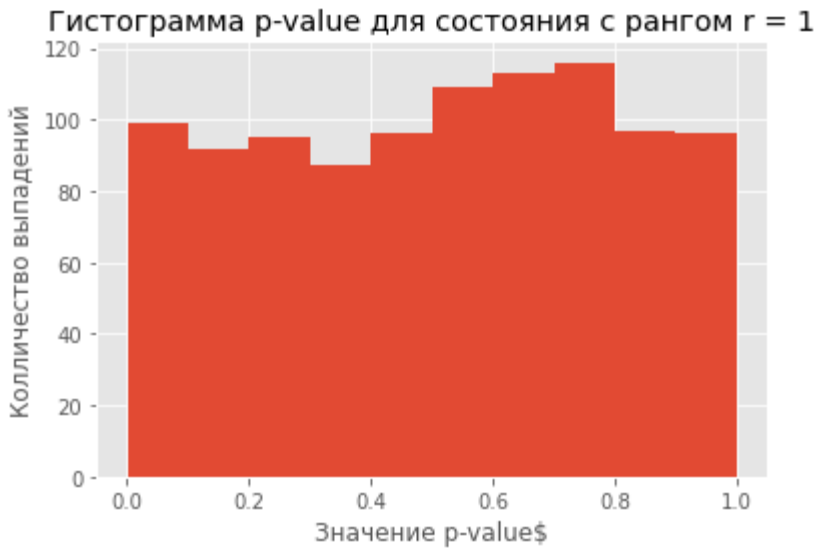
Сгенерируем статистические данные

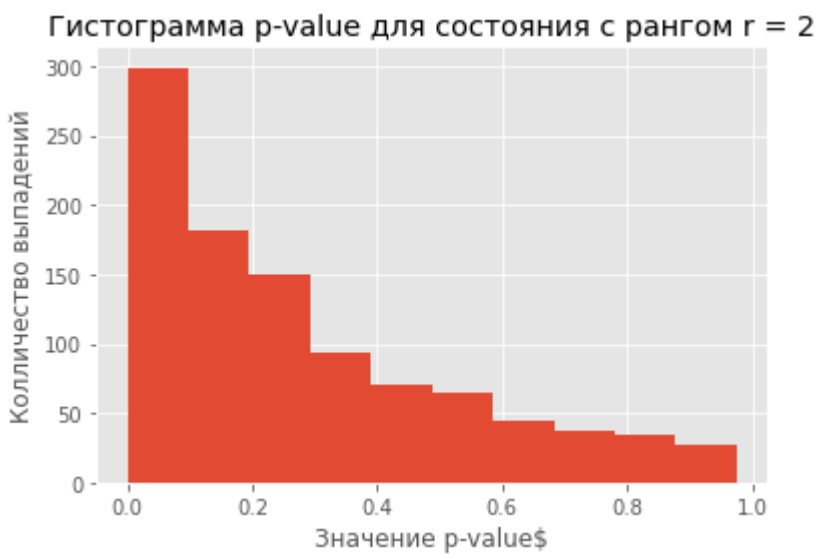
In [27]:
```
%%time
d = 4
n = 100
N = 1000

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
```

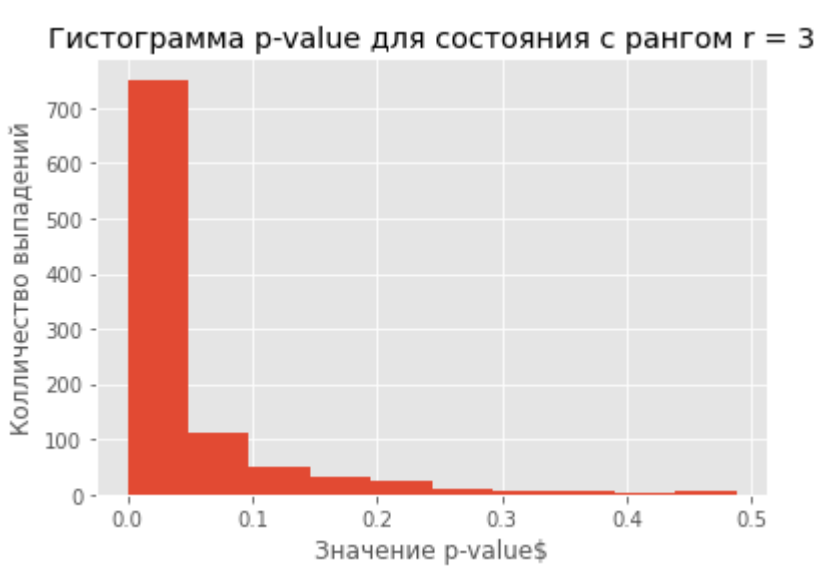
```
/home/stas/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:6: ComplexWarning: Casting complex values to real discards the imaginary part

CPU times: user 4min 21s, sys: 80.9 ms, total: 4min 21s
Wall time: 4min 21s
```

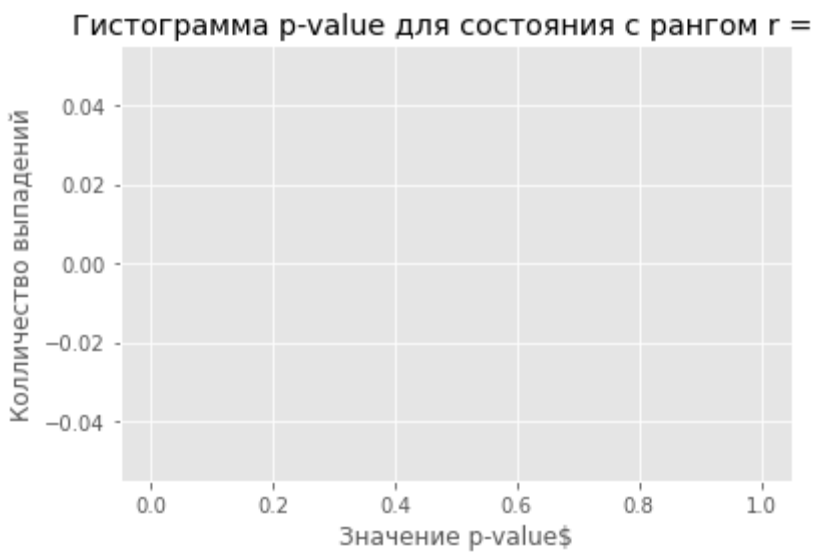
Построим распределение p-value для различных рангов r

In [28]:
```
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()
```





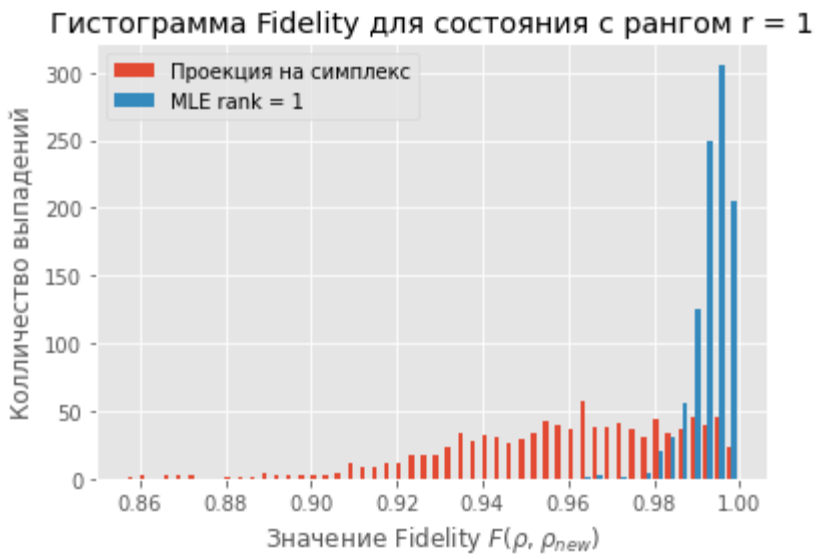


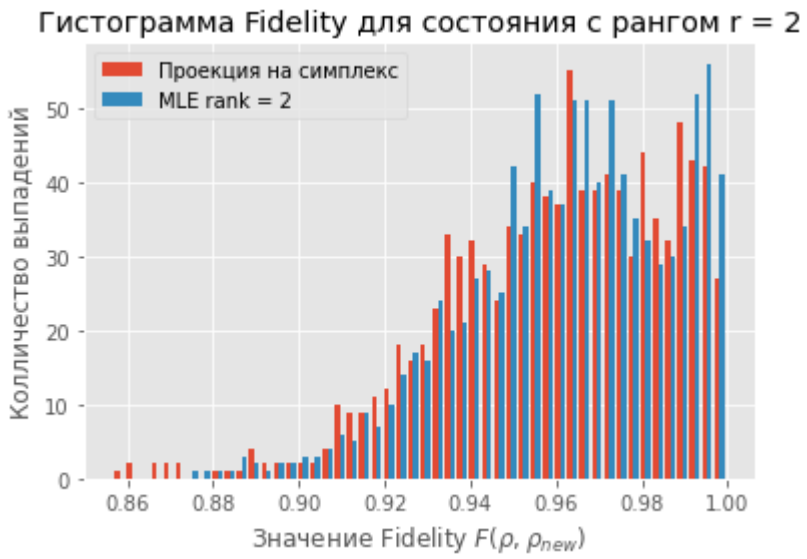


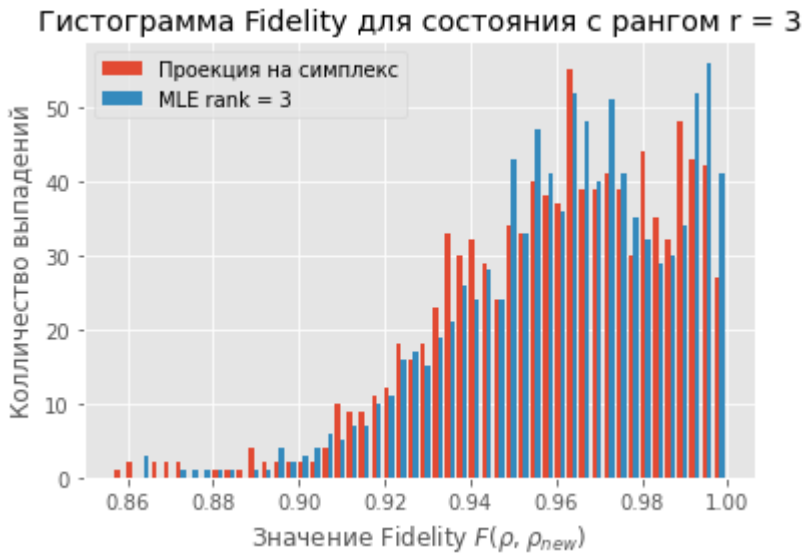
Адекватные результаты (равномерное p-value) наблюдается только для ранга r=1. Возможно такие результаты связаны с тем, что восстанавливаемое состояние чистое. Таким образом точность при оптимизации с начльным приближением большего ранга не даёт дополнительного прироста точности. В то же время количество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

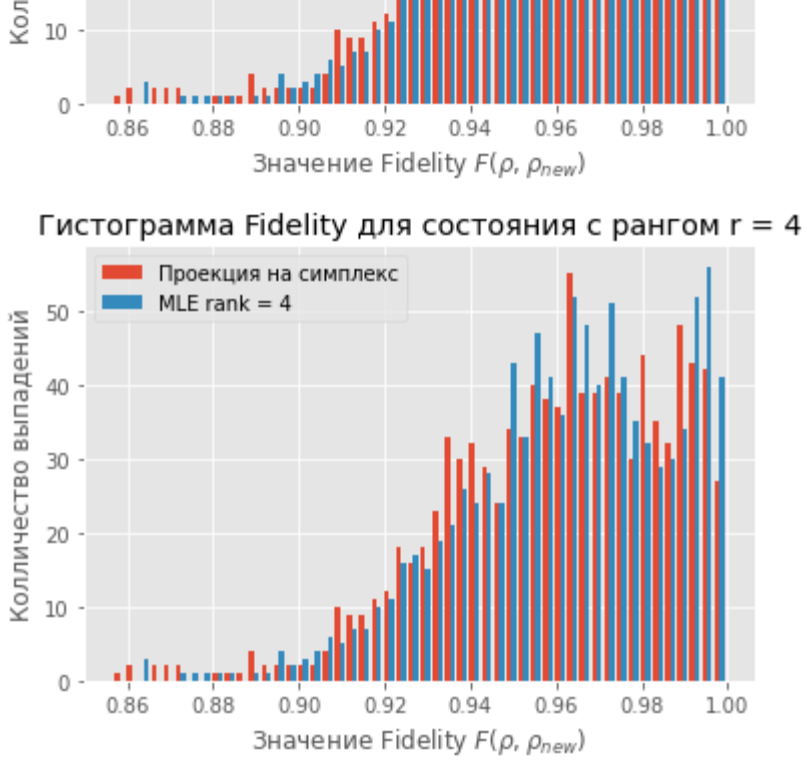
Сравним распределения фиделити для восстановленных состояний с применением MLE для каждого ранга с распределением фиделити до применения MLE.

In [29]:
```
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 [30]:
```
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 [31]:
```
def build_H(psi, P_list, prob_list, n_shots = 100):

#     n_shots = 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 [32]:
```
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 [33]:
```
F_list = Fidelity_MML_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 [34]:
```
prob_res
```

Out[34]:
```
array([0.12, 0.37, 0.28, 0.23, 0.01, 0.02, 0.78, 0.19, 0.27, 0.21, 0.35,
       0.17, 0.44, 0.04, 0.06, 0.46, 0.02, 0.63, 0.3 , 0.05])
```

In [53]:
```
# 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 = 2*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:6: ComplexWarning: Casting complex values to
real discards the imaginary part
```

Сравним результаты

In [54]:
```
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: 0.0054342808920368165
Статистическая оценка среднего ошибки: 0.006765810935330614

Теоретическая дисперсия ошибки: 1.2590535618708127e-05
Статистическая оценка дисперсии ошибки: 1.9167681007121383e-05
```

Теоритические величины приблизительно совпадают со статистическими значениями

In [ ]:

In [ ]: