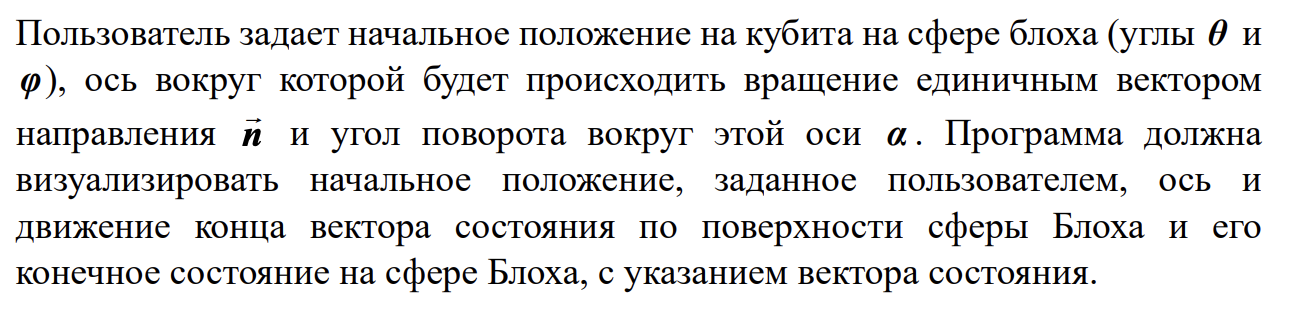
|  |  |
| --- | --- |
| Группа М32051 | К работе допущен |
| Студент Пантелеев, Артамонов | Работа выполнена |
| Преподаватель Ефремова Е.А. | Отчет принят |

**Рабочий протокол и отчет по  
задаче от лектора №2**

**Задание**



**Решение**

Пользователь либо оставляет по умолчанию, либо вводит в консоли следующие параметры:

1. углы и (начальное положение на сфере)
2. ось, вокруг которой будет происходить поворот (x, y или z)
3. угол поворота

Получаем оператор поворота кубита на угол относительно оси вектора

Действуем оператором на начальное состояние для получения конечного состояния, отображаем их на сфере Блоха

**Результат работы программы**

**A screenshot of a black screen

Description automatically generated with low confidence**

**Исходный код**

import numpy as np  
import qutip  
from cmath import phase  
from math import cos, sin, acos, e, pi  
from scipy.linalg import expm  
  
  
def pauli\_x():  
 return np.array([[0, 1], [1, 0]])  
  
  
def pauli\_y():  
 return np.array([[0, -1j], [1j, 0]])  
  
  
def pauli\_z():  
 return np.array([[1, 0], [0, -1]])  
  
  
def round\_complex(x, precision=10):  
 *"""  
 Округляет комплексное число.  
  
 :param x: комплексное число  
 :param precision: точность округления  
 :return: округленное комплексное число  
 """* return round(x.real, precision) + 1j \* round(x.imag, precision)  
  
  
def vector\_point\_to\_bloch\_angles(vector\_points):  
 *"""  
 Преобразует точки вектора в углы Эйлера на сфере Блоха.  
  
 :param vector\_points: точки вектора  
 :return: углы theta и phi на сфере Блоха  
 """* return 2 \* acos(round(abs(vector\_points[0]), 12)), phase(vector\_points[1]) - phase(vector\_points[0])  
  
  
def bloch\_angles\_to\_vector\_point(theta, phi):  
 *"""  
 Преобразует углы Эйлера на сфере Блоха в точки вектора.  
  
 :param theta: угол theta  
 :param phi: угол phi  
 :return: точки вектора  
 """* return np.array([cos(theta / 2), sin(theta / 2) \* e \*\* (1j \* phi)])  
  
  
def point\_coords\_by\_angles(theta, phi):  
 *"""  
 Вычисляет координаты (x, y, z) на сфере Блоха из углов theta и phi.  
  
 :param theta: угол theta  
 :param phi: угол phi  
 :return: массив координат (x, y, z)  
 """* x = cos(phi) \* sin(theta)  
 y = sin(phi) \* sin(theta)  
 z = cos(theta)  
 return np.array([x, y, z])  
  
  
def point\_on\_sphere(i, j, r0, R, phi):  
 *"""  
 Вычисляет координаты точки на сфере с центром в r0 и радиусом R.  
  
 :param i: координата i  
 :param j: координата j  
 :param r0: центр сферы  
 :param R: радиус сферы  
 :param phi: угол phi  
 :return: координаты точки на сфере  
 """* return r0 + R \* cos(phi) \* i + R \* sin(phi) \* j  
  
  
def normalize\_vector(v):  
 *"""  
 Нормализует вектор.  
  
 :param v: вектор  
 :return: нормализованный вектор  
 """* return v / np.linalg.norm(v)  
  
  
def rotation\_trace\_points(n, A, B, alpha, total\_points=50):  
 *"""  
 Вычисляет точки вращения вокруг оси n между точками A и B.  
  
 :param n: ось вращения  
 :param A: начальная точка  
 :param B: конечная точка  
 :param alpha: угол вращения  
 :param total\_points: общее количество точек  
 :return: список точек вращения  
 """* R = np.linalg.norm((B - A)) / (2 \* sin(abs(alpha) / 2))  
 j = normalize\_vector(np.array(B) - np.array(A))  
 if alpha > 0:  
 i = np.cross(j, n)  
 r0 = B - R \* (cos(alpha / 2) \* i + sin(alpha / 2) \* j)  
 else:  
 i = np.cross(n, j)  
 r0 = A - R \* (cos(alpha / 2) \* i + sin(alpha / 2) \* j)  
  
 angles = np.linspace(-alpha / 2, alpha / 2, total\_points)  
 points = [point\_on\_sphere(i, j, r0, R, phi) for phi in angles]  
  
 return points  
  
  
def format\_angle\_for\_view(x):  
 *"""  
 Форматирует угол для отображения.  
  
 :param x: угол  
 :return: отформатированное значение угла  
 """* x = round(x / pi, 10)  
 angle = ""  
 if x != 1:  
 angle += str(x)  
 if x != 0:  
 angle += "pi"  
 return angle  
  
  
def format\_vector\_point(vector\_state):  
 *"""  
 Форматирует состояние вектора для отображения.  
  
 :param vector\_state: состояние вектора  
 :return: отформатированное состояние вектора  
 """* x1 = round\_complex(vector\_state[0], 3)  
 x2 = round\_complex(vector\_state[1], 3)  
 return f'[{x1}, {x2}]'  
  
  
def format\_sphere\_coords(x, y, z):  
 *"""  
 Форматирует координаты (x, y, z) для отображения.  
  
 :param x: координата x  
 :param y: координата y  
 :param z: координата z  
 :return: отформатированные координаты (x, y, z)  
 """* return f'({round(x, 3)}, {round(y, 3)}, {round(z, 3)})'  
  
  
def get\_start\_values(input\_mode=False):  
 if not input\_mode:  
 print("Числитель для θ:\t1")  
 print("Знаменатель для θ:\t2")  
 print("Числитель для φ:\t1")  
 print("Знаменатель для φ:\t4")  
 print("Координатная ось:\tZ")  
 print("Числитель для α:\t-1")  
 print("Знаменатель для α:\t2")  
 return pi / 2, pi / 4, np.array([0, 0, 1]), -1 \* pi / 2  
  
 numerator\_theta = int(input("Введите числитель для θ:\t"))  
 denominator\_theta = int(input("Введите знаменатель для θ:\t"))  
 start\_theta = pi \* numerator\_theta / denominator\_theta  
  
 numerator\_phi = int(input("Введите числитель для φ:\t"))  
 denominator\_phi = int(input("Введите знаменатель для φ:\t"))  
 start\_phi = pi \* numerator\_phi / denominator\_phi  
  
 chosen\_axis = int(input("Выберите координатную ось: 0 - x, 1 - y, 2 - z:\t")) % 3  
 array = [0, 0, 0]  
 array[chosen\_axis] = 1  
 axis = np.array(array)  
  
 numerator\_alpha = int(input("Введите числитель для α:\t"))  
 denominator\_alpha = int(input("Введите знаменатель для α:\t"))  
 alpha = normalize\_vector(pi \* numerator\_alpha / denominator\_alpha)  
  
 return start\_theta, start\_phi, axis, alpha  
  
  
start\_theta, start\_phi, axis, alpha = get\_start\_values(input\_mode=False)  
nx, ny, nz = axis[0], axis[1], axis[2]  
U = expm(-1j \* alpha / 2 \* (nx \* pauli\_x() + ny \* pauli\_y() + nz \* pauli\_z()))  
  
start\_vector\_point = bloch\_angles\_to\_vector\_point(start\_theta, start\_phi)  
end\_vector\_point = np.dot(U, start\_vector\_point)  
  
theta, phi = vector\_point\_to\_bloch\_angles(start\_vector\_point)  
start\_point = point\_coords\_by\_angles(theta, phi)  
  
print()  
print('Начало вектора')  
print(f'θ: {format\_angle\_for\_view(theta)}')  
print(f'φ: {format\_angle\_for\_view(phi)}')  
print(f'Точка: {format\_vector\_point(start\_vector\_point)}')  
print(f'Координаты: {format\_sphere\_coords(\*start\_point)}')  
  
theta, phi = vector\_point\_to\_bloch\_angles(end\_vector\_point)  
end\_point = point\_coords\_by\_angles(theta, phi)  
  
print()  
print('Конец вектора')  
print(f'θ: {format\_angle\_for\_view(theta)}')  
print(f'φ: {format\_angle\_for\_view(phi)}')  
print(f'Точка: {format\_vector\_point(end\_vector\_point)}')  
print(f'Координаты: {format\_sphere\_coords(\*end\_point)}')  
  
b = qutip.Bloch()  
b.vector\_color = ['#000000']  
b.vector\_width = 5  
b.point\_color = ['b', 'r', '#00FF00']  
b.add\_points(start\_point)  
b.add\_points(end\_point)  
b.add\_vectors(axis)  
if not np.array\_equal(start\_point, end\_point):  
 rotation\_points = rotation\_trace\_points(axis, start\_point, end\_point, alpha)  
 b.add\_points(list(zip(\*rotation\_points)), meth='l')  
b.show()

Исходный код опубликован на GitHub и может быть скачан и запущен:

<https://github.com/SmartOven/Python/blob/master/Physics/LecTask2/main.py>