ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ CИСТЕМЫ»**

**ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА И ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ»**

**ВАРИАНТ ЗАДАНИЯ № 23**

Выполнил(а) студент группы М8О-201Б-23

Тутаев В.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Зав. каф. 802, Волков Е.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

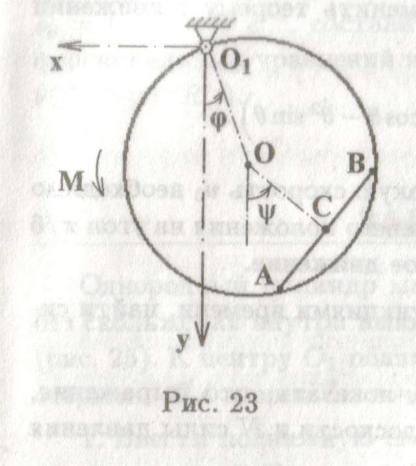
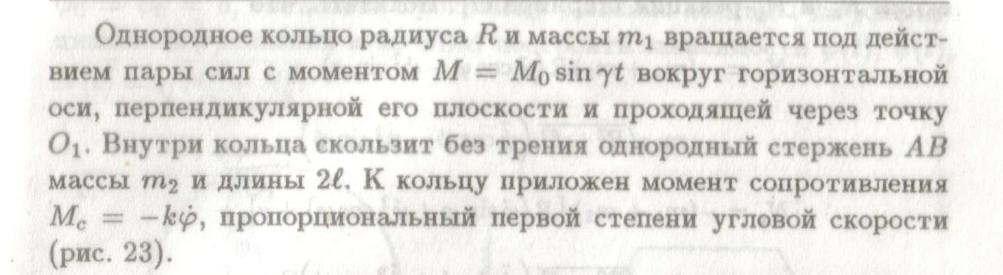
подпись, дата

с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2024

*Задание:* построить анимацию движения системы с помощью Python.

*Вариант 23:*



*Код:*

*Файл lab2.py:*

import numpy as np  
import matplotlib.pyplot as plt  
from matplotlib.animation import FuncAnimation  
import sympy as sp  
import math  
  
# Функция для поворота координат на угол Alpha  
def Rot2D(X, Y, Alpha):  
 RX = X \* np.cos(Alpha) - Y \* np.sin(Alpha)  
 RY = X \* np.sin(Alpha) + Y \* np.cos(Alpha)  
 return RX, RY  
  
# Параметры системы  
R = 5  
l = 2  
r = math.sqrt(R \* R - l \* l)  
  
# Символьная переменная времени  
t = sp.Symbol('t')  
  
# Углы поворота, зависящие от времени  
phi = 0.3 \* t  
ksi = 0.5 \* t  
  
# Координаты точки O (движение по окружности радиуса R)  
x\_O = -R \* sp.sin(phi)  
y\_O = R \* sp.cos(phi)  
  
# Координаты точки C (относительно точки O)  
x\_C = x\_O - r \* sp.sin(ksi)  
y\_C = y\_O + r \* sp.cos(ksi)  
  
# Относительные координаты точки C относительно точки O  
x\_rel = -r \* sp.sin(ksi)  
y\_rel = r \* sp.cos(ksi)  
  
# Скорость точки C (производная координат по времени)  
Vx\_C = sp.diff(x\_C, t)  
Vy\_C = sp.diff(y\_C, t)  
V\_mod\_C = sp.sqrt(Vx\_C\*\*2 + Vy\_C\*\*2)  
  
# Ускорение точки C (вторая производная координат по времени)  
Ax\_C = sp.diff(x\_C, t, 2)  
Ay\_C = sp.diff(y\_C, t, 2)  
A\_mod\_C = sp.sqrt(Ax\_C\*\*2 + Ay\_C\*\*2)  
  
# Массив времени для анимации  
T = np.linspace(0, 45, 1000)  
  
# Преобразование символьных выражений в численные функции  
X\_O\_def = sp.lambdify(t, x\_O, modules='numpy')  
Y\_O\_def = sp.lambdify(t, y\_O, modules='numpy')  
X\_C\_def = sp.lambdify(t, x\_C, modules='numpy')  
Y\_C\_def = sp.lambdify(t, y\_C, modules='numpy')  
X\_REL\_def = sp.lambdify(t, x\_rel, modules='numpy')  
Y\_REL\_def = sp.lambdify(t, y\_rel, modules='numpy')  
  
# Вычисление координат для всех моментов времени  
X\_O = X\_O\_def(T)  
Y\_O = Y\_O\_def(T)  
X\_C = X\_C\_def(T)  
Y\_C = Y\_C\_def(T)  
X\_REL = X\_REL\_def(T)  
Y\_REL = Y\_REL\_def(T)  
  
# Настройка графика  
fig = plt.figure(figsize=(10, 8))  
ax1 = fig.add\_subplot(1, 1, 1)  
ax1.set\_aspect('equal')  
ax1.set\_xlim([-10, 10])  
ax1.set\_ylim([-20, 20])  
ax1.set\_xlabel('Ось x')  
ax1.set\_ylabel('Ось y')  
ax1.invert\_xaxis()  
ax1.invert\_yaxis()  
ax1.grid(True)  
  
# Начальное состояние графиков  
PointO1, = ax1.plot([0], [0], 'bo')  
Circ\_Angle = np.linspace(0, 2 \* np.pi, 100)  
Circ, = ax1.plot(X\_O[0] + R \* np.cos(Circ\_Angle), Y\_O[0] + R \* np.sin(Circ\_Angle), 'g')  
  
# Начальное состояние стержня  
ArrowX = np.array([0, 0, 0])  
ArrowY = np.array([l, 0, -l])  
R\_Stick\_ArrowX, R\_Stick\_ArrowY = Rot2D(ArrowX, ArrowY, math.atan2(Y\_REL[0], X\_REL[0]))  
Stick\_Arrow, = ax1.plot(R\_Stick\_ArrowX + X\_C[0], R\_Stick\_ArrowY + Y\_C[0], 'r-')  
  
# Линии, соединяющие точки  
O1O, = ax1.plot([0, X\_O[0]], [0, Y\_O[0]], 'b:')  
OC, = ax1.plot([X\_O[0], X\_C[0]], [Y\_O[0], Y\_C[0]], 'b:')  
  
# Функция для обновления анимации  
def anima(i):  
 O1O.set\_data([0, X\_O[i]], [0, Y\_O[i]])  
 OC.set\_data([X\_O[i], X\_C[i]], [Y\_O[i], Y\_C[i]])  
 Circ.set\_data(X\_O[i] + R \* np.cos(Circ\_Angle), Y\_O[i] + R \* np.sin(Circ\_Angle))  
  
 # Поворот стержня  
 angle = math.atan2(Y\_REL[i], X\_REL[i])  
 R\_Stick\_ArrowX, R\_Stick\_ArrowY = Rot2D(ArrowX, ArrowY, angle)  
 Stick\_Arrow.set\_data(R\_Stick\_ArrowX + X\_C[i], R\_Stick\_ArrowY + Y\_C[i])  
  
 return O1O, OC, Circ, Stick\_Arrow, PointO1  
  
anim = FuncAnimation(fig, anima, frames=len(T), interval=20, blit=True)  
  
plt.show()

*Процесс выполнения работы:*

1. Инициализация параметров и функций:

* Задаются параметры системы: радиус R, длина стержня 2l, масса m1​, m2​, момент сопротивления Mc​, и момент внешней силы M.
* Определяются зависимости углов φ и ξ от времени t, а также координаты точек OO и CC.

2. Аналитические вычисления:

* Используется библиотека SymPy для нахождения скоростей и ускорений точек C и O.
* Вычисляются модули скорости и ускорения, их аналитические выражения преобразуются в численные функции.

3. Генерация данных:

* Создаются массивы данных для времени T, координат, скоростей и ускорений с помощью NumPy.

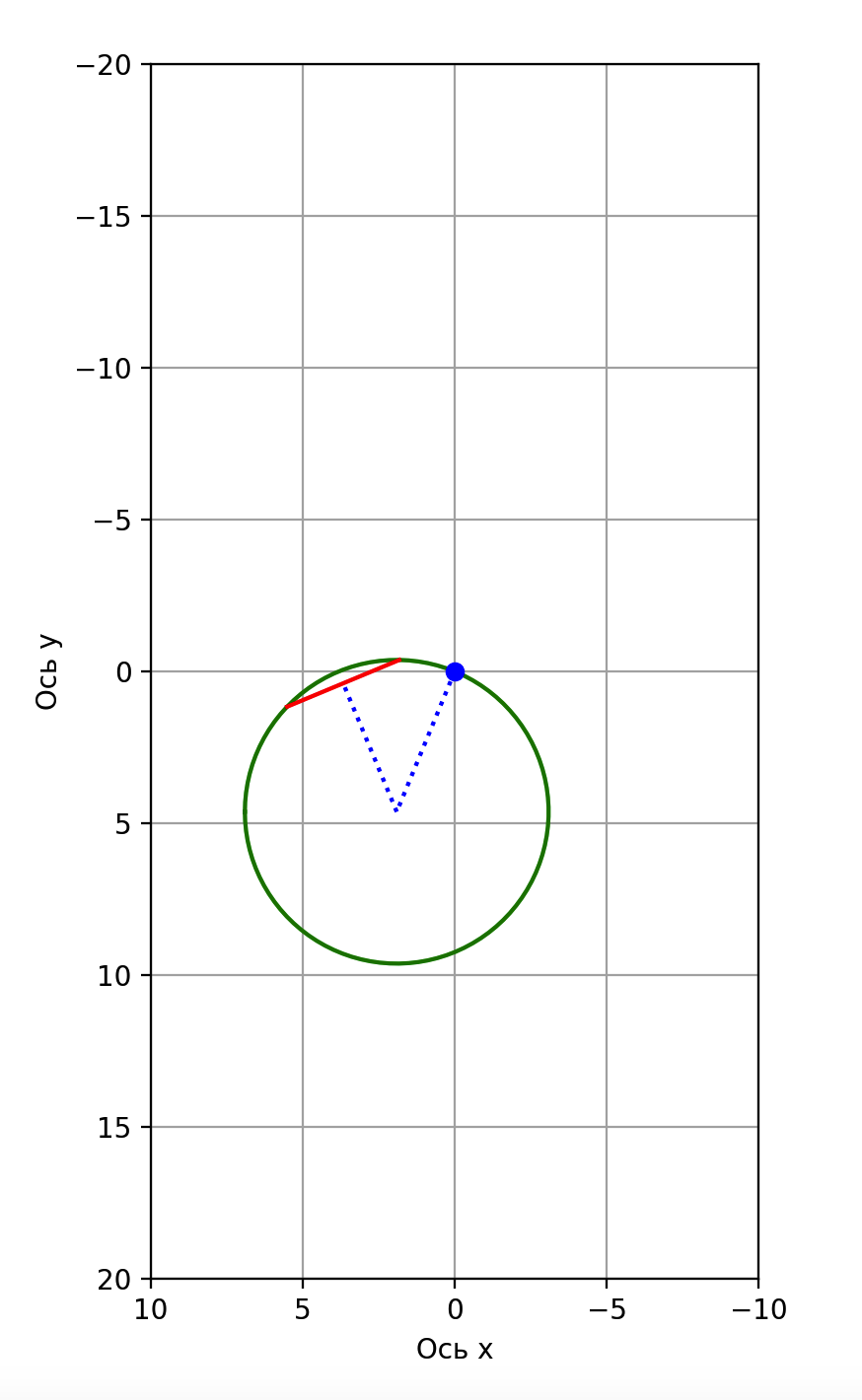
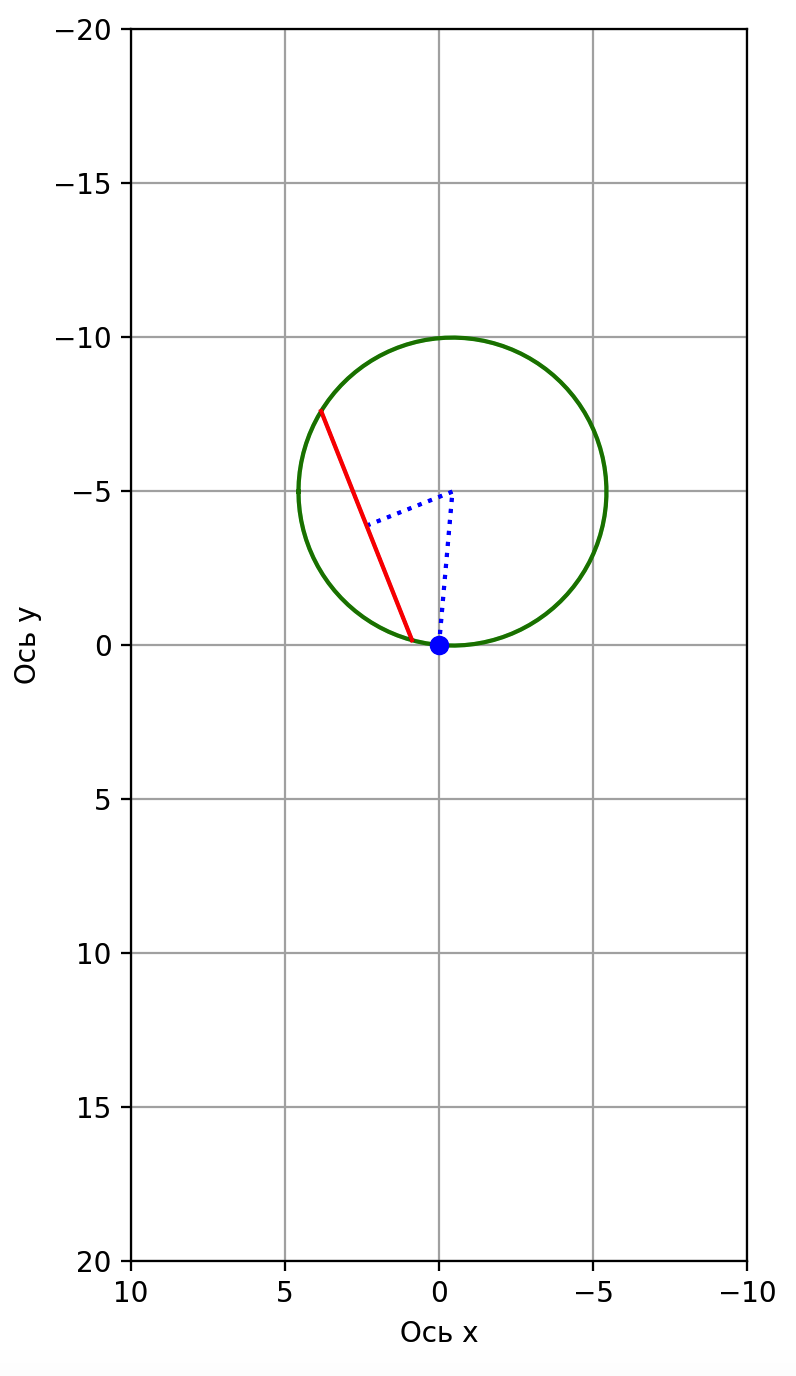
4. Визуализация:

* Используется Matplotlib для создания графиков движения кольца и стержня в реальном времени.
* Анимация включает:
* Изменение положения кольца и стержня.
* Построение графиков зависимости скорости и ускорения от времени.

5. Анимация движения:

* Реализована функция обновления кадров, которая пересчитывает координаты и графики для каждого момента времени.

*Результат работы программы:*

На данных изображениях можно увидеть анимацию системы при двух различных значениях радиуса диска и длины стержня. Зависимость углов поворота от времени в данном случае произвольна.

*Вывод:* научился использованию средств языка Python для моделирования сложного движения элементов системы. Вместо произвольных зависимостей теперь можно вставить закон движения, чтобы увидеть реальное движение заданной системы.