ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ОТЧЕТ

О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«АНИМАЦИЯ СИСТЕМЫ»

ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА И ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ»

ВАРИАНТ ЗАДАНИЯ №41

Выполнил(а) студент группы М8О-201Б-22

Шляхтуров Александр Викторович______________________________

подпись, дата

Проверил и принял

Авдюшкин А.Н.______________________________

подпись, дата

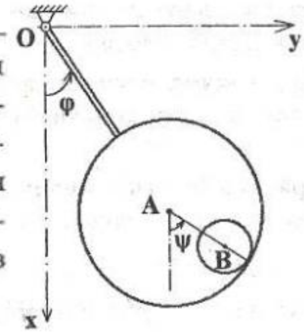
с оценкой ______________________________

Москва, 2023

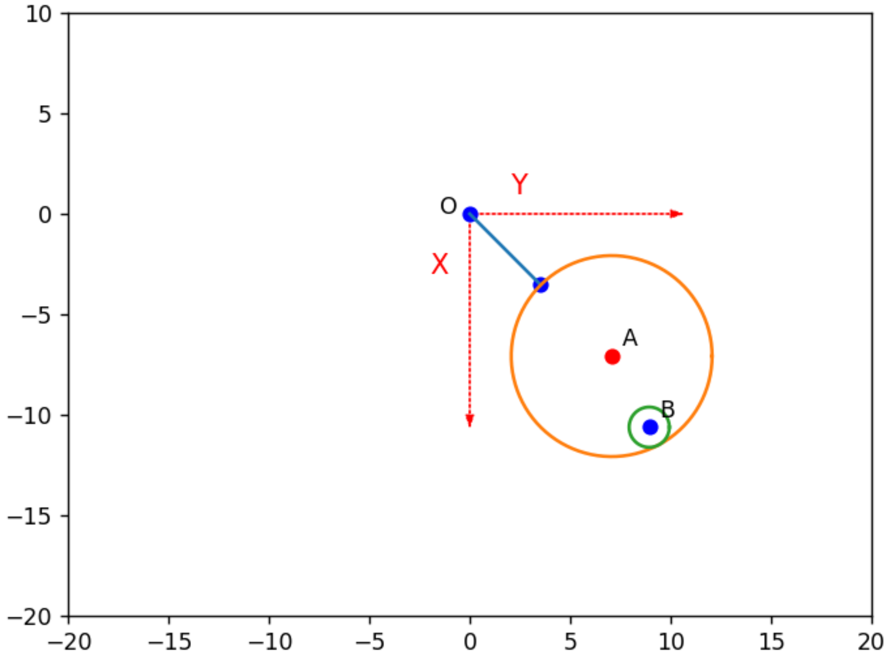
Задание: построить анимацию движения системы с помощью Python.

1) *Условия задачи 41 варианта:*

Механическая система состоит из тонкого однородного стержня массы m_1 и длины ℓ, жестко спаянного с ним однородного тонкостенного цилиндра A массы m_2 и радиуса R и сплошного однородного цилиндра B массы m_3 и радиуса r, который может катиться без проскальзывания по внутренней поверхности цилиндра A. Система закреплена в точке O в неподвижном шарнире и находится в поле тяжести.



2) *Рисунок получившейся физической модели:*



3) *Код программы:*

4)
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import math

l = 5
r1 = 5
r2 = 1
t = np.linspace(0, 50, 500)

phi = np.sin(t)
ksi = np.cos(t)*np.sin(t)

StickBX = l * np.sin(phi)
StickBY = -l * np.cos(phi)
```

```
CenterX = (l + r1) * np.sin(phi)
CenterY = -(l + r1) * np.cos(phi)

Center2X = CenterX + (r1-r2)*np.sin(ksi)
Center2Y = CenterY - (r1-r2)*np.cos(ksi)

fig = plt.figure()
ax1 = fig.add_subplot()
ax1.axis('equal')
plt.gca().set_adjustable("box")
ax1.set(xlim=[-20, 20], ylim=[-20, 10])


ax1.arrow(0, 0, 10, 0, head_width=0.4, head_length=0.6, fc='r', ec='r',
linestyle='dashed', linewidth=0.5)
ax1.arrow(0, 0, 0, -10, head_width=0.4, head_length=0.6, fc='r',
ec='r', linestyle='dashed', linewidth=0.5)

# Подписываем оси координат
ax1.text(2, 1, 'Y', fontsize=12, color='r')
ax1.text(-2, -3, 'X', fontsize=12, color='r')


ax1.plot(0, 0, marker='o', c='b')
OText = plt.text(-1.5, 0, 'O')
AText = plt.text(CenterX[0] + 0.5, CenterY[0] + 0.5, 'A')
BText = plt.text(Center2X[0] + 0.5, Center2X[0]  + 0.5, 'B')
# CText = plt.text(CX[0] + forLetters, CY[0]  + forLetters, 'C')


StickB = ax1.plot(StickBX[0], StickBY[0], marker='o', c='b')[0]
ABLine = ax1.plot([0, StickBX[0]], [0, StickBY[0]])[0]

Center = ax1.plot(CenterX[0], CenterY[0], marker='o', c='r')[0]
Center2 = ax1.plot(Center2X[0], Center2Y[0], marker='o', c = 'b')[0]

phiForCirc = np.linspace(0, 2 * math.pi, 100)
Circ = ax1.plot(CenterX[0] + r1 * np.cos(phiForCirc), CenterY[0] + r1 *
np.sin(phiForCirc))[0]
Circ2 = ax1.plot(Center2X[0] + r2 * np.cos(phiForCirc), Center2Y[0] +
r2 * np.sin(phiForCirc))[0]


def anima(i):
    StickB.set_data([StickBX[i]], [StickBY[i]])
    ABLine.set_data([0, StickBX[i]], [0, StickBY[i]])
    Center.set_data([CenterX[i]], [CenterY[i]])
    Center2.set_data([Center2X[i]], [Center2Y[i]])
    Circ.set_data(CenterX[i] + r1 * np.cos(phiForCirc), CenterY[i] + r1
* np.sin(phiForCirc))
    Circ2.set_data(Center2X[i] + r2 * np.cos(phiForCirc), Center2Y[i] +
r2 * np.sin(phiForCirc))
    AText.set_position([CenterX[i] + 0.5, CenterY[i] + 0.5])
    BText.set_position([Center2X[i] + 0.5, Center2Y[i] + 0.5])

anim = FuncAnimation(fig, anima, frames=100, interval=100)

plt.show()
```

4) *Пояснение*

В этой лабораторной работе функции phi(t) и ksi(t) мы инициализируем как угодно, ведь целью является просто смоделировать движение представленной системы в пространстве. Моделировать систему мы будем поэтапно. Первым делом пропишем, как будет двигаться нижняя точка стержня, к которой будет в дальнейшем прикреплен больший цилиндр. Координаты этой точки зависят от выбранной нами функции phi(t). Зная в каждый момент времени положение нижней точки стержня, легко выражается координата центра большего цилиндра. Последним этапом мы связываем координату центра меньшего цилиндра с координатой центра большего цилиндра. За эту связь отвечает также выбранная нами функция ksi(t). В конечном счете мы создаем функцию анимации, которая для разбиения t подсчитывает положения всех вышеупомянутых точек.

5) *Вывод*

В этой лабораторной работе я замоделировал механическую систему движущихся тел с помощью языка Python и библиотеки для визуализации Matplotlib. Для создания правильной модели движения нужно было выражать координаты движения основных точек системы через две функции phi(t) и ksi(t). В результате получилась красивая цветная реализация механической системы с подписанными точками.