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 №23

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

Шляхтуров Александр Викторович______________________________

подпись, дата

Проверил и принял

Авдюшкин А.Н.______________________________

подпись, дата

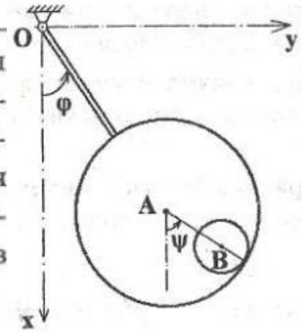
с оценкой ______________________________

Москва, 2023

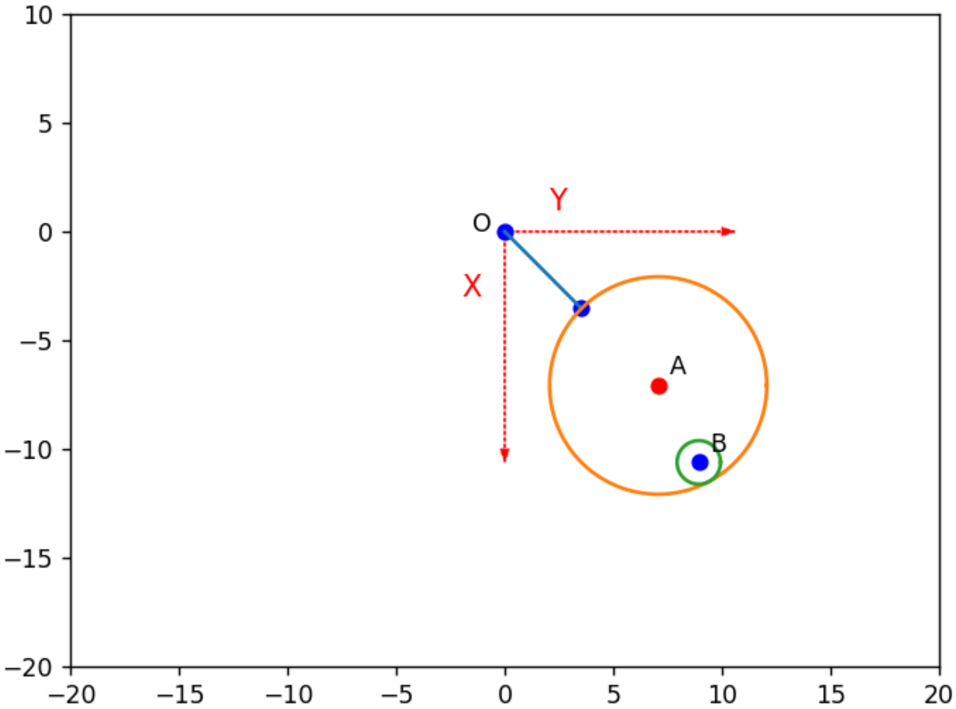
<u>Задание:</u> проинтегрировать систему дифференциальных уравнений движения системы с двумя степенями свободы с помощью средств Python. Построить анимацию движения системы, а также графики законов движения системы и указанных в задании реакций для разных случаев системы.

1) *Условия задачи 23 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:*

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import math
from scipy.integrate import odeint

def odesys(y, t, m1, m2, m3, l, r1, r2):
    dy = np.zeros(4)
    dy[0] = y[2]
    dy[1] = y[3]

    a11 = (m1/3)*l*l + (m2+m3)*(r1+l)*(r1+l) + (m2 + (m3/2))*r1*r1
    a12 = m3*(r1-r2)*((r1+l)*np.cos(y[0] - y[1]) - (r1/2))
    a21 = (r1 + l)*np.cos(y[0] - y[1]) - (r1/2)
    a22 = (3/2)*(r1-r2)

    b1 = -((m1/2)*l + (m2+m3)*(r1+l))*9.8*np.sin(y[0]) - m3*(r1 -
r2)*((r1+l)*np.sin(y[0] - y[1])*y[3]*y[3])
    b2 = (r1 + l)*np.sin(y[0] - y[1])*y[2]*y[2] - 9.8*np.sin(y[1])

    dy[2] = (b1*a22 - b2*a12)/(a11*a22 - a12*a21)
    dy[3] = (b2*a11 - b1*a21)/(a11*a22 - a12*a21)

    return dy

time = 70
t = np.linspace(0, time, 700)

phi0 = np.pi*(1/6)
ksi0 = np.pi*(1/3)
dphi0 = 0
dksi0 = 0

l = 1
r1 = 0.5
r2 = 0.2

m1 = 2
m2 = 5
m3 = 3

y0 = [phi0, ksi0, dphi0, dksi0]

Y = odeint(odesys, y0, t, (m1, m2, m3, l, r1, r2))


phi = Y[:,0]
ksi = Y[:,1]
# 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=[-3, 3], ylim=[-3, 2])


ax1.arrow(0, 0, 2, 0, head_width=0.1, head_length=0.3, fc='r', ec='r',
linestyle='dashed', linewidth=0.1)
ax1.arrow(0, 0, 0, -2, head_width=0.1, head_length=0.3, fc='r', ec='r',
linestyle='dashed', linewidth=0.1)

# Подписываем оси координат
ax1.text(0.5, 0.25, 'Y', fontsize=12, color='r')
ax1.text(-0.5, -0.75, 'X', fontsize=12, color='r')


ax1.plot(0, 0, marker='o', c='b')
OText = plt.text(-0.5, 0, 'O')
AText = plt.text(CenterX[0] + 0.1, CenterY[0] + 0.1, 'A')
BText = plt.text(Center2X[0] + 0.1, Center2X[0]  + 0.1, 'B')


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] + 0.1, CenterY[i] + 0.1])
    BText.set_position([Center2X[i] + 0.1, Center2Y[i] + 0.1])

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

anim_running = True
def onClick(event):
    global anim_running
    if anim_running:
        anim.event_source.stop()
        anim_running = False
    else:
        anim.event_source.start()
        anim_running = True
fig.canvas.mpl_connect('button_press_event', onClick)


fig_for_graphs = plt.figure(figsize=[8,7])
ax_for_graphs = fig_for_graphs.add_subplot(2, 1, 1)
ax_for_graphs.plot(t, phi, color='Blue')
ax_for_graphs.set_title("phi(t)")
```

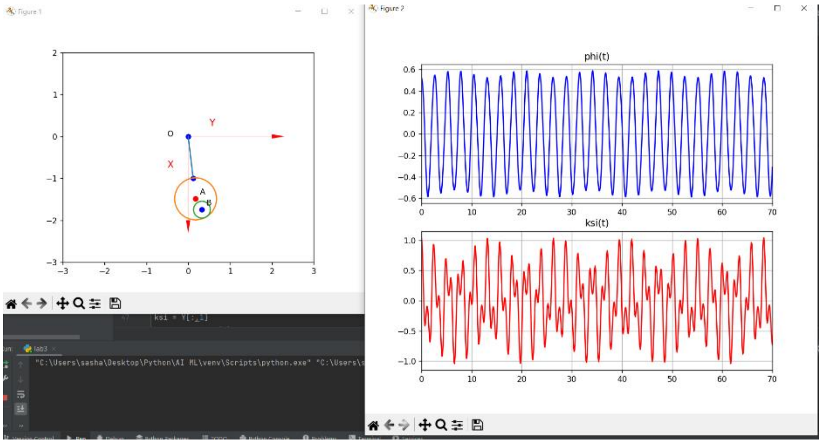
```
ax_for_graphs.set(xlim=[0, time])
ax_for_graphs.grid(True)

ax_for_graphs = fig_for_graphs.add_subplot(2, 1, 2)
ax_for_graphs.plot(t, ksi, color='Red')
ax_for_graphs.set_title("ksi(t)")
ax_for_graphs.set(xlim=[0, time])
ax_for_graphs.grid(True)


plt.show()
```

4) *Вариация констант и начальных условий:*

Константы: m1 = 2 кг, m2 = 5 кг, m3 = 3 кг, g = 9.81, l = 1 м, r1 = 0.5 м, r2 = 0.2 м

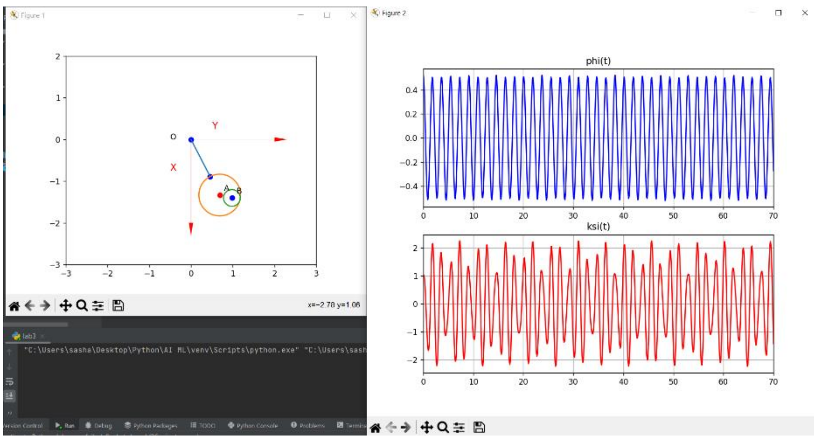
Начальные условия: $\varphi_0 = \pi/6$, $\psi_0 = \pi/3$, $d\varphi_0/dt = 0$, $d\psi_0/dt = 0$

a. m1 = 2 кг, m2 = 5 кг, m3 = 3 кг



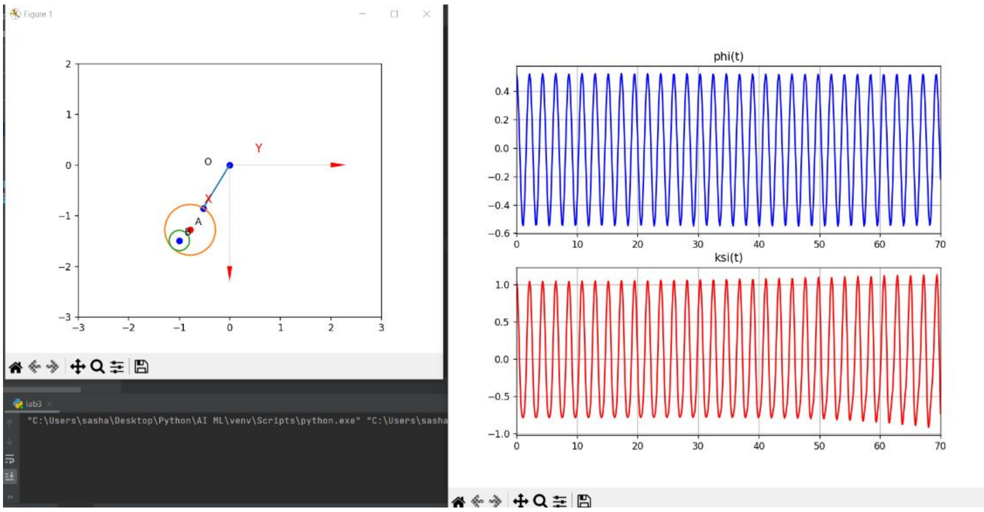
Графики зависимости phi(t) и ksi(t) при массах, данных в условии

b. m1 = 200 кг. m2 = 5 кг, m3 = 3 кг



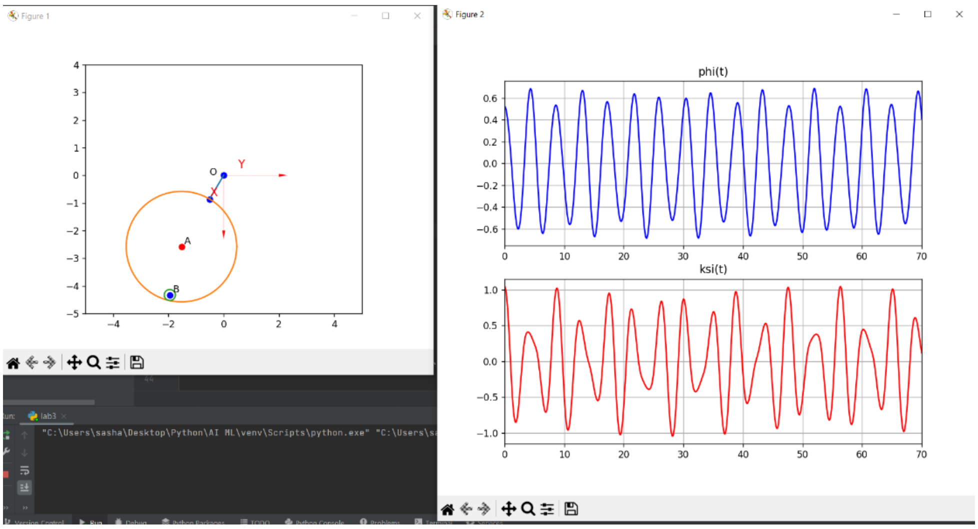
Увеличивая массу стержня, мы наблюдаем более равномерные колебания угла phi и ksi, поскольку масса малого цилиндра слишком мала, чтобы существенно повлиять на колеблющийся тяжелый стержень. Поэтому даже при движении в противофазе колебания меньшего цилиндра и стержня практически не гасят друг друга.

c. m1 = 200 кг, m2 = 5 кг, m3 = 30 кг



Экспериментально были подобраны такие массы грузов, чтобы колебания угла phi и ksi происходили практически софазно и с одинаковым периодом колебаний. В таком случае мы наблюдаем колебания обеих величин с неизменной амплитудой, поскольку меньший цилиндр не воздействует на больший в сторону, противоположную его движению.

d. r1 = 2 м при остальных параметрах без изменения



Увеличивая радиус большего цилиндра мы увеличиваем период колебаний угла phi.

5. *Вывод*

Выполняя эту лабораторную работу, я научился решать дифференциальные уравнения с помощью библиотеки scipy и ее модуля odeint для языка Python. После решения уравнений стало возможным построить анимацию движения системы в гравитационном поле и при заданных параметрах длины стержня и массах цилиндра. Также были построены графики phi(t) и ksi(t).