ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«ДИНАМИКА СИСТЕМЫ»**

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

**ВАРИАНТ ЗАДАНИЯ №21**

Выполнил(а) студент группы М8О-203Б-22

Клименко В.М.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Авдюшкин А.Н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Москва, 2023

*Задание:* 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.

*Код:*

import math

import numpy

import sympy

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from scipy.integrate import odeint

def draw\_circle(X, Y):

CX = [X + r \* math.cos(i / 10) for i in range(0, 64)]

CY = [Y + r \* math.sin(i / 10) for i in range(0, 64)]

return CX, CY

def animation(i):

beam1.set\_data([-l, -l + X\_ab[i]], [0, Y\_ab[i]])

beam2.set\_data([l, l + X\_ab[i]], [0, Y\_ab[i]])

beam3.set\_data([-l + X\_ab[i], l + X\_ab[i]], [Y\_ab[i], Y\_ab[i]])

beam4.set\_data([X\_ab[i], X\_ab[i] + X\_o[i]], [Y\_ab[i], Y\_ab[i] + Y\_o[i]])

circle.set\_data(\*draw\_circle(X\_ab[i] + X\_o[i], Y\_ab[i] + Y\_o[i]))

return beam1, beam2, beam3, beam4, circle

def formY(y, t, f\_omega\_theta, f\_omega\_phi):

y1, y2, y3, y4 = y

dydt = [y4, y3, f\_omega\_theta(y1, y2, y3, y4), f\_omega\_phi(y1, y2, y3, y4)]

return dydt

g = 10

m\_1 = 5

m\_2 = 5

l = 1

b = 0.125

k = 0.1

r = 0.25

t = sympy.Symbol('t')

phi = sympy.Function('phi')(t)

theta = sympy.Function('theta')(t)

omega\_phi = sympy.Function('omega\_phi')(t)

omega\_theta = sympy.Function('omega\_theta')(t)

E\_ab = (m\_2 \* omega\_phi\*\*2 \* l\*\*2) / 2

Vx\_c = omega\_phi \* l \* sympy.sin(phi)

Vy\_c = omega\_phi \* l \* sympy.cos(phi)

Vx\_o = omega\_theta \* sympy.sin(theta) \* b

Vy\_o = omega\_theta \* sympy.cos(theta) \* b

J\_cir = (m\_1 \* r\*\*2) / 2

E\_cir = m\_1 \* ((Vx\_o + Vx\_c)\*\*2 + (Vy\_o + Vy\_c)\*\*2) / 2 + (J\_cir \* omega\_theta\*\*2)/2

Ekin = E\_ab + E\_cir

Pi1 = m\_1 \* g \* (l \* (1 - sympy.cos(phi)) + b \* (1 - sympy.cos(theta)))

Pi2 = m\_2 \* g \* (l \* (1 - sympy.cos(phi)) + b)

Epot = Pi1 + Pi2

M = - k \* omega\_theta

L = Ekin - Epot

eq1 = sympy.diff(sympy.diff(L, omega\_theta), t) - sympy.diff(L, theta) - M

eq2 = sympy.diff(sympy.diff(L, omega\_phi), t) - sympy.diff(L, phi)

# вычисление вторых производных (DV/dt и domega/dt) с использованием метода Крамера

a11 = eq1.coeff(sympy.diff(omega\_theta, t), 1)

a12 = eq1.coeff(sympy.diff(omega\_phi, t), 1)

a21 = eq2.coeff(sympy.diff(omega\_theta, t), 1)

a22 = eq2.coeff(sympy.diff(omega\_phi, t), 1)

b1 = -(eq1.coeff(sympy.diff(omega\_theta, t), 0)).coeff(sympy.diff(omega\_phi, t), 0).subs([(sympy.diff(theta, t), omega\_theta), (sympy.diff(phi, t), omega\_phi)])

b2 = -(eq2.coeff(sympy.diff(omega\_theta, t), 0)).coeff(sympy.diff(omega\_phi, t), 0).subs([(sympy.diff(theta, t), omega\_theta), (sympy.diff(phi, t), omega\_phi)])

detA = a11\*a22 - a12\*a21

detA1 = b1\*a22 - b2\*a21

detA2 = a11\*b2 - b1\*a21

domega\_phidt = detA1/detA

domega\_thetadt = detA2/detA

iterations = 500

T = numpy.linspace(0, 10, iterations)

# Построение системы д/у

f\_omega\_phi = sympy.lambdify([phi, theta, omega\_theta, omega\_phi], domega\_phidt, "numpy")

f\_omega\_theta = sympy.lambdify([phi, theta, omega\_theta, omega\_phi], domega\_thetadt, "numpy")

y0 = [sympy.pi/2, sympy.pi/2, 15, 1]

sol = odeint(formY, y0, T, args=(f\_omega\_phi, f\_omega\_theta))

# generate coords of the line

X\_ab\_func = sympy.lambdify(phi, sympy.sin(phi) \* l)

Y\_ab\_func = sympy.lambdify(phi, -sympy.cos(phi) \* l)

X\_ab = X\_ab\_func(sol[:, 0])

Y\_ab = Y\_ab\_func(sol[:, 0])

# generate coords of the circle

X\_o\_func = sympy.lambdify(theta, sympy.sin(theta) \* b)

Y\_o\_func = sympy.lambdify(theta, -sympy.cos(theta) \* b)

X\_o = X\_o\_func(sol[:, 1])

Y\_o = Y\_o\_func(sol[:, 1])

# draw graphs

fig = plt.figure()

ax1 = fig.add\_subplot(2, 1, 1)

ax1.axis('equal')

ax1.set(xlim = [-3\*l, 3\*l], ylim = [-1.5\*l, 0.5])

ax2 = fig.add\_subplot(4, 1, 3)

ax2.plot(T, sol[:, 3])

ax2.set\_xlabel('T')

ax2.set\_ylabel('omega\_phi phi')

ax3 = fig.add\_subplot(4, 1, 4)

ax3.plot(T, sol[:, 2])

ax3.set\_xlabel('T')

ax3.set\_ylabel('omega\_phi theta')

# draw the system

beam1, = ax1.plot([-l, -l + X\_ab[0]], [0, Y\_ab[0]], 'black')

beam2, = ax1.plot([l, l + X\_ab[0]], [0, Y\_ab[0]], 'black')

beam3, = ax1.plot([-l + X\_ab[0], l + X\_ab[0]], [Y\_ab[0], Y\_ab[0]], 'black')

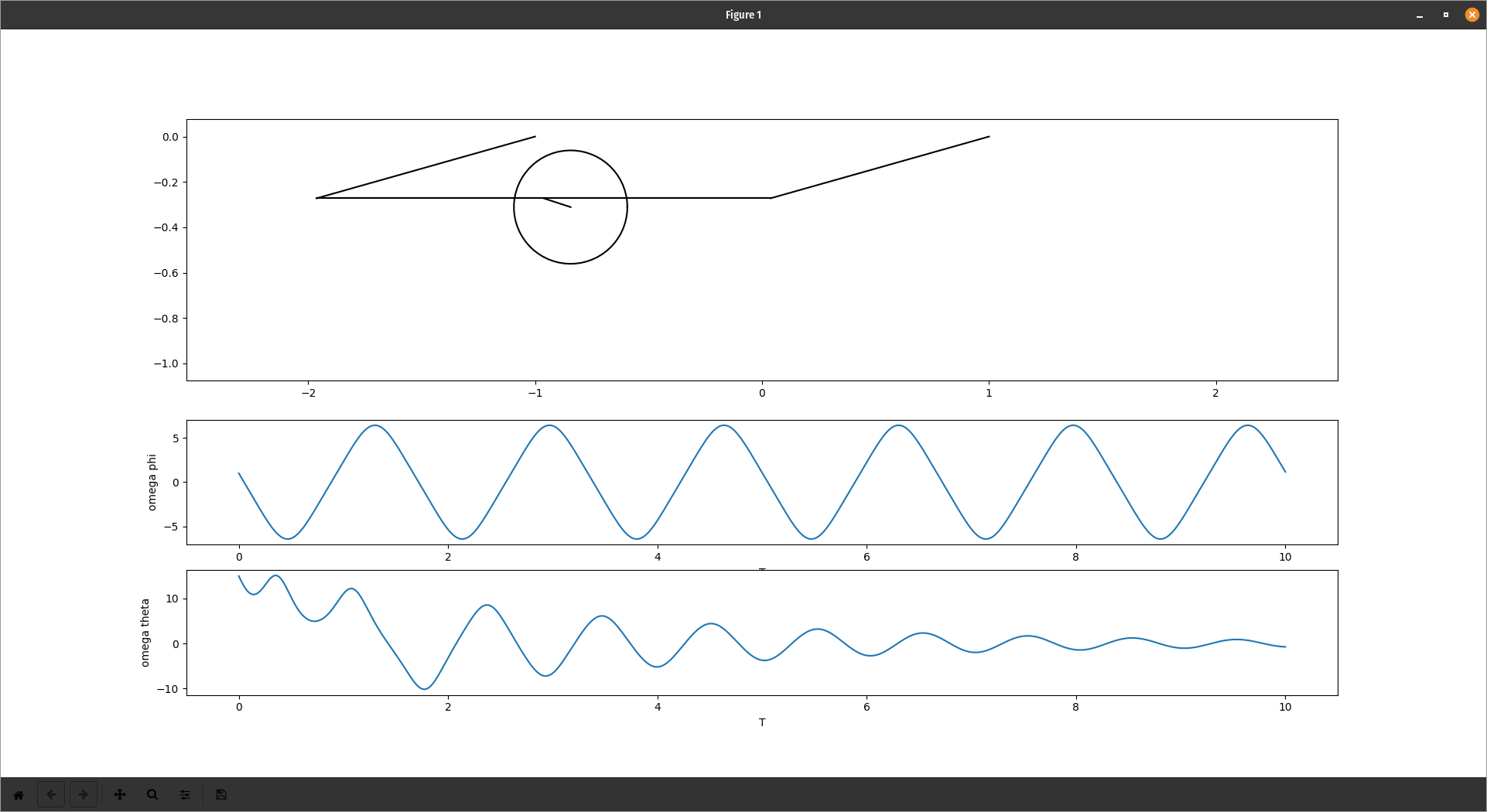
beam4, = ax1.plot([X\_ab[0], X\_ab[0] + X\_o[0]], [Y\_ab[0], Y\_ab[0] + Y\_o[0]], 'black')

circle, = ax1.plot(\*draw\_circle(X\_ab[0] + X\_o[0], Y\_ab[0] + Y\_o[0]), 'black')

anim = FuncAnimation(fig, animation, frames=iterations - 1, interval=0, blit=True)

plt.show()

*Скриншот:*

**