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 № 18**

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

Ибрагимов Роман Рифхатович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

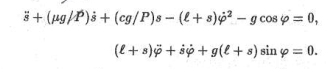
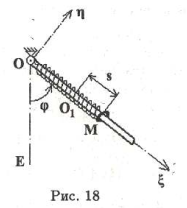
Авдюшкин А.Н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

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

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 программы

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from scipy.integrate import odeint

def SystDiffEq(y, t, P, l, c, mu, g):

dy = np.zeros\_like(y)

dy[0] = y[2]

dy[1] = y[3]

s = y[0]

phi = y[1]

ds = y[2]

dphi = y[3]

# a11 \* s'' + a12 \* phi'' = b1

# a21 \* s'' + a22 \* phi'' = b2

a11 = 1

a12 = 0

a21 = 0

a22 = (l + s)

b1 = g \* np.cos(phi) - ((mu \* g) / P) \* ds - ((c \* g) / P) \* s + (l + s) \* (dphi) \*\* 2

b2 = -ds \* dphi - g \* (l + s) \* np.sin(phi)

detA = a11 \* a22 - a12 \* a21

detA1 = b1 \* a22 - a12 \* b2

detA2 = a11 \* b2 - b1 \* a21

dy[2] = detA1 / detA

dy[3] = detA2 / detA

return dy

def spring(start, end, nodes, width):

nodes = max(int(nodes), 1)

start, end = np.array(start).reshape((2,)), np.array(end).reshape((2,))

if (start == end).all():

return start[0], start[1]

length = np.linalg.norm(np.subtract(end, start))

u\_t = np.subtract(end, start) / length

u\_n = np.array([[0, -1], [1, 0]]).dot(u\_t)

spring\_coords = np.zeros((2, nodes + 2))

spring\_coords[:,0], spring\_coords[:,-1] = start, end

normal\_dist = np.sqrt(max(0, width\*\*2 - (length\*\*2 / nodes\*\*2))) / 2

for i in range(1, nodes + 1):

spring\_coords[:,i] = (

start

+ ((length \* (2 \* i - 1) \* u\_t) / (2 \* nodes))

+ (normal\_dist \* (-1)\*\*i \* u\_n))

return spring\_coords[0,:], spring\_coords[1,:]

def anim(i):

pM.set\_data([M\_x[i], M\_y[i]])

OE.set\_data([O\_x, E\_x[i]], [O\_y, E\_y[i]])

Spr.set\_data(\*spring((O\_x, O\_y), (M\_x[i], M\_y[i]), 10, 0.3))

return

steps = 1000

t = np.linspace(0, 10, steps)

y0 = [0, np.pi / 10, 0, 0.3]

P = 10

l = 0.5

c = 20

g = 9.8

mu = 0

Y = odeint(SystDiffEq, y0, t, (P, l, c, mu, g))

s = Y[:, 0]

phi = Y[:, 1]

ds = Y[:, 2]

dphi = Y[:, 3]

Stt = np.zeros\_like(t)

Phitt = np.zeros\_like(t)

for i in range(len(t)):

Stt[i] = SystDiffEq(Y[i], t[i], P, l, c, mu, g)[2]

Phitt[i] = SystDiffEq(Y[i], t[i], P, l, c, mu, g)[3]

diff\_solve = plt.figure()

s\_plt = diff\_solve.add\_subplot(2, 1, 1)

s\_plt.plot(t, s)

phi\_plt = diff\_solve.add\_subplot(2, 1, 2)

phi\_plt.plot(t, phi)

x0 = 2

y0 = 3

L = 1.5

O\_x = x0

O\_y = y0

phi = phi - np.pi / 2

E\_x = x0 + L \* np.cos(phi)

E\_y = y0 + L \* np.sin(phi)

L\_sping\_max = l

L\_sping\_curr = L\_sping\_max \* np.cos(s)

M\_x = x0 + L\_sping\_curr \* np.cos(phi)

M\_y = y0 + L\_sping\_curr \* np.sin(phi)

fgr = plt.figure()

gr = fgr.add\_subplot(1, 1, 1)

gr.axis('equal')

gr.set(xlim=[0, 5], ylim=[-1, 5])

gr.plot([0, 0, 4], [4, 0, 0], linewidth=3)

gr.plot([1.75, 2.25], [3, 3], 'black', linewidth=3)

pO = gr.plot(O\_x, O\_y)[0]

pE = gr.plot(E\_x[0], E\_y[0])[0]

pM = gr.plot(M\_x[0], M\_y[0], 'yellow', marker='o')[0]

OE = gr.plot([O\_x, E\_x[0]], [O\_y, E\_y[0]], 'grey')[0]

Spr = gr.plot(\*spring((O\_x, O\_y), (M\_x[0], M\_y[0]), 10, 0.3), 'red')[0]

an = FuncAnimation(fgr, anim, frames=steps, interval=1)

plt.show()

Результат выполнения программы

