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-212Б-22

Козырев П.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Зав. каф. 802, Бардин Б.С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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.

**Задание системы формулируется следующим образом:**

Колечко M веса P прикреплено к концу пружины OM и может скользить вдоль стержня, который качается в вертикальной плоскости вокруг шарнира О. Жесткость пружины с. Длина недеформированной пружины ОО1 = l.

**Дифференциальные уравнения движения системы:**

**Рисунок получившейся анимации движения**

*Изображение выглядит как снимок экрана, линия, График, текст

Автоматически созданное описание*

*Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описание*

**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, P, l, c, g, nu):

dy = np.zeros(4)

dy[0] = y[2]

dy[1] = y[3]

a11 = 1

a12 = 0

a21 = 0

a22 = (l + y[0])

b1 = -(nu \* g / P) \* y[2] - (c \* g / P) \* y[0] + (l + y[0]) \* y[3] \*\* 2 + g \* np.cos(y[1])

b2 = -(y[2] \* y[3]) - g \* (l + y[0]) \* 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

Steps = 1001

t\_fin = 20

t = np.linspace(0, t\_fin, Steps)

l\_kernel = 20 # длина стержня

s\_0 = 8 # длина O-О1

P\_1 = 0.1 # вес стержня

P = 0.1 # вес колечка

l = 0.5 # длина недеформированной пружины

c = 200 # жесткость пружины

g = 9.81 # ускорение свободного падения

nu = 1 # трение

phi\_0 = math.pi/10 # начальный угол

S\_0 = 0 # начальная длина пружины

dphi\_0 = 0

dS\_0 = 0

y0 = [S\_0, phi\_0, dS\_0, dphi\_0]

Y = odeint(odesys, y0, t, (P, l, c, g, nu))

s = Y[:, 0]

phi = Y[:, 1]

ds = Y[:, 2]

dphi = Y[:, 3]

dds = np.array([odesys(y, t, P, l, c, g, nu)[2] for y, t in zip(Y, t)])

ddphi = np.array([odesys(y, t, P, l, c, g, nu)[3] for y, t in zip(Y, t)])

N = P \* np.sin(phi) + (P / g) \* ((l + s) \* ddphi + 2 \* dphi \* ds)

M\_x = s\_0 \* np.sin(phi) \* s + np.sin(phi)\*s\_0

M\_y = (-1) \* s\_0 \* np.cos(phi) \* s - s\_0\*np.cos(phi)

Kernel\_x = l\_kernel \* np.sin(phi)

Kernel\_y = (-1) \* l\_kernel \* np.cos(phi)

SprX\_0 = 4

K = 19

Sh = 0.3

b = 1 / (K - 2)

X\_Spr = np.zeros(K)

Y\_Spr = np.zeros(K)

X\_Spr[0] = 0

Y\_Spr[0] = 0

X\_Spr[K - 1] = 0

Y\_Spr[K - 1] = 1

for i in range(K - 2):

Y\_Spr[i + 1] = b \* ((i + 1) - 1 / 2)

X\_Spr[i + 1] = Sh \* (-1) \*\* i

L\_Spr = (s\_0 \* np.cos(phi) \* s \* (-1)) - s\_0 \* np.cos(phi)

X\_U = np.linspace(0, 1, K)

fig\_for\_graphs = plt.figure(figsize=[13, 7])

ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2, 2, 1)

ax\_for\_graphs.plot(t, s, color='Blue')

ax\_for\_graphs.set\_title('s(t)')

ax\_for\_graphs.set(xlim=[0, t\_fin])

ax\_for\_graphs.grid(True)

ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2, 2, 3)

ax\_for\_graphs.plot(t, phi, color='Red')

ax\_for\_graphs.set\_title('phi(t)')

ax\_for\_graphs.set(xlim=[0, t\_fin])

ax\_for\_graphs.grid(True)

ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2, 2, 2)

ax\_for\_graphs.plot(t, N, color='Orange')

ax\_for\_graphs.set\_title('N')

ax\_for\_graphs.set(xlim=[0, t\_fin])

ax\_for\_graphs.grid(True)

fig = plt.figure(figsize=[12, 7])

ax = fig.add\_subplot(1, 1, 1)

ax.axis('equal')

ax.set(xlim=[-20, 20], ylim=[-30, 10])

Kernel = ax.plot([0, Kernel\_x[0]], [0, Kernel\_y[0]], linewidth=5, color="#7ED7C1")[0]

ax.plot([0, 0], [-20, 20], "k--")

ax.plot([-50, 50], [0, 0], "k--")

ax.plot(0, 0, marker='o')

ax.annotate("O", xy=(0, 0), xytext=(-1, -1.5))

ax.plot([-1, 0, 1], [1, 0, 1], "black")

ax.plot([-1, 1], [1, 1], "black")

Point\_M = ax.plot(M\_x[0], M\_y[0], marker='o', color='r')[0]

Drawed\_Spring = ax.plot(X\_Spr + X\_U \* np.sin(phi)[0] \* (s\_0 \* s[0])+np.sin(phi)[0]\*s\_0\*X\_U, Y\_Spr \* L\_Spr[0])[0]

def anima(i):

if s[i] < 0: tmp = -1

else: tmp = 1

Point\_M.set\_data([M\_x[i] \* tmp], [M\_y[i]\*tmp])

Kernel.set\_data([0, Kernel\_x[i]], [0, Kernel\_y[i]])

Drawed\_Spring.set\_data((X\_Spr + X\_U \* np.sin(phi)[i]\*(s\_0 \* s[i])+np.sin(phi)[i]\*s\_0\*X\_U)\*tmp, Y\_Spr \* L\_Spr[i]\*tmp)

return [Point\_M, Kernel, Drawed\_Spring]

anim = FuncAnimation(fig, anima, frames=len(t), interval=40, repeat=False)

plt.show()

**Примеры работы программы:**

1. Начальные условия из учебника

P = 10H, l = 0.5м, с = 20 Н/м,

Изображение выглядит как текст, линия, диаграмма, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, линия, График, диаграмма

Автоматически созданное описание

Стержень движется с небольшим ускорением, из-за небольшого угла отклонения. Вследствие чего колечко скользит медленно. Стержень и шарик совершают колебательное движение.

Изображение выглядит как текст, линия, диаграмма, График

Автоматически созданное описаниеИзображение выглядит как текст, диаграмма, линия, График

Автоматически созданное описание

Стержень не отклонен и статически висит, не совершая движение. Колечко совершает колебательное движение за счет силы тяжести.

Изображение выглядит как линия, диаграмма, График, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, линия, Шрифт, График

Автоматически созданное описание

4)

Изображение выглядит как линия, диаграмма, График, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, линия, снимок экрана, График

Автоматически созданное описание

*Вывод:* Я проинтегрировал систему дифференциальных уравнений движения системы с двумя степенями свободы с помощью средств Python. Построил анимацию движения системы, а также графики законов движения системы и указанных в задании реакций для разных случаев системы.