ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«ДИНАМИКА СИСТЕМЫ»**

**ПО ДИСЦИПЛИНЕ «ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ ДИНАМИЧЕСКИХ СИСТЕМ»**

**ВАРИАНТ ЗАДАНИЯ №20**

Выполнил(а) студент группы М8О-201Б-23

Терентьев М.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Ст. преп. каф. 802 Волков Е.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

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

Москва, 2024

**Вариант №20**

**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 20 варианта формулируется следующим образом:**

Однородная балка DE длины 2а и массы m1 закреплена в не-

подвижном шарнире D и концом Е соединена с пружиной жесткость-

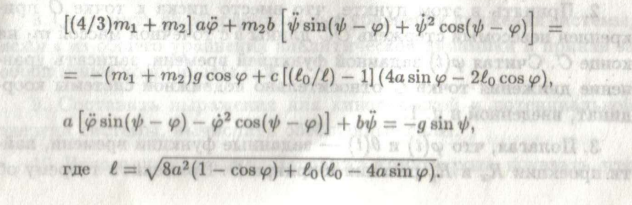
ти с (рис. 20). К середине балки прикреплен невесомый стержень

AD длины и с точечным грузом массы m2 на конце В. Длина неде-

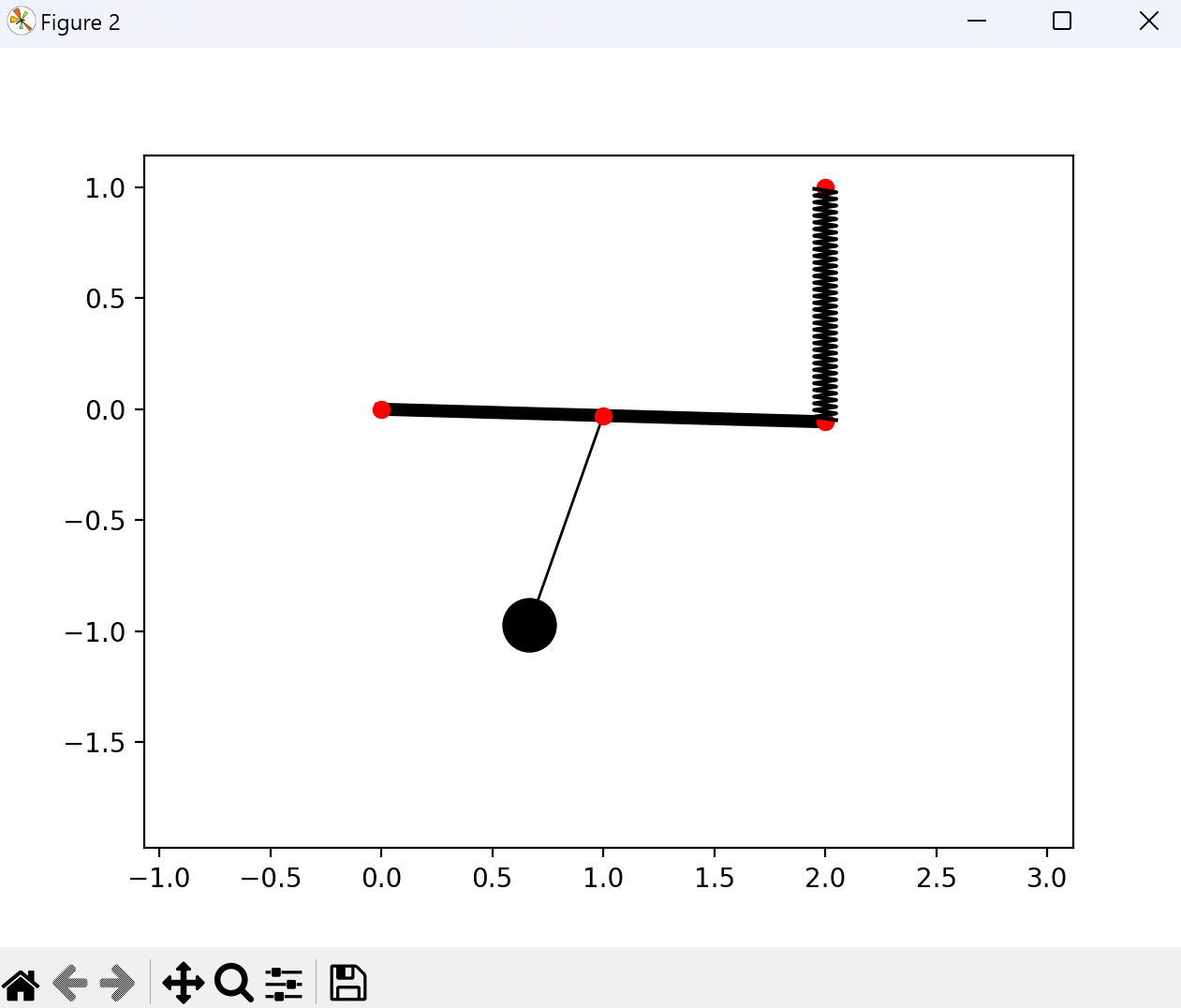
формированной пружины равна l0; при горизонтальном положении

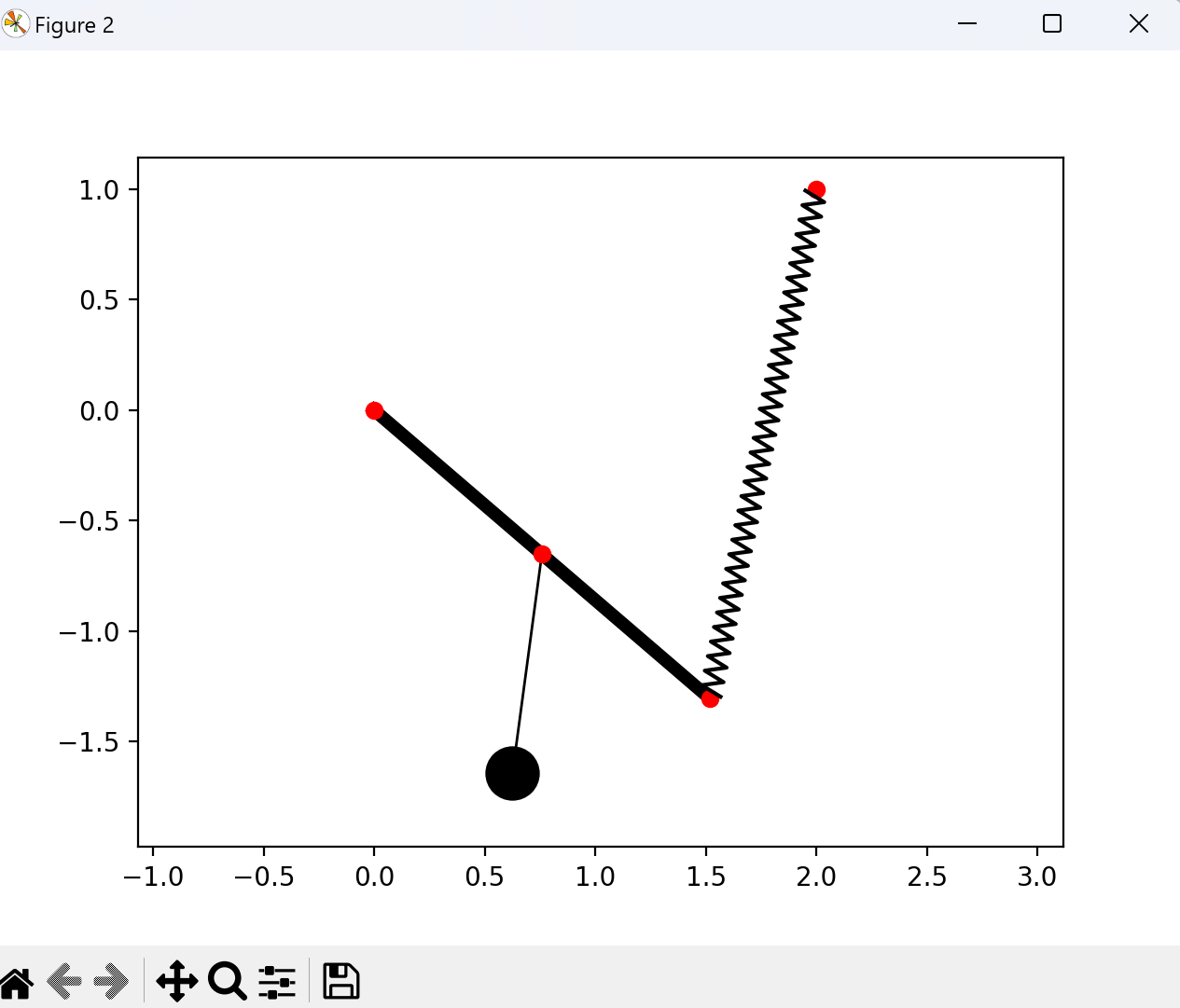
балки DE пружина вертикальная.

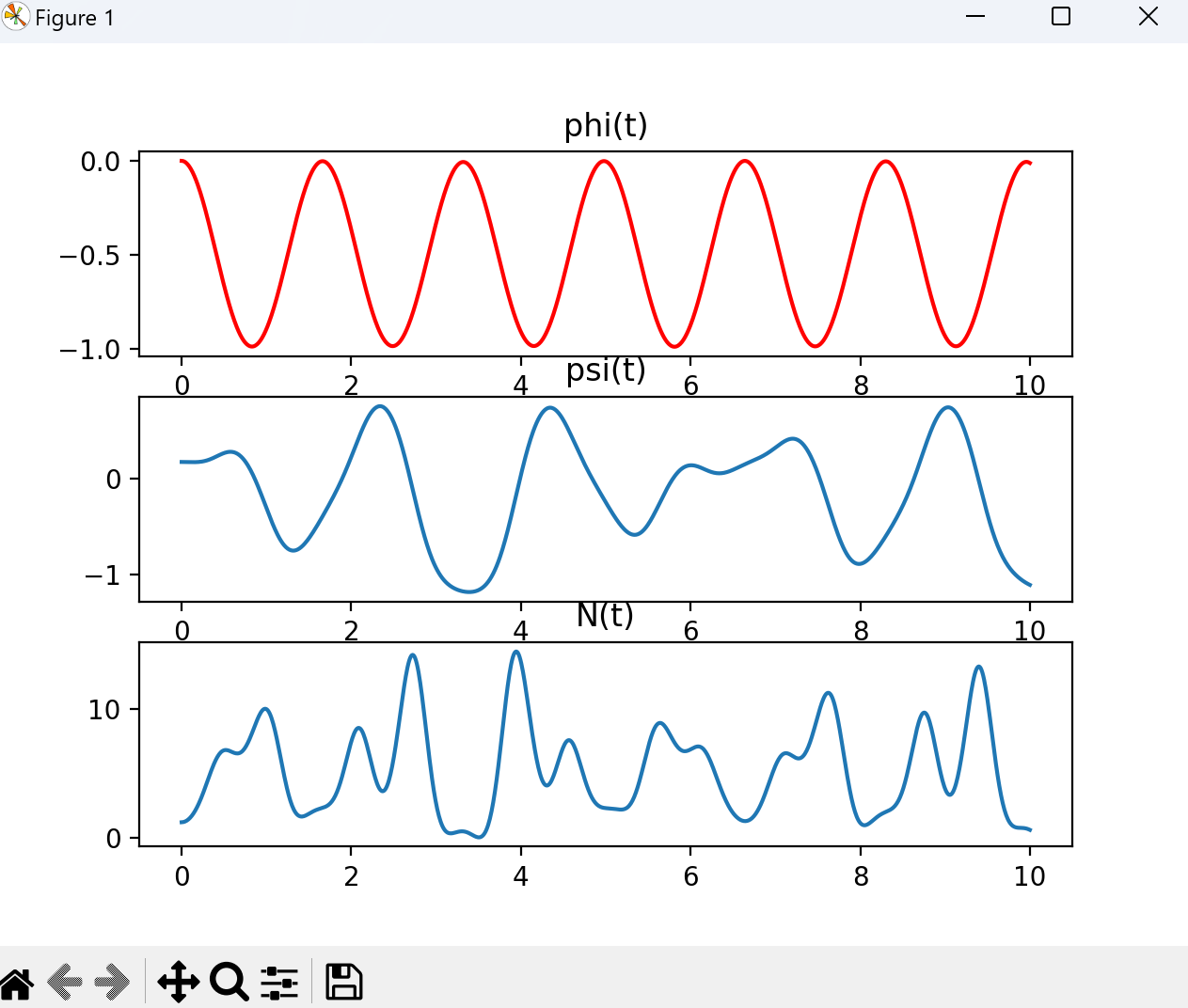
Дифференциальные уравнения движения системы имеют вид:



**Рисунок получившейся анимации движения:**







**Код программы:**

import matplotlib.pyplot as plt

import numpy as np

from matplotlib.animation import FuncAnimation

from scipy.integrate import odeint

def SystDiffEq(y, t, m1, m2, a, b, l0, c, g):

# y = [phi, psi, phi', psi'] -> dy = [phi', psi', phi'', psi'']

dy = np.zeros\_like(y)

dy[0] = y[2]

dy[1] = y[3]

phi = y[0]

psi = y[1]

dphi = y[2]

dpsi = y[3]

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

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

l = np.sqrt(8 \* a \*\* 2 \* (1 - np.cos(phi)) +

l0 \* (l0 - 4 \* a \* np.sin(phi)))

a11 = ((4/3) \* m1 + m2) \* a

a12 = m2 \* np.sin(psi - phi)

b1 = (-(m1 + m2) \* g \* np.cos(phi)

+ c \* ((l0 / l) - 1) \* (4 \* a \* np.sin(phi) - 2 \* l0 \* np.cos(phi))

- m2 \* b \* dpsi \*\* 2 \* np.cos(psi - phi))

a21 = a \* np.sin(psi - phi)

a22 = b

b2 = - g \* np.sin(psi) + a \* dphi \*\* 2 \* np.cos(psi - 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

# Дано:

a = b = l0 = 1

DE = 2 \* a

g = 9.8

m1 = 50

m2 = 0.5

a = b = l0 = 1

c = 250

t0 = 0

phi0 = 0

psi0 = np.pi / 18

dphi0 = 0

dpsi0 = 0

# Задаю функции phi(t) и psi(t)

step = 1000

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

y0 = np.array([phi0, psi0, dphi0, dpsi0])

Y = odeint(SystDiffEq, y0, t, (m1, m2, a, b, l0, c, g))

phi = Y[:, 0]

psi = Y[:, 1]

dphi = Y[:, 2]

dpsi = Y[:, 3]

ddphi = np.zeros\_like(t)

for i in np.arange(len(t)):

ddphi[i] = SystDiffEq(Y[i], t[i], m1, m2, a, b, l0, c, g)[2]

N = m2 \* (g \* np.cos(psi)

+ b \* dpsi \*\* 2

+ a \* (ddphi \* np.cos(psi - phi)

+ dphi \*\* 2 \* np.sin(psi - phi)))

fgrt = plt.figure()

phiplt = fgrt.add\_subplot(3, 1, 1)

plt.title("phi(t)")

phiplt.plot(t, phi, color='r')

psiplt = fgrt.add\_subplot(3, 1, 2)

plt.title("psi(t)")

psiplt.plot(t, psi)

nplt = fgrt.add\_subplot(3, 1, 3)

plt.title("N(t)")

nplt.plot(t, N)

fgrt.show()

fig = plt.figure()

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

gr.axis('equal')

# Балка DE

Xd = 0

Yd = 0

Xe = Xd + DE \* np.cos(phi)

Ye = Yd + DE \* np.sin(phi)

balkaDE = gr.plot([Xd, Xe[0]], [Yd, Ye[0]], color='black', linewidth=5)[0]

pD = gr.plot(Xd, Yd, marker='o', color='r')[0]

pE = gr.plot(Xe, Ye, marker='o', color='r')[0]

# Пружина

Xc = DE

Yc = l0

pC = gr.plot(Xc, Yc, marker='o', color='r')[0]

def get\_spring(coils, width, start, end):

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

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

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

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

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

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

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

for i in np.arange(1, coils + 1):

spring\_coords[:, -i] = (start

+ ((len \* (2 \* i - 1) \* u\_t) / (2 \* coils))

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

return spring\_coords[0, 2:], spring\_coords[1, 2:]

pS = gr.plot(\*get\_spring(70, 0.1, [Xe[0], Ye[0]], [Xc, Yc]), color='black')[0]

# Стержень AB

Xa = Xd + DE / 2 \* np.cos(phi)

Ya = Yd + DE / 2 \* np.sin(phi)

Xb = Xa + b \* np.cos(psi - np.pi / 2)

Yb = Ya + b \* np.sin(psi - np.pi / 2)

sterjenAB = gr.plot([Xa[0], Xb[0]], [Ya[0], Yb[0]],

color='black', linewidth=1)[0]

pA = gr.plot(Xa, Ya, marker='o', color='r')[0]

pB = gr.plot(Xb, Yb, marker='o', color='black', markersize=20)[0]

def run(i):

balkaDE.set\_data([Xd, Xe[i]], [Yd, Ye[i]])

pE.set\_data(Xe[i], Ye[i])

pS.set\_data(\*get\_spring(70, 0.1, [Xe[i], Ye[i]], [Xc, Yc]))

pA.set\_data(Xa[i], Ya[i])

pB.set\_data(Xb[i], Yb[i])

sterjenAB.set\_data([Xa[i], Xb[i]], [Ya[i], Yb[i]])

anim = FuncAnimation(fig, run, frames=step, interval=1)

plt.show()

**Вывод:**

В ходе выполнения лабораторной работы, с использованием языка программирования Python и библиотек matplotlib и numpy было схематично проанимировано движение двух стержней и пружины, а также решена система дифференциальных уравнений.

В ходе выполнения работы приобретены навыки работы с 2D-анимацией в matplotlib, что послужило основой для последующих лабораторных работ. Программа основывается на реальных законах движения, что позволяет визуализировать поведение системы в реальной жизни.

Начало формы