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 № 31**

Выполнил(а) студент группы М8О-206Б-22

Постнов Л. А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

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

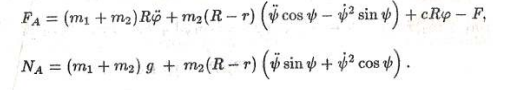
подпись, дата

Москва, 2023

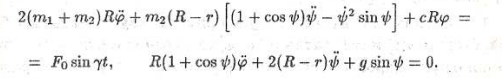
*Задание:* проинтегрировать систему дифференциальных уравнений движения системы с двумя степенями свободы с помощью средств Python. Построить графики законов движения системы и указанных в задании реакций для разных случаев системы.

*Вариант:*

Силы трения и давления на плоскость:



Дифференциальные уравнения движения системы:



*Код программы:*

import numpy as n

import matplotlib.pyplot as plt

from scipy.integrate import odeint

from matplotlib.animation import FuncAnimation

def sys\_diff\_eq(y, t, m1, m2, R, r, c, F, gamma, g):

# y = [phi, psi, phi', psi', t] -> dy = [phi', psi', phi'', psi'', t]

dy = n.zeros\_like(y)

dy[0] = y[2]

dy[1] = y[3]

dy[4] = y[4] + 3 \* n.pi / 1000

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

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

a11 = 2 \* (m1 + m2) \* R

a12 = m2 \* (R - r) \* (1 + n.cos(y[1]))

b1 = F \* n.sin(gamma \* y[4]) + m2 \* (R - r) \* y[3] \*\* 2 \* n.sin(y[1]) - c \* R \* y[0]

a21 = R \* (1 + n.cos(y[1]))

a22 = 2 \* (R - r)

b2 = -g \* n.sin(y[1])

det\_a = a11 \* a22 - a12 \* a21

det\_a1 = b1 \* a22 - a12 \* b2

det\_a2 = a11 \* b2 - a21 \* b1

dy[2] = det\_a1 / det\_a

dy[3] = det\_a2 / det\_a

return dy

m1 = 5

m2 = 1

R = 1

r = 0.1

c = 10

F = 1

gamma = n.pi / c

g = 9.81

y0 = [0, n.pi / 6, 0, 0, 0]

x0 = 2

L = 0.9

step = 1000

t = n.linspace(0, 3 \* n.pi, step)

Y = odeint(sys\_diff\_eq, y0, t, (m1, m2, R, r, c, F, gamma, g))

phi = Y[:, 0]

psi = Y[:, 1]

phi\_t = Y[:, 2]

psi\_t = Y[:, 3]

x = n.zeros\_like(t)

phi\_tt = n.zeros\_like(t)

psi\_tt = n.zeros\_like(t)

F\_a = n.zeros\_like(t)

N\_a = n.zeros\_like(t)

for i in range(len(t)):

x[i] = phi[i] \* R

phi\_tt[i] = sys\_diff\_eq(Y[i], t[i], m1, m2, R, r, c, F, gamma, g)[2]

psi\_tt[i] = sys\_diff\_eq(Y[i], t[i], m1, m2, R, r, c, F, gamma, g)[3]

F\_a[i] = (m1 + m2) \* R \* phi\_tt[i] + m2 \* (R - r) \* (psi\_tt[i] \* n.cos(psi[i])) \

- psi\_t[i]\*\*2 \* n.sin(psi[i]) + c \* R \* phi[i] - F

N\_a[i] = (m1 + m2) \* g + m2 \* (R - r) \* (psi\_tt[i] \* n.sin(psi[i]) + psi\_t[i]\*\*2 \* n.cos(psi[i]))

fgr = plt.figure()

gr = fgr.add\_subplot(4, 2, (1, 7))

gr.axis('equal')

phi\_plt = fgr.add\_subplot(4, 2, 2)

phi\_plt.plot(t, phi)

phi\_plt.set\_title("φ(t)")

psi\_plt = fgr.add\_subplot(4, 2, 4)

psi\_plt.plot(t, psi)

psi\_plt.set\_title("ψ(t)")

F\_a\_plt = fgr.add\_subplot(4, 2, 6)

F\_a\_plt.plot(t, F\_a)

F\_a\_plt.set\_title("F(t)")

N\_a\_plt = fgr.add\_subplot(4, 2, 8)

N\_a\_plt.plot(t, N\_a)

N\_a\_plt.set\_title("N(t)")

gr.plot([0, 0, 4], [2, 0, 0], linewidth=3)

Xa = x0 + x

Ya = R

Xb = Xa + L \* n.sin(psi)

Yb = Ya - L \* n.cos(psi)

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

Alp = n.linspace(0, 2 \* n.pi, 100)

Xc = n.cos(Alp)

Yc = n.sin(Alp)

Main\_cylinder = gr.plot(Xc \* R + Xa[0], Yc \* R + Ya)[0]

Sub\_cylinder = gr.plot(Xc \* r + Xb[0], Yc \* r + Yb[0])[0]

Np = 20

Xp = n.linspace(0, 1, 2 \* Np + 1)

Yp = 0.06 \* n.sin(n.pi / 2 \* n.arange(2 \* Np + 1))

Spring = gr.plot((x0 + x[0]) \* Xp, Yp + R)[0]

def run(i):

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

Main\_cylinder.set\_data([Xc \* R + Xa[i]], [Yc \* R + Ya])

Sub\_cylinder.set\_data([Xc \* r + Xb[i]], [Yc \* r + Yb[i]])

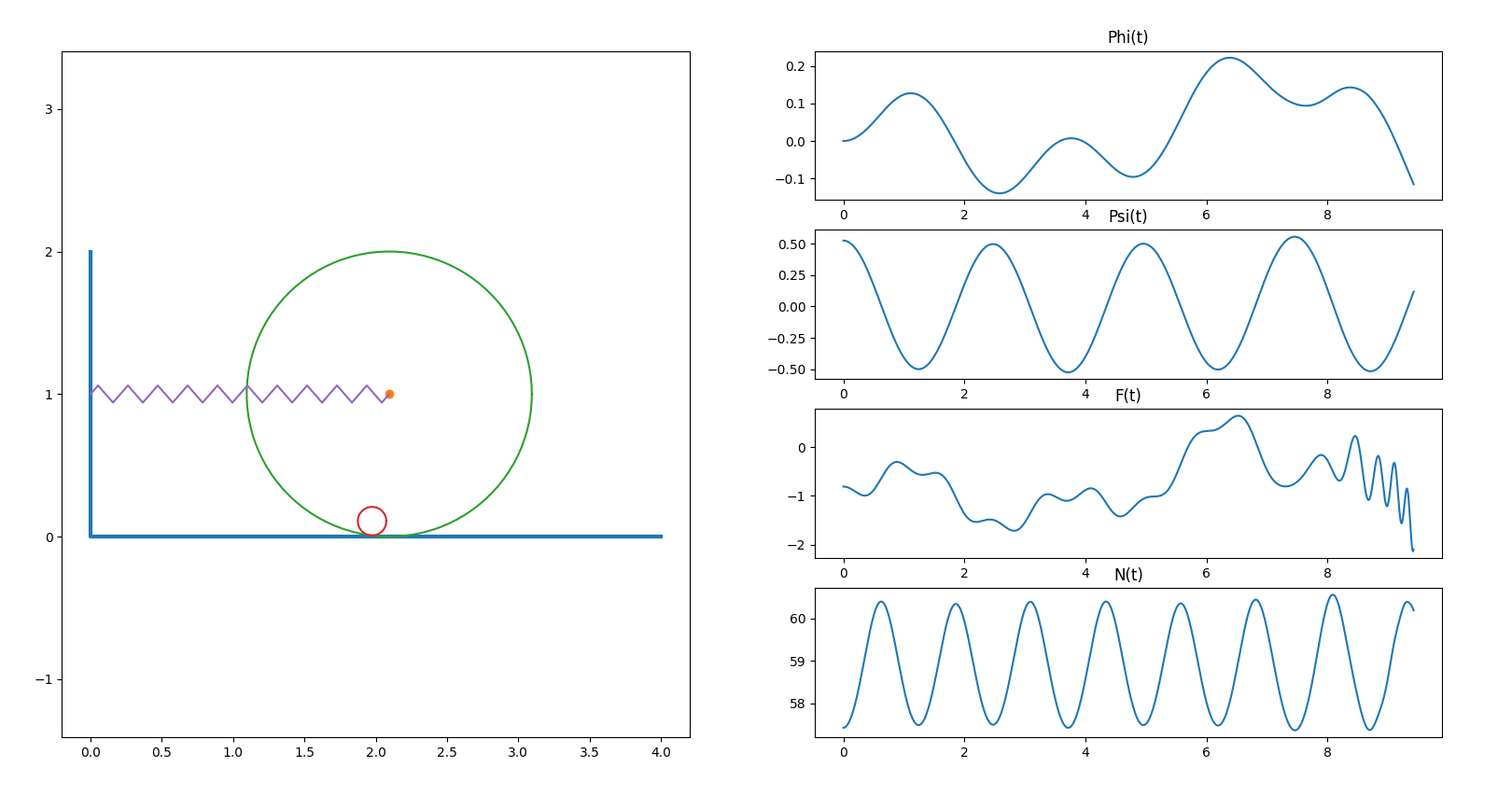
Spring.set\_data([(x0 + x[i]) \* Xp], [Yp + R])

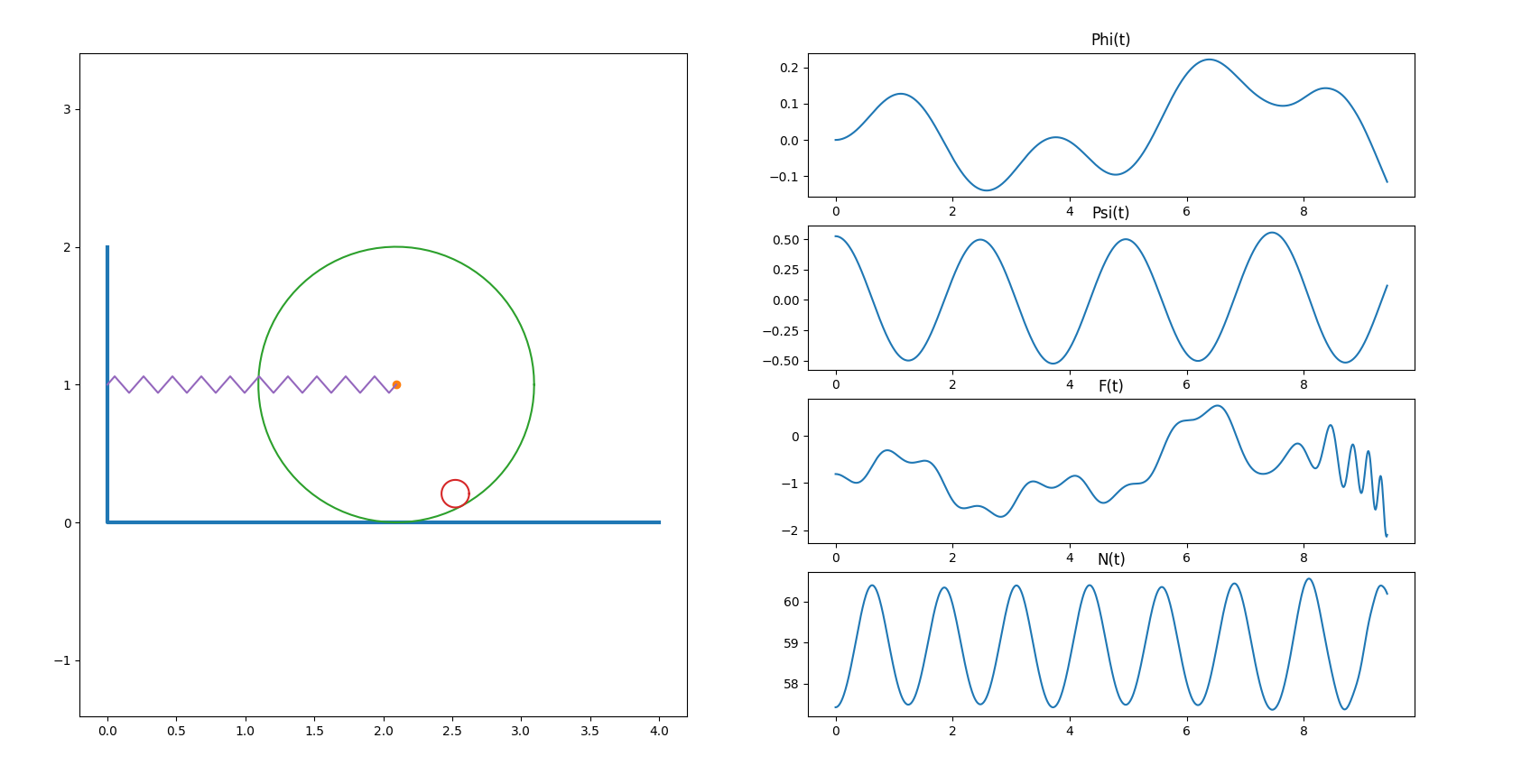
return [pA, Main\_cylinder, Sub\_cylinder, Spring]

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

plt.show()

*Результат работы программы:*





*Вывод:* в результате проделанной работы, я научился интегрировать систему дифференциальных уравнений движения системы с двумя степенями свободы с помощью средств Python и строить графики законов движения системы и указанных в задании реакций для разных случаев системы.