ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ СИСТЕМЫ»**

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

**ВАРИАНТ ЗАДАНИЯ №23**

Выполнил(а) студент группы М8О-207Б-22

Лебедько Платон Владимирович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

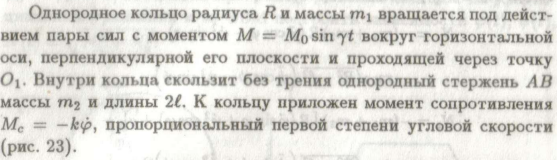
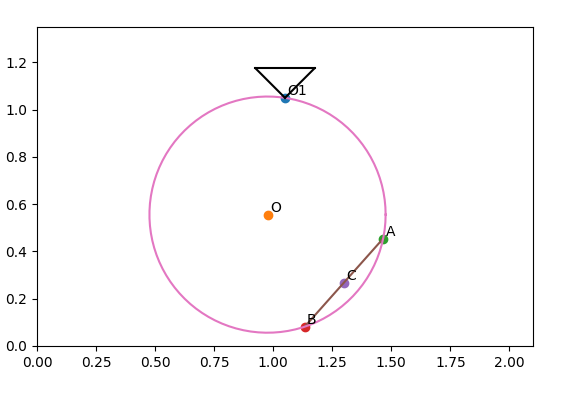
Зав. каф. 802, Бардин Б.С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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.

1. *Условия задачи 23 варианта:*  
   
2. *Рисунок получившейся физической модели:  
   *
3. *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, R, l, m1, m2, g, M0, gamma, k):

dy = np.zeros(4)

dy[0] = y[2]

dy[1] = y[3]

sqrt\_ = math.sqrt(R\*\*2 - l\*\*2)

sin\_ = np.sin(y[1] - y[0])

cos\_ = np.cos(y[1] - y[0])

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

a12 = m2\*R\*sqrt\_\*cos\_

a21 = sqrt\_\*R\*cos\_

a22 = R\*\*2 - 2/3\*l\*\*2

b1 = m2\*R\*sqrt\_\*y[3]\*\*2\*sin\_ - (m1 + m2)\*g\*R\*np.sin(y[0]) + M0\*np.sin(gamma\*t) - k\*y[2]

b2 = -sqrt\_\*(R\*y[2]\*\*2\*sin\_ + g\*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

# Константы

R = 0.5

l = 0.25

m1 = 2

m2 = 1

g = 9.81

M0 = 15

gamma = 3/2\*math.pi

k = 10

# Вспомогательные

forLetters = 0.01

TriangleLength = R \* 0.5

# Кадры

steps = 1001

t\_fin = 20

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

# Начальные условия

phi0 = 0

psi0 = math.pi / 6

dphi0 = 0

dpsi0 = 0

y0 = [phi0, psi0, dphi0, dpsi0]

# Проинтегрированная система

Y = odeint(odesys, y0, t, (R, l, m1, m2, g, M0, gamma, k))

# Функции

phi = Y[:,0]

psi = Y[:,1]

dphi = Y[:,2]

dpsi = Y[:,3]

ddphi = [odesys(y, t, R, l, m1, m2, g, M0, gamma, k)[2] for y,t in zip(Y,t)]

ddpsi = [odesys(y, t, R, l, m1, m2, g, M0, gamma, k)[3] for y,t in zip(Y,t)]

NO1X = -(m1 + m2)\*R\*(ddphi\*np.cos(phi) - dphi\*\*2\*np.sin(phi)) - m2\*math.sqrt(R\*\*2 - l\*\*2)\*(ddpsi\*np.cos(psi) - dpsi\*\*2\*np.sin(psi))

NO1Y = -(m1 + m2)\*R\*((ddphi\*np.sin(phi) + dphi\*\*2\*np.cos(phi)) + g) - m2\*math.sqrt(R\*\*2 - l\*\*2)\*(ddpsi\*np.sin(psi) + dpsi\*\*2\*np.cos(psi))

NO1 = np.sqrt(NO1X\*\*2 + NO1Y\*\*2)

# Вычисленные константы

alpha = -math.acos(l / R)

OC = -R \* math.sin(alpha)

# Задаем положение

O1X = 2 \* R + R / 10

O1Y = 2 \* R + R / 10

# относительно т. О1

OX = O1X + R \* np.sin(phi)

OY = O1Y - R \* np.cos(phi)

# относительно т. О

CX = OX + OC \* np.sin(psi)

CY = OY - OC \* np.cos(psi)

AX = OX + R \* np.cos(alpha + psi)

AY = OY + R \* np.sin(alpha + psi)

BX = OX + R \* np.cos(math.pi - alpha + psi)

BY = OY + R \* np.sin(math.pi - alpha + psi)

fig = plt.figure()

ax1 = fig.add\_subplot()

ax1.axis('equal')

plt.gca().set\_adjustable("box")

ax1.set(xlim=[0, 4 \* R + 2 \* R / 10], ylim=[0, 2 \* R + TriangleLength + 2 \* R / 10])

ax1.plot(O1X, O1Y, marker = 'o')

plt.text(O1X + forLetters, O1Y + forLetters, 'O1')

OPoint = ax1.plot(OX[0], OY[0], marker = 'o')[0]

APoint = ax1.plot(AX[0], AY[0], marker = 'o')[0]

BPoint = ax1.plot(BX[0], BY[0], marker = 'o')[0]

CPoint = ax1.plot(CX[0], CY[0], marker = 'o')[0]

OText = plt.text(OX[0] + forLetters, OY[0] + forLetters, 'O')

AText = plt.text(AX[0] + forLetters, AY[0] + forLetters, 'A')

BText = plt.text(BX[0] + forLetters, BY[0] + forLetters, 'B')

CText = plt.text(CX[0] + forLetters, CY[0] + forLetters, 'C')

ABLine = ax1.plot([ AX[0], BX[0] ], [ AY[0], BY[0] ])[0]

phiForCirc = np.linspace(0, 2\*math.pi, 100)

Circ = ax1.plot(OX[0] + R \* np.cos(phiForCirc), OY[0] + R \* np.sin(phiForCirc))[0]

ax1.plot([O1X - TriangleLength / 2, O1X + TriangleLength / 2], [O1Y + TriangleLength / 2, O1Y + TriangleLength / 2], '000')

ax1.plot([O1X - TriangleLength / 2, O1X], [O1Y + TriangleLength / 2, O1Y], '000')

ax1.plot([O1X, O1X + TriangleLength / 2], [O1Y, O1Y + TriangleLength / 2], '000')

def anima(i):

OPoint.set\_data([OX[i]], [OY[i]])

APoint.set\_data([AX[i]], [AY[i]])

BPoint.set\_data([BX[i]], [BY[i]])

CPoint.set\_data([CX[i]], [CY[i]])

OText.set\_position([OX[i] + forLetters, OY[i] + forLetters])

AText.set\_position([AX[i] + forLetters, AY[i] + forLetters])

BText.set\_position([BX[i] + forLetters, BY[i] + forLetters])

CText.set\_position([CX[i] + forLetters, CY[i] + forLetters])

ABLine.set\_data([ AX[i], BX[i] ], [ AY[i], BY[i] ])

Circ.set\_data(OX[i] + R \* np.cos(phiForCirc), OY[i] + R \* np.sin(phiForCirc))

anim = FuncAnimation(fig, anima, frames = steps, interval = 50)

anim\_running = True

def onClick(event):

global anim\_running

if anim\_running:

anim.event\_source.stop()

anim\_running = False

else:

anim.event\_source.start()

anim\_running = True

fig.canvas.mpl\_connect('button\_press\_event', onClick)

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

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

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

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, 3)

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

ax\_for\_graphs.set\_title("psi(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, NO1, color='Black')

ax\_for\_graphs.set\_title("NO1(t)")

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

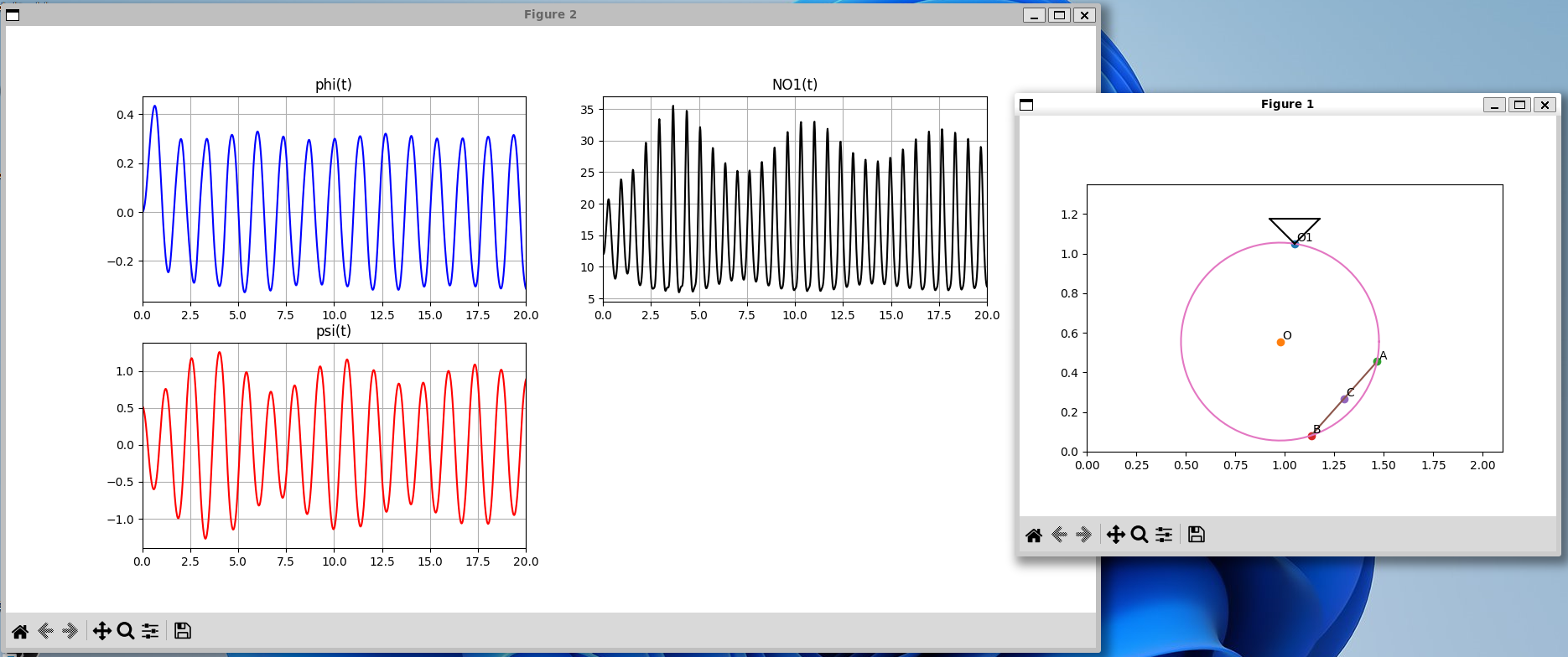
ax\_for\_graphs.grid(True)

plt.show()

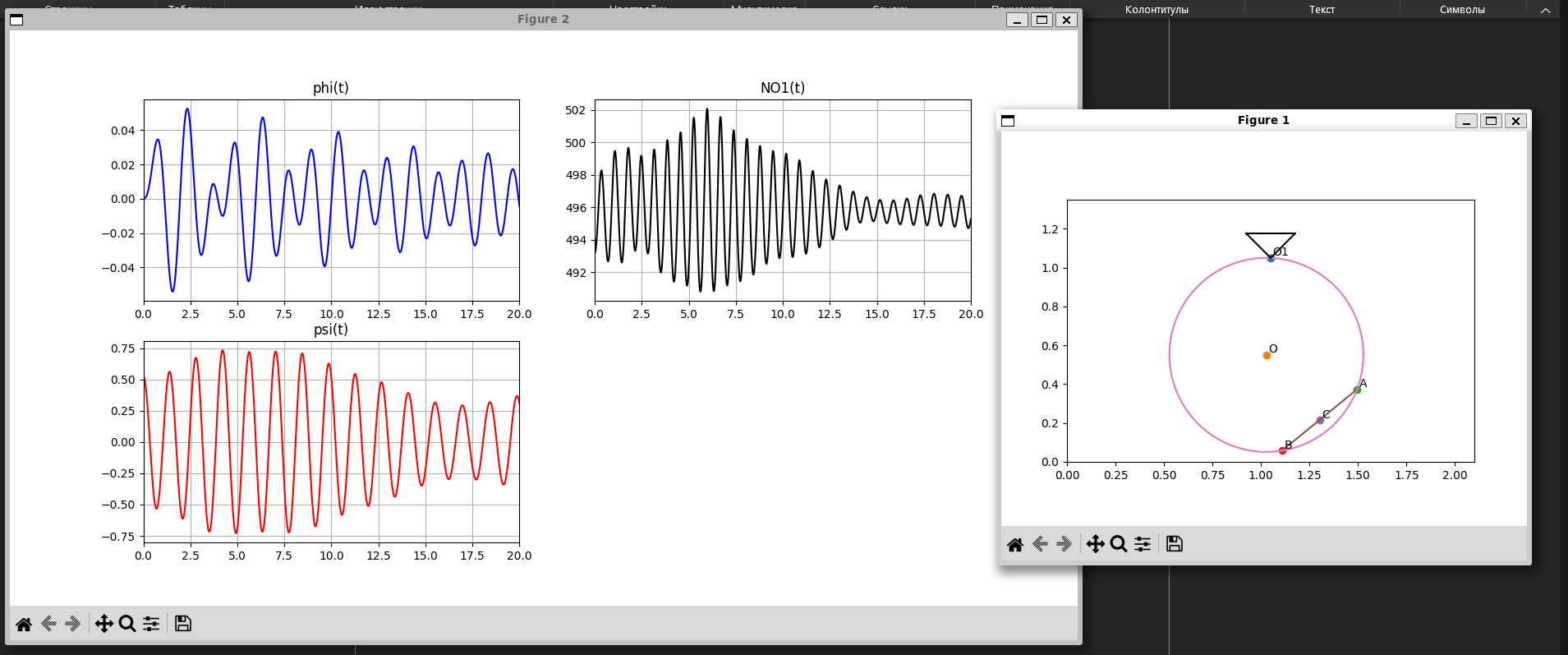
1. *Вариация констант и начальных условий:*

Константы: R = 0.5, l = 0.25, m1 = 2, m2 = 1, g = 9.81, M0 = 15, γ = 3/2\*π, k = 10

Начальные условия: φ0 = 0, ψ0 = π/6, dφ0/dt = 0, dψ0/dt = 0

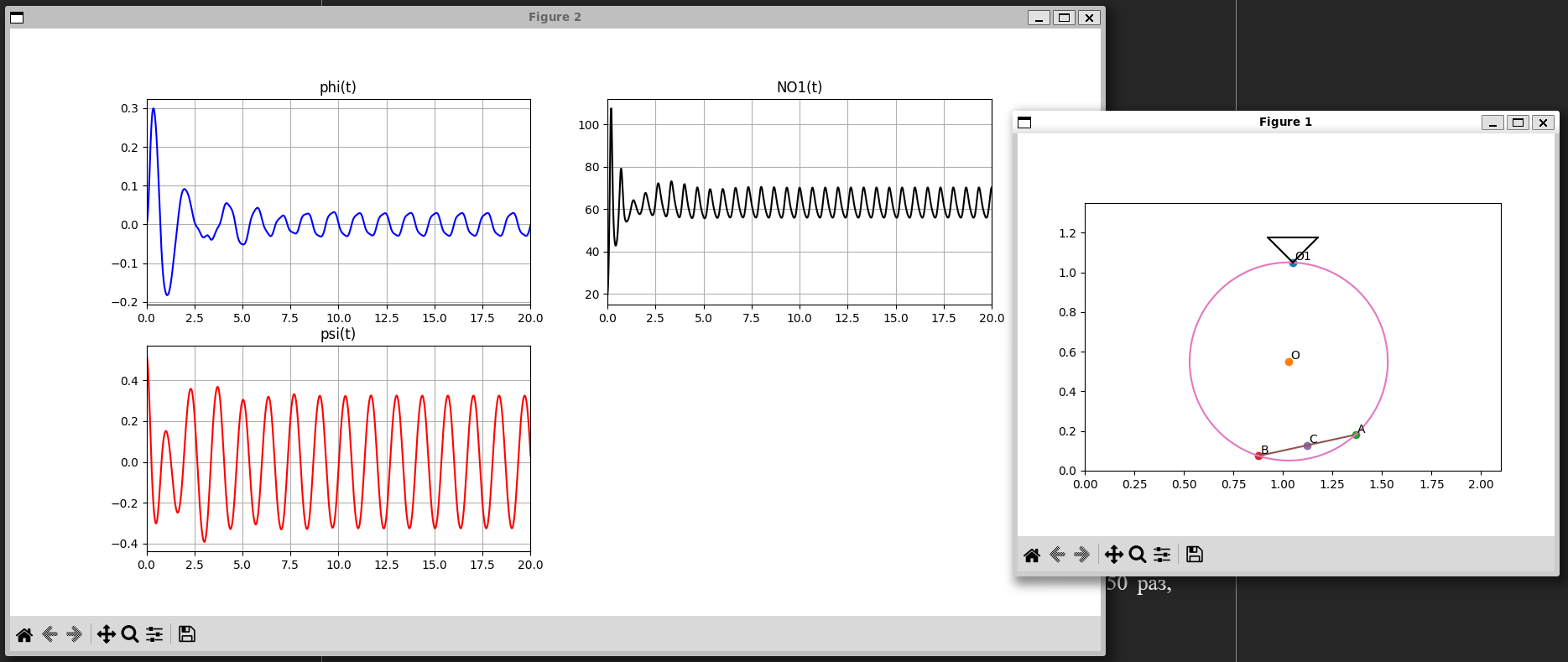


* 1. m1 = 100



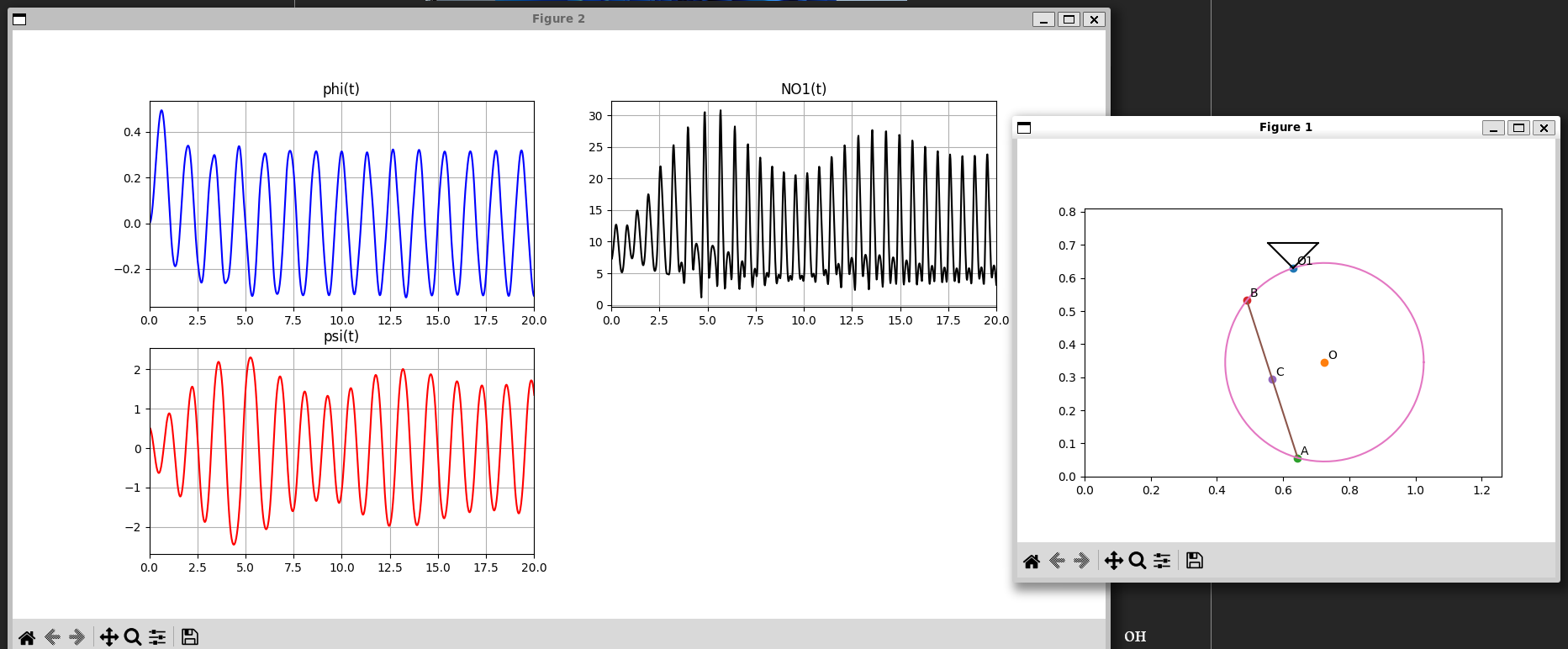
Кольцо стало очень тяжелым, его масса увеличилась в 50 раз, амплитуда φ сильно снизилась, примерно в 10 раз.

* 1. m2 = 10



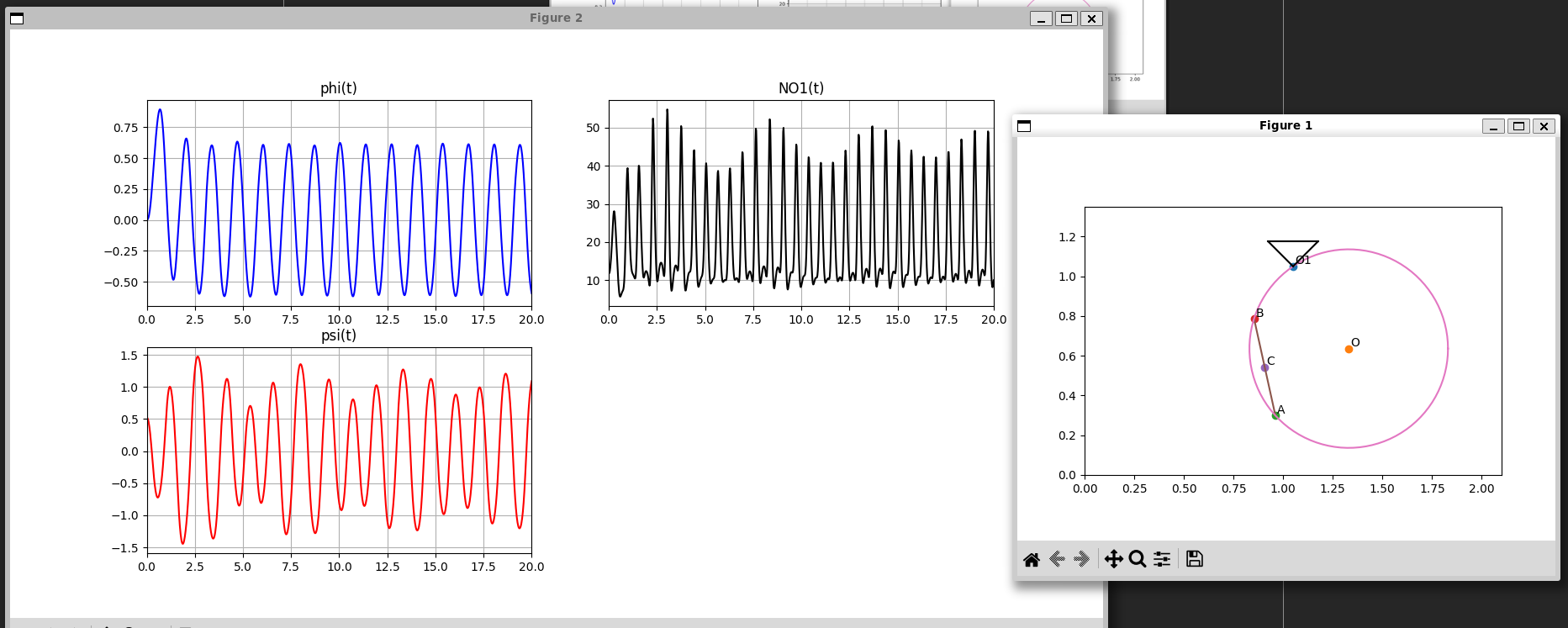
Стержень стал значительно тяжелее кольца, поэтому он достаточно быстро уравновешивает всю систему.

* 1. R = 0.3



Радиус окружности стал меньше, стержень начал более свободно перемещаться внутри окружности, амплитуда колебаний угла ψ увеличилась.

* 1. M0 = 30



Увеличили момент силы, действующей на кольцо. Амплитуды φ и ψ также увеличились, сил стало хватать, чтобы раскачать кольцо сильнее.

1. *Вывод*

Я научился интегрировать систему дифференциальных уравнений движения системы с двумя степенями свободы с помощью средств Python. Я построил анимацию движения системы, а также графики законов движения системы и указанных в задании реакций для разных случаев системы.