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

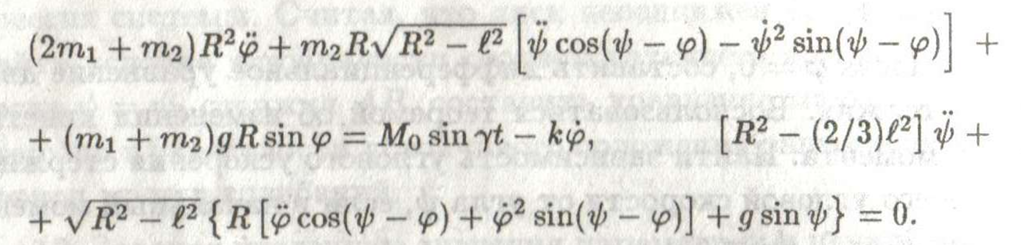
**Вариант №20**

**Задание:**

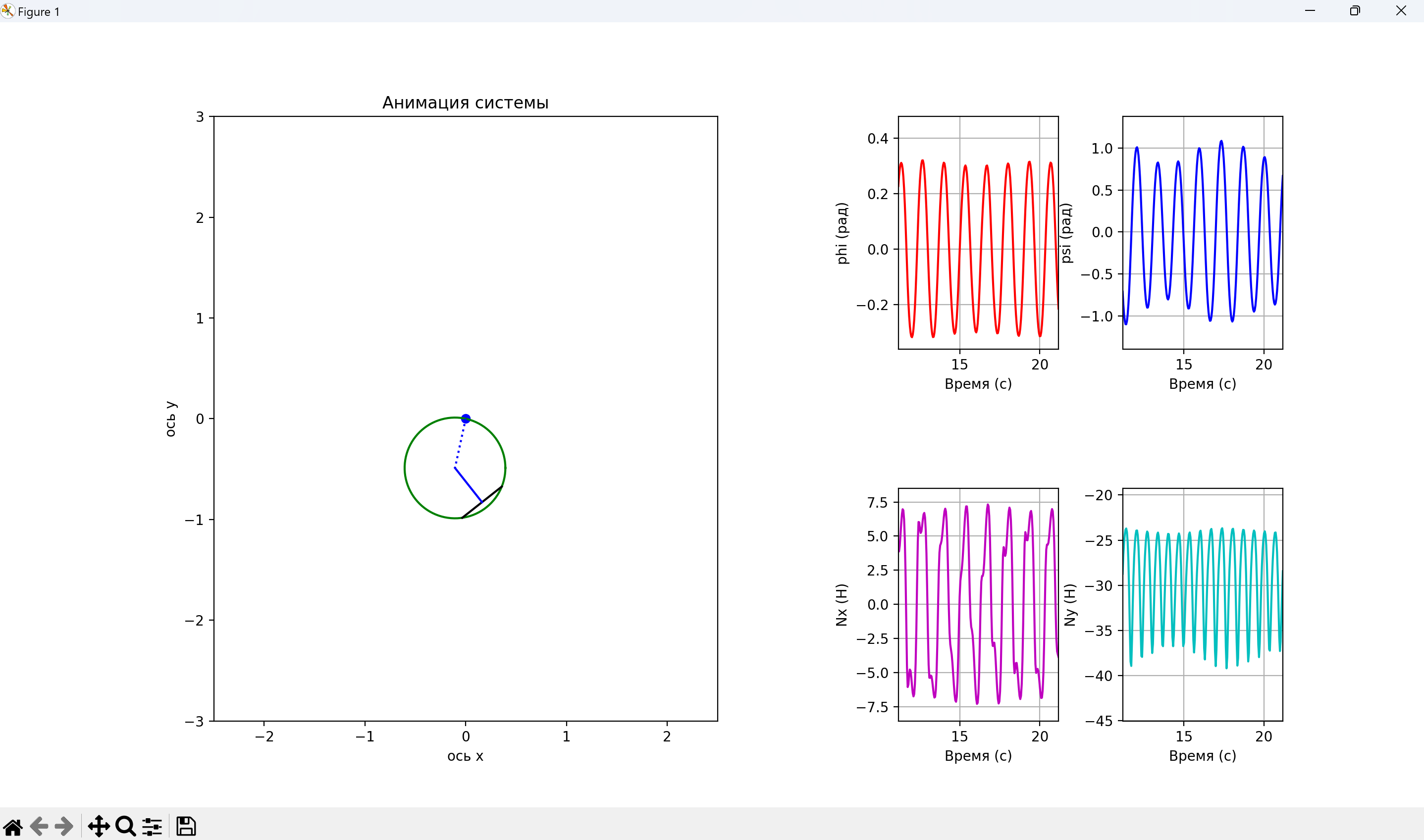
проинтегрировать систему дифференциальных уравнений движения системы с двумя степенями свободы с помощью средств Python. Построить анимацию движения системы, а также графики законов движения системы и указанных в задании реакций для разных случаев системы.

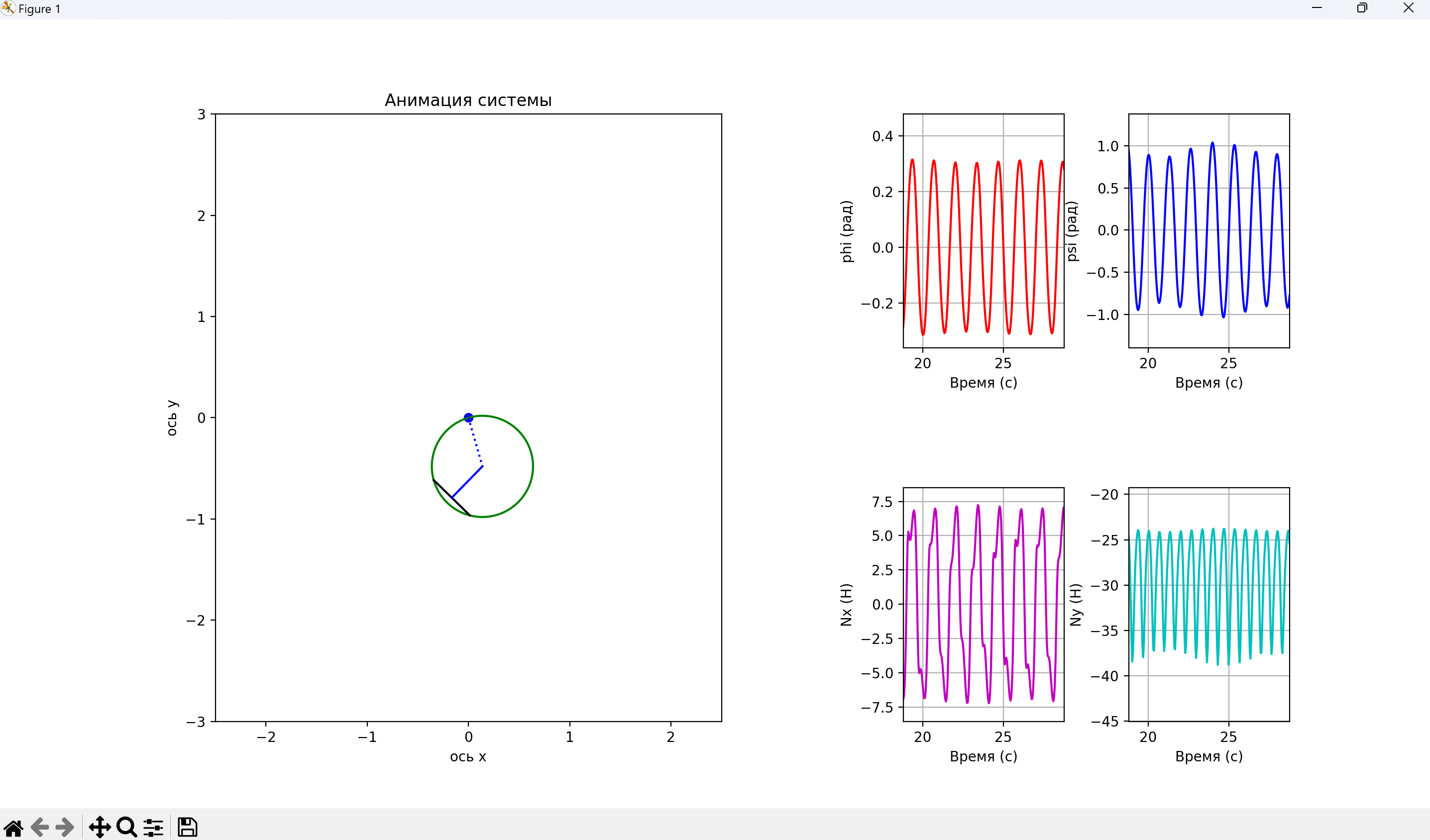
**Задание системы 20 варианта формулируется следующим образом:**

Однородное кольцо радиуса R и массы m1 вращается под действием пары сил с моментом М = М0\*sin(γt) вокруг горизонтальной оси, перпендикулярной его плоскости и проходящей через точку O1. Внутри кольца скользит без трения однородный стержень АВ массы m2 и длины 2l. К кольцу приложен момент сопротивления Мс = -k\*(fi)`, пропорциональный первой степени угловой скорости. Дифференциальные уравнения движения системы имеют вид:



**Рисунок получившейся анимации движения:**





**Код программы:**

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

import math

from scipy.integrate import odeint

import matplotlib.gridspec as gridspec

def Rot2D(X, Y, Alpha):

RX = X \* np.cos(Alpha) - Y \* np.sin(Alpha)

RY = X \* np.sin(Alpha) + Y \* np.cos(Alpha)

return RX, RY

m1 = 2.0

m2 = 1.0

R = 0.5

l = 0.25

M0 = 15.0

gamma = 3 \* math.pi / 2.0

k = 10.0

g = 9.81

t0 = 0.0

phi0 = 0.0

psi0 = math.pi / 6

dphi0 = 0.0

dpsi0 = 0.0

y0 = [phi0, psi0, dphi0, dpsi0]

Tmax = 45.0

Nsteps = 1000

T = np.linspace(t0, Tmax, Nsteps)

def SystDiffEq(y, t, m1, m2, R, l, M0, gamma, k, g):

phi, psi, dphi, dpsi = y

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

sin\_psi\_phi = math.sin(psi - phi)

cos\_psi\_phi = math.cos(psi - phi)

A11 = (2 \* m1 + m2) \* R\*\*2

A12 = m2 \* R \* sqrt\_Rl \* cos\_psi\_phi

A21 = sqrt\_Rl \* R \* cos\_psi\_phi

A22 = R\*\*2 - (2.0 / 3.0) \* l\*\*2

B1 = M0 \* math.sin(gamma \* t) - k \* dphi - (m1 + m2) \* g \* R \* \

math.sin(phi) + m2 \* R \* sqrt\_Rl \* (dpsi\*\*2) \* sin\_psi\_phi

B2 = -sqrt\_Rl \* (R \* (dphi\*\*2) \* sin\_psi\_phi + g \* math.sin(psi))

detA = A11 \* A22 - A12 \* A21

detA1 = B1 \* A22 - A12 \* B2

detA2 = A11 \* B2 - B1 \* A21

ddphi = detA1 / detA

ddpsi = detA2 / detA

return [dphi, dpsi, ddphi, ddpsi]

Y = odeint(SystDiffEq, y0, T, args=(m1, m2, R, l, M0, gamma, k, g))

phi\_array = Y[:, 0]

psi\_array = Y[:, 1]

dphi\_array = Y[:, 2]

dpsi\_array = Y[:, 3]

ddphi\_array = np.zeros(Nsteps)

ddpsi\_array = np.zeros(Nsteps)

for i in range(Nsteps):

y\_i = Y[i]

t\_i = T[i]

derivs = SystDiffEq(y\_i, t\_i, m1, m2, R, l, M0, gamma, k, g)

ddphi\_array[i] = derivs[2]

ddpsi\_array[i] = derivs[3]

Nx = - (m1 + m2) \* R \* (ddphi\_array \* np.cos(phi\_array) - dphi\_array\*\*2 \* np.sin(phi\_array)) \

- m2 \* (R\*\*2 - l\*\*2) \* (ddpsi\_array \* np.cos(psi\_array) -

dpsi\_array\*\*2 \* np.sin(psi\_array))

Ny = - (m1 + m2) \* R \* (ddphi\_array \* np.sin(phi\_array) + dphi\_array\*\*2 \* np.cos(phi\_array)) \

- (m1 + m2) \* g \

- m2 \* (R\*\*2 - l\*\*2) \* (ddpsi\_array \* np.sin(psi\_array) +

dpsi\_array\*\*2 \* np.cos(psi\_array))

x\_O = -R \* np.sin(phi\_array)

y\_O = R \* np.cos(phi\_array)

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

x\_C = x\_O - r \* np.sin(psi\_array)

y\_C = y\_O + r \* np.cos(psi\_array)

x\_rel = -r \* np.sin(psi\_array)

y\_rel = r \* np.cos(psi\_array)

x\_O\_rot = -x\_O

y\_O\_rot = -y\_O

x\_C\_rot = -x\_C

y\_C\_rot = -y\_C

x\_rel\_rot = -x\_rel

y\_rel\_rot = -y\_rel

fig = plt.figure(figsize=(16, 8))

gs = gridspec.GridSpec(1, 2, width\_ratios=[3, 2], wspace=0.3)

ax\_anim = plt.subplot(gs[0])

ax\_anim.set\_xlim(-2.5, 2.5)

ax\_anim.set\_ylim(-3, 3)

ax\_anim.set\_xlabel('ось x')

ax\_anim.set\_ylabel('ось y')

ax\_anim.set\_aspect('equal')

ax\_anim.set\_title('Анимация системы')

PointO1, = ax\_anim.plot([0], [0], 'bo')

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

Circ, = ax\_anim.plot(x\_O\_rot[0] + R \* np.cos(Circ\_Angle),

y\_O\_rot[0] + R \* np.sin(Circ\_Angle), 'g')

ArrowX = np.array([0, 0, 0])

ArrowY = np.array([-l, 0, l])

initial\_angle = math.atan2(y\_rel\_rot[0], x\_rel\_rot[0])

R\_Stick\_ArrowX, R\_Stick\_ArrowY = Rot2D(ArrowX, ArrowY, initial\_angle)

Stick\_Arrow, = ax\_anim.plot(

R\_Stick\_ArrowX + x\_C\_rot[0], R\_Stick\_ArrowY + y\_C\_rot[0], 'k-')

O1O, = ax\_anim.plot([0, x\_O\_rot[0]], [0, y\_O\_rot[0]], 'b:')

OC, = ax\_anim.plot([x\_O\_rot[0], x\_C\_rot[0]], [

y\_O\_rot[0], y\_C\_rot[0]], 'b-')

gs\_plots = gridspec.GridSpecFromSubplotSpec(

2, 2, subplot\_spec=gs[1], wspace=0.4, hspace=0.6)

ax\_phi = plt.subplot(gs\_plots[0, 0])

ax\_phi.set\_xlim(t0, 10)

ax\_phi.set\_ylim(min(phi\_array) \* 1.1, max(phi\_array) \* 1.1)

ax\_phi.set\_xlabel('Время (с)')

ax\_phi.set\_ylabel('phi (рад)')

line\_phi, = ax\_phi.plot([], [], 'r-')

ax\_phi.grid(True)

ax\_psi = plt.subplot(gs\_plots[0, 1])

ax\_psi.set\_xlim(t0, 10)

ax\_psi.set\_ylim(min(psi\_array) \* 1.1, max(psi\_array) \* 1.1)

ax\_psi.set\_xlabel('Время (с)')

ax\_psi.set\_ylabel('psi (рад)')

line\_psi, = ax\_psi.plot([], [], 'b-')

ax\_psi.grid(True)

ax\_Nx = plt.subplot(gs\_plots[1, 0])

ax\_Nx.set\_xlim(t0, 10)

ax\_Nx.set\_ylim(min(Nx) \* 1.1, max(Nx) \* 1.1)

ax\_Nx.set\_xlabel('Время (с)')

ax\_Nx.set\_ylabel('Nx (Н)')

line\_Nx, = ax\_Nx.plot([], [], 'm-')

ax\_Nx.grid(True)

ax\_Ny = plt.subplot(gs\_plots[1, 1])

ax\_Ny.set\_xlim(t0, 10)

padding = 0.1 \* max(abs(min(Ny)), abs(max(Ny)))

ax\_Ny.set\_ylim(min(Ny) - padding, max(Ny) + padding)

ax\_Ny.set\_xlabel('Время (с)')

ax\_Ny.set\_ylabel('Ny (Н)')

line\_Ny, = ax\_Ny.plot([], [], 'c-')

ax\_Ny.grid(True)

phi\_xdata, phi\_ydata = [], []

psi\_xdata, psi\_ydata = [], []

Nx\_xdata, Nx\_ydata = [], []

Ny\_xdata, Ny\_ydata = [], []

def anima(i):

O1O.set\_data([0, x\_O\_rot[i]], [0, y\_O\_rot[i]])

OC.set\_data([x\_O\_rot[i], x\_C\_rot[i]], [y\_O\_rot[i], y\_C\_rot[i]])

Circ.set\_data(x\_O\_rot[i] + R \* np.cos(Circ\_Angle),

y\_O\_rot[i] + R \* np.sin(Circ\_Angle))

current\_angle = math.atan2(y\_rel\_rot[i], x\_rel\_rot[i])

R\_Stick\_ArrowX, R\_Stick\_ArrowY = Rot2D(ArrowX, ArrowY, current\_angle)

Stick\_Arrow.set\_data(

R\_Stick\_ArrowX + x\_C\_rot[i], R\_Stick\_ArrowY + y\_C\_rot[i])

phi\_xdata.append(T[i])

phi\_ydata.append(phi\_array[i])

line\_phi.set\_data(phi\_xdata, phi\_ydata)

psi\_xdata.append(T[i])

psi\_ydata.append(psi\_array[i])

line\_psi.set\_data(psi\_xdata, psi\_ydata)

Nx\_xdata.append(T[i])

Nx\_ydata.append(Nx[i])

line\_Nx.set\_data(Nx\_xdata, Nx\_ydata)

Ny\_xdata.append(T[i])

Ny\_ydata.append(Ny[i])

line\_Ny.set\_data(Ny\_xdata, Ny\_ydata)

window = 10

if T[i] > window:

ax\_phi.set\_xlim(T[i] - window, T[i])

ax\_psi.set\_xlim(T[i] - window, T[i])

ax\_Nx.set\_xlim(T[i] - window, T[i])

ax\_Ny.set\_xlim(T[i] - window, T[i])

while phi\_xdata and phi\_xdata[0] < T[i] - window:

phi\_xdata.pop(0)

phi\_ydata.pop(0)

line\_phi.set\_data(phi\_xdata, phi\_ydata)

while psi\_xdata and psi\_xdata[0] < T[i] - window:

psi\_xdata.pop(0)

psi\_ydata.pop(0)

line\_psi.set\_data(psi\_xdata, psi\_ydata)

while Nx\_xdata and Nx\_xdata[0] < T[i] - window:

Nx\_xdata.pop(0)

Nx\_ydata.pop(0)

line\_Nx.set\_data(Nx\_xdata, Nx\_ydata)

while Ny\_xdata and Ny\_xdata[0] < T[i] - window:

Ny\_xdata.pop(0)

Ny\_ydata.pop(0)

line\_Ny.set\_data(Ny\_xdata, Ny\_ydata)

else:

ax\_phi.set\_xlim(t0, window)

ax\_psi.set\_xlim(t0, window)

ax\_Nx.set\_xlim(t0, window)

ax\_Ny.set\_xlim(t0, window)

return O1O, OC, Circ, Stick\_Arrow, line\_phi, line\_psi, line\_Nx, line\_Ny

anim = FuncAnimation(fig, anima,

frames=Nsteps, interval=20, blit=False)

plt.tight\_layout()

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