ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«ДИНАМИКА СИСТЕМЫ»**

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

**ВАРИАНТ ЗАДАНИЯ №15**

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

КРУТЫХ РОМАН ВЯЧЕСЛАВОВИЧ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

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

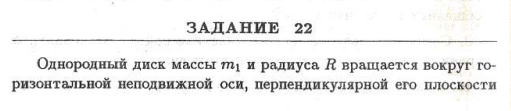
подпись, дата

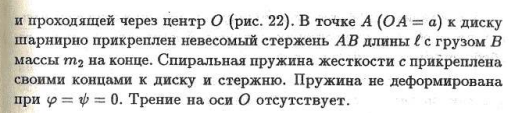
с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

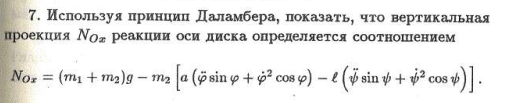
Москва, 2023

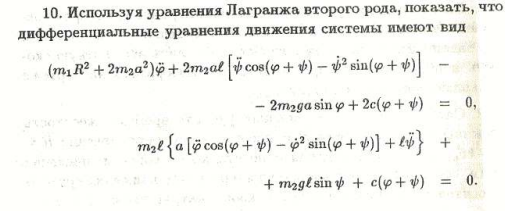
*Задание:* проинтегрировать систему дифференциальных уравнений движения системы с двумя степенями свободы с помощью средств Python. Построить анимацию движения системы, а также графики законов движения системы и указанных в задании реакций для разных случаев системы.

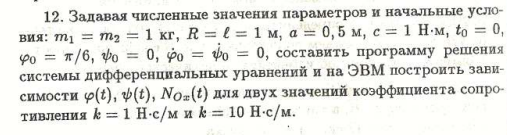
*Условия задачи для 22 (15 + 7) варианта:*

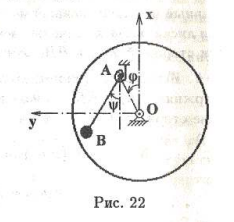
**

**

**

**

**

**

*Код программы:*

import math

import numpy as np

from matplotlib.animation import FuncAnimation

import matplotlib.pyplot as plt

from scipy.integrate import odeint

def odesys(y, t, m1, m2, r, l, a, c , g):

dy = np.zeros(4)

dy[0] = y[2]

dy[1] = y[3]

a11 = m1\*r\*r + 2 \* m2 \* a \* a

a12 = 2 \* m2 \* a \* l \* np.cos(y[0] + y[1])

a21 = m2\*l\*a\*np.cos(y[0] + y[1])

a22 = m2\*l\*l

b1 = 2\*m2\*a\*l\*y[3]\*y[3]\*np.sin(y[0] + y[1]) + 2\*m2\*g\*a\*np.sin(y[0]) - 2\*c\*(y[0] + y[1])

b2 = m2\*l\*a\*y[2]\*y[2]\*np.sin(y[0] + y[1]) - m2\*g\*l\*np.sin(y[1]) - c\*(y[0] + y[1])

dy[2] = (b1\*a22 - b2\*a12)/(a11\*a22 - a12\*a21)

dy[3] = (b2\*a11 - b1\*a21)/(a11\*a22 - a12\*a21)

return dy

# data of task

m1 = 1

m2 = 1

r = 1

l = 1

a =0.5

c = 1

g = 9.81

steps = 1001

t\_fin = 10

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

phi0 = np.pi/6

psi0 = 0

dphi0 = 0

dpsi0 = 0

y0 = [phi0, psi0, dphi0, dpsi0]

Y = odeint(odesys, y0, t, ( m1, m2, r, l, a, c , g))

phi = Y[:, 0]

psi = Y[:, 1]

dphi = Y[:,2]

dpsi = Y[:, 3]

ddphi = [odesys(y, t, m1, m2, r, l, a, c, g)[2] for y, t in zip(Y, t)]

ddpsi = [odesys(y, t, m1, m2, r, l, a, c, g)[3] for y, t in zip(Y, t)]

Nox = (m1 + m2) \* g - m2 \* (a\*(ddphi\*np.sin(phi) + dphi \* dphi \* np.cos(phi)) - l\*(ddpsi\*np.sin(psi) + dpsi\*dpsi \* np.cos(psi)))

#code from old lab

def Circle1(X, Y, radius):

CX = [X + radius \* math.cos(i / 100) for i in range(0, 628)]

CY = [Y + radius \* math.sin(i / 100) for i in range(0, 628)]

return CX, CY

l = 5

alpha\_angle = 30

alpha = 5

r = 8

c = 1

g = 9.81

t = np.linspace(0, 10, 1001)

#задаем точки системы и строим линии

X\_start = 0

Y\_start = 0

X\_O = X\_start

Y\_O = Y\_start

X\_A = X\_O - alpha \* np.sin(phi)

Y\_A = Y\_O + alpha \* np.cos(phi)

X\_B = X\_A - l \* np.sin(psi)

Y\_B = Y\_A - l \* np.cos(psi)

fig = plt.figure(figsize=[10, 7])

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

ax.axis('equal')

ax.set(xlim=[-25, 25], ylim=[-25, 25])

Point\_O = ax.plot(X\_O, Y\_O, marker='o', color='black')[0]

Point\_A = ax.plot(X\_A[0], Y\_A[0], marker='o', color='black')[0]

Point\_B = ax.plot(X\_B[0], Y\_B[0], marker='o', markersize=20, color='black')[0]

circle1 = ax.plot(\*Circle1(X\_O, Y\_O, r), 'red') # main circle

triangle, = ax.plot([-0.5, 0, 0.5],

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

line\_tr = ax.plot([- 0.5, 0.5], [-1, -1],

color='black')[0]

Line\_AB = ax.plot([X\_A[0], X\_B[0]], [Y\_A[0], Y\_B[0]], color='black', linewidth=3)[0]

# spiral spring

Nv = 2.6

R1 = 0.2

R2 = 5

thetta = np.linspace(0, Nv \* 6.28 - psi[0], 1001)

X\_SpiralSpr = -(R1 \* thetta \* (R2 - R1) / thetta[-1]) \* np.sin(thetta)

Y\_SpiralSpr = (R1 \* thetta \* (R2 - R1) / thetta[-1]) \* np.cos(thetta)

Drawed\_Spiral\_Spring = ax.plot(X\_SpiralSpr + X\_A[0], Y\_SpiralSpr + Y\_A[0], color='black')[0]

#рисуем графики

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, Nox, color='Black')

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

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

ax\_for\_graphs.grid(True)

def anima(i): #функция анимации

Point\_A.set\_data(X\_A[i], Y\_A[i])

Point\_B.set\_data(X\_B[i], Y\_B[i])

Line\_AB.set\_data([X\_A[i], X\_B[i]], [Y\_A[i], Y\_B[i]])

thetta = np.linspace(0, Nv \* 6 - psi[i], 100)

X\_SpiralSpr = -(R1 \* thetta \* (R2 - R1) / thetta[-1]) \* np.sin(thetta)

Y\_SpiralSpr = (R1 \* thetta \* (R2 - R1) / thetta[-1]) \* np.cos(thetta)

Drawed\_Spiral\_Spring.set\_data(X\_SpiralSpr + X\_A[i], Y\_SpiralSpr +

Y\_A[i])

return [Point\_A, Point\_B, Line\_AB, Drawed\_Spiral\_Spring]

anim = FuncAnimation(fig, anima, frames=1000, interval=10)

plt.show()

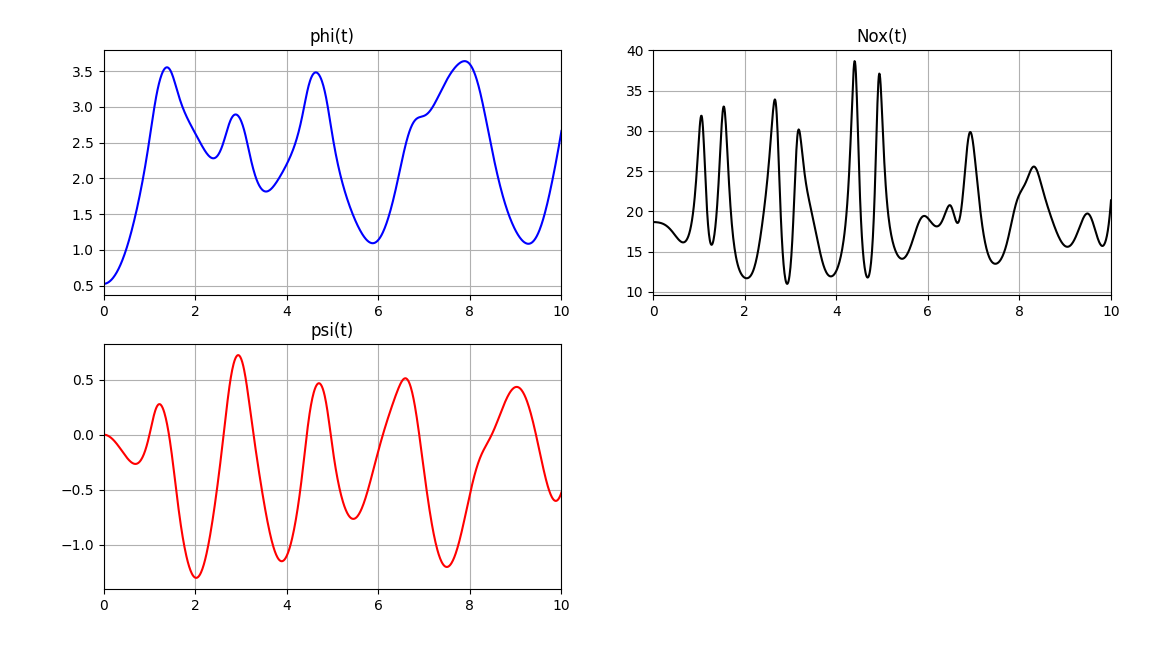
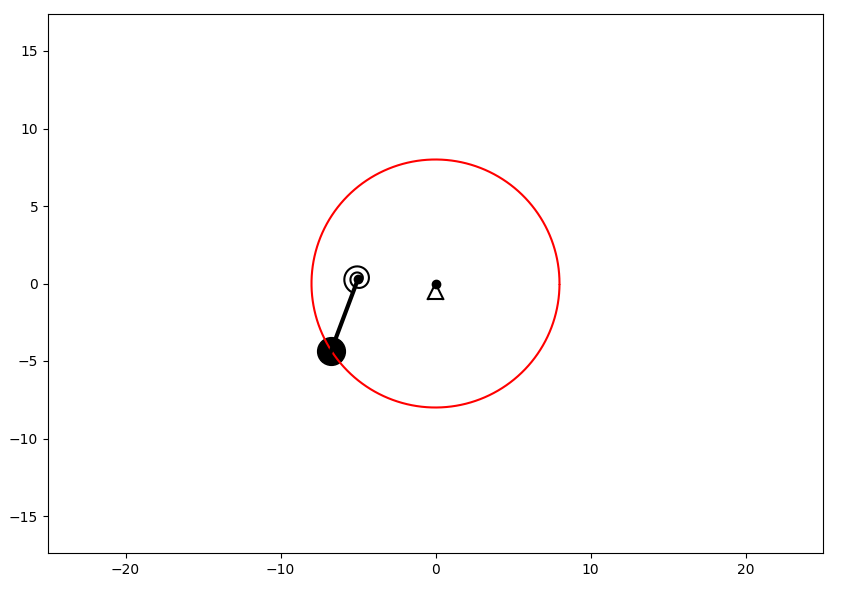
*Результаты работы программы:*

*1. m1 = m2 = 1; R = 1; l = 1; a =0.5; c = 1; g = 9.81.*

*phi0 = pi/6;*

*psi0 = dpsi0 = dphi0 = 0.*

*При данных значениях система движется относительно медленно, диск медленно вращается вокруг оси, пружина немного раскручивает стержень с грузом.*

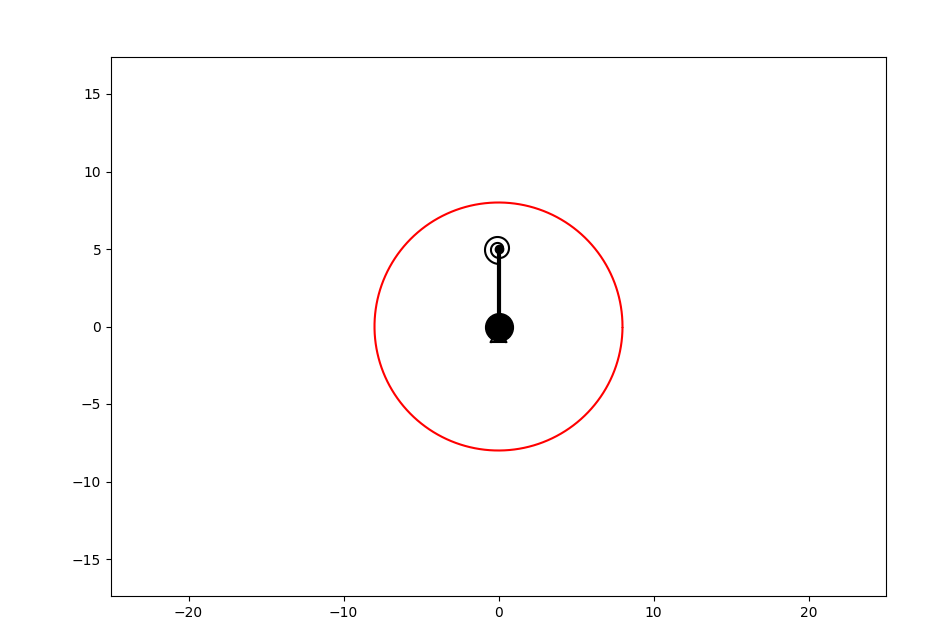


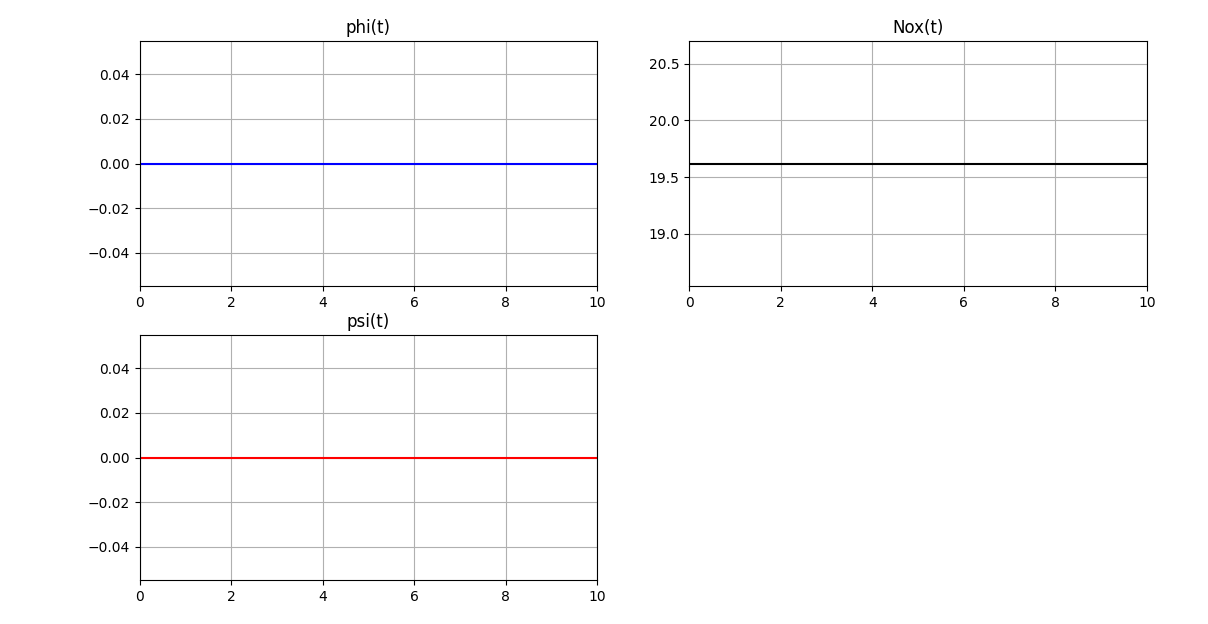
*2. m1 = 0.05; m2 = 1; R = 1; l = 1; a =0.5; c = 1; g = 9.81.*

*phi0 =0;*

*psi0 = dpsi0 = dphi0 = 0.*

*При уменьшении массы диска система начинает активнее и быстрее вращаться.*



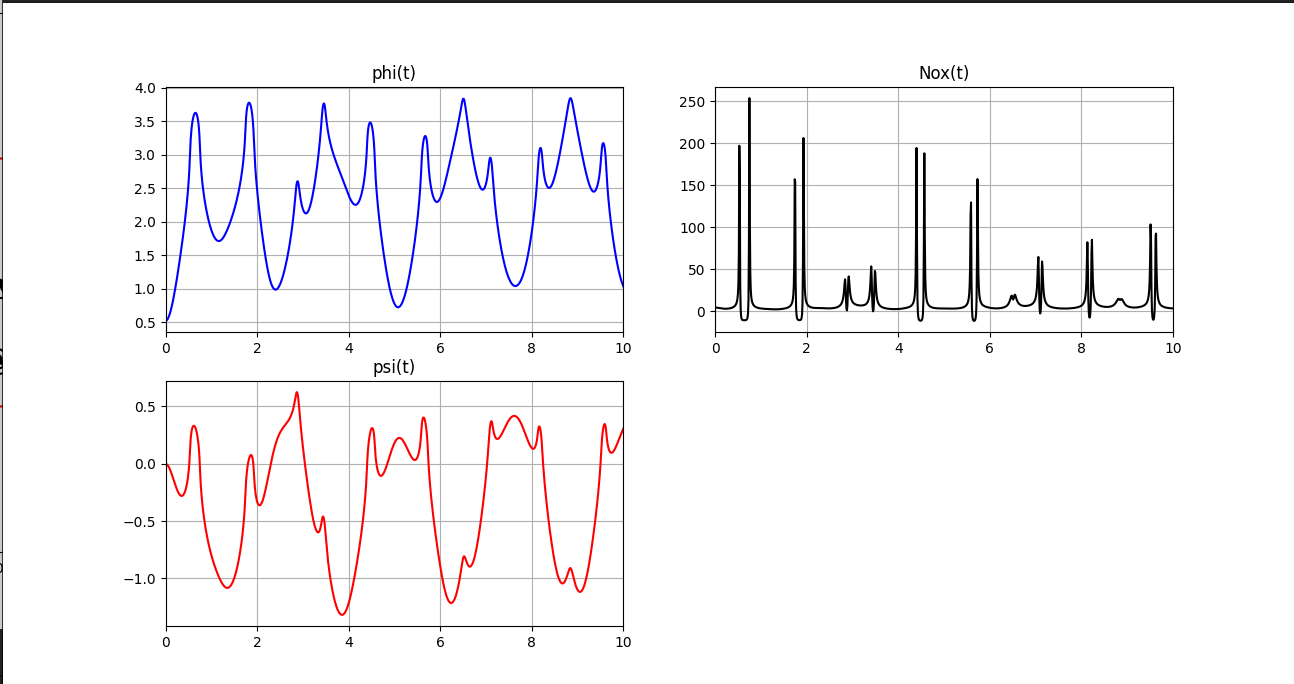
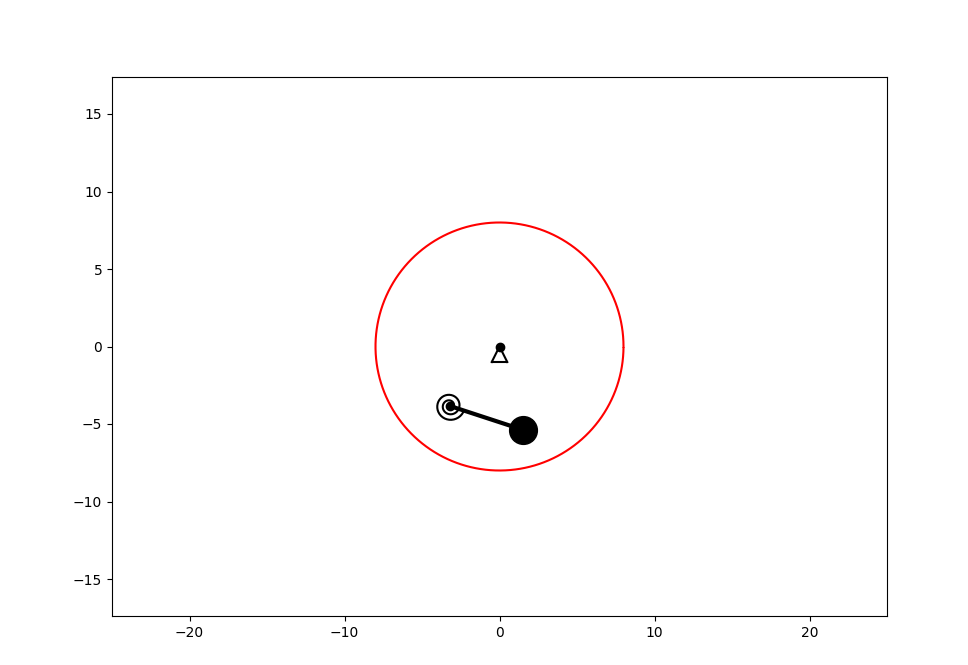
**

*3. m1 = 0.05; m2 = 1; R = 1; l = 1; a =0.5; c = 1; g = 9.81.*

*phi0 =0;*

*psi0 = dpsi0 = dphi0 = 0.*

*При уменьшении массы диска система начинает активнее и быстрее вращаться.*

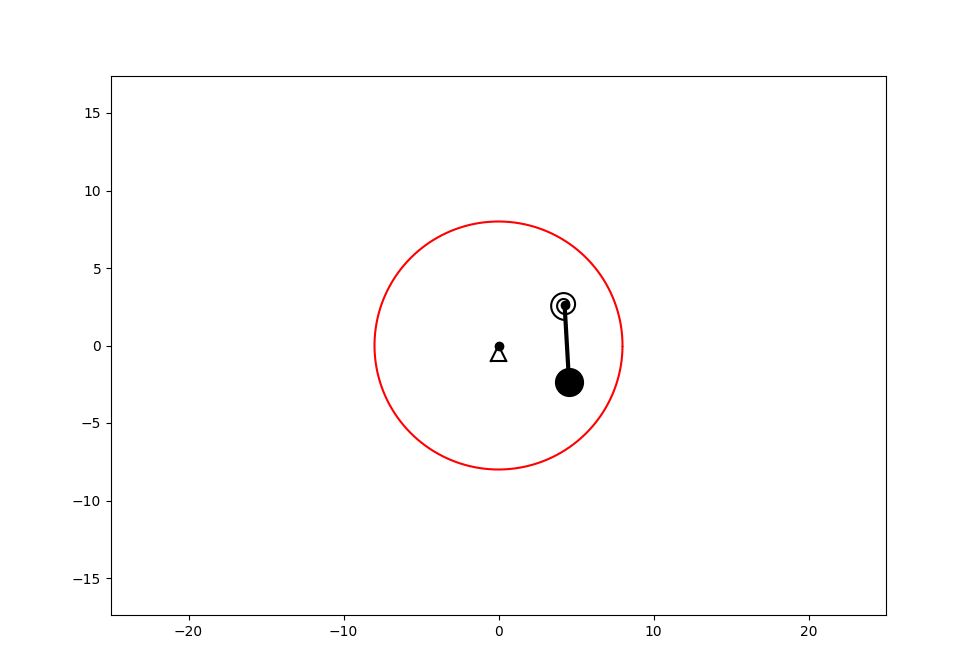
**

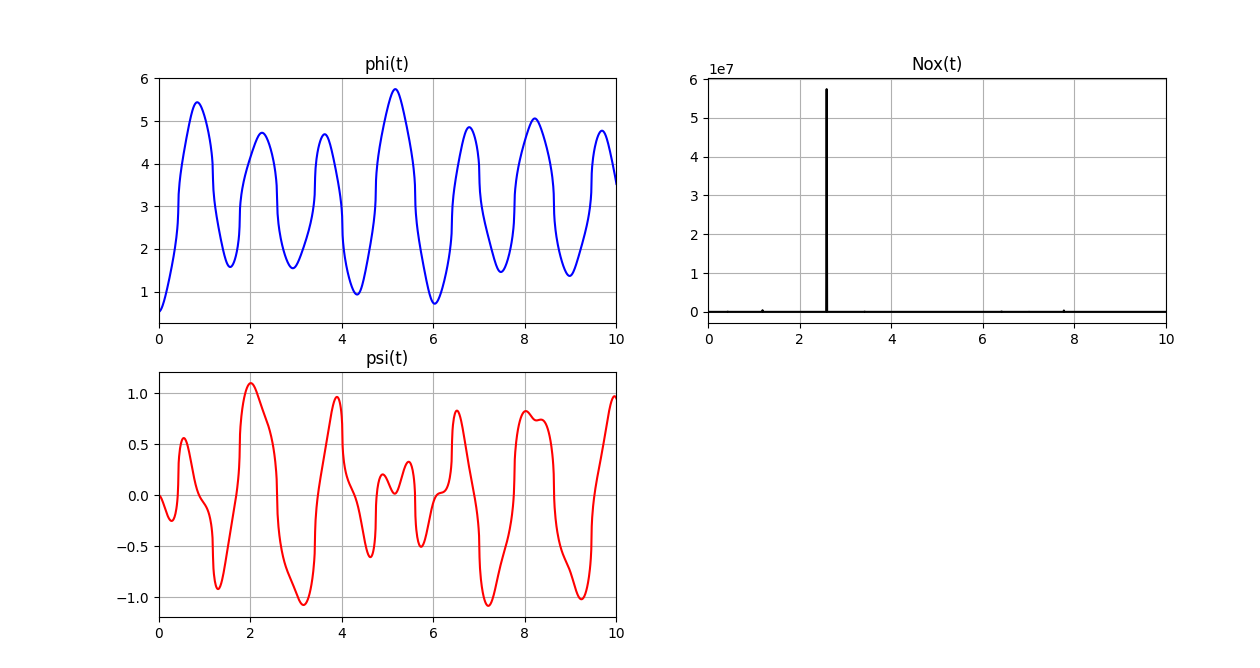
*4. 1. m1 = 1; m2 = 2000; R = 1; l = 1; a =0.5; c = 1; g = 9.81.*

*phi0 = pi/6;*

*psi0 = dpsi0 = dphi0 = 0.*

*При увеличении массы груза система будет двигаться значительно быстрее. Диск будет сильнее вращаться, но в то же время пружина будет меньше закручиваться.*

**

**

*Вывод:*

*В ходе выполнения данной лабораторной работы я написал код на языке программирования Python, который строит анимацию движения системы. В этом коде уравнения движения могут быть модифицированы путем изменения коэффициентов (начальных значений).*

*Также я реализовал 3 графика, которые отображают изменение phi(t), psi(t), N0x(t) (вертикальная проекция реакции оси диска).*