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 №27**

Выполнил(а) студент группы М8О-201Б-22

Чибугаев И.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Волков В.Е.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Москва, 2024

*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.

*Код:*

import numpy as n

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from scipy.integrate import odeint

def SystDiffEq(y, t, m1, m2, m3, r, R, g, M1, M2, c):

*# y = [ phi, theta, phi', theta'] -> dy = [ phi', theta', phi'', theta'']*

    dy = n.zeros\_like(y)

    dy[0] = y[2]

    dy[1] = y[3]

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

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

*# коэффициенты первого уравнения*

    a11 = (3\*m2 +(2/3)\*m3) \* (R-r)\*\*2

    a12 = (R-r) \* m2 \* R

    b1 = (2\*m2 + m3) \* g \*(R-r)\* n.sin(y[0]) - 2\*c\*y[0] + 2\*M2

*# коэффициенты второго уравнения*

    a21 = m2 \* R \* (R-r)

    a22 = (m1 + m2) \* R\*\*2

    b2 = 2 \* M1

*# решение правилом Крамера*

    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

*#задается интервал*

step = 1000

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

*#начальные значения*

O = 0 *#начало координат*

r2 = 0.2*#внешний радиус пружины*

r1 = 0.05*#внутренний радиус пружины*

m1 = 2

m2 = 1

m3 = 1

c = 2

M1 = 0.5

M2 = 0.2

g = 9.8

r = 0.2 *#r маленького колеса*

R = 1 *#R большого колеса*

*#задаем начальные значения*

y0 = [n.pi/6, n.pi/2, 0 , 2]

*#заполняем массив производными*

Y = odeint(SystDiffEq, y0, t, (m1, m2, m3, r, R, g, M1, M2, c))

*#заполняем функции*

phi = Y[:,0]

theta = Y[:,1]

phit = Y[:,2]

thetat = Y[:,3]

phitt = n.zeros\_like(t)

thetatt = n.zeros\_like(t)

NAx = n.zeros\_like(t)

NAy = n.zeros\_like(t)

for i in range(len(t)):

    phitt[i] = SystDiffEq(Y[i], t[i], m1, m2, m3, r, R, g, M1, M2, c)[2]

    thetatt[i] =  SystDiffEq(Y[i], t[i], m1, m2, m3, r, R, g, M1, M2, c)[3]

    NAx = m2\*(R-r)\*(phitt\*n.cos(phi[i]) - phit[i]\*n.sin(phi[i]))

    NAy = m2\*(-(R-r)\*(phitt\*n.sin(phi[i]) + phit[i]\*n.cos(phi[i])+g))

*#выводим графики зависимости*

fgrt = plt.figure()

phiplt = fgrt.add\_subplot(2,2,1)

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

phiplt.set\_title('Phi(t)')

thetaplt = fgrt.add\_subplot(2,2,2)

thetaplt.plot(t, theta, color = 'red')

thetaplt.set\_title('Theta(t)')

NAxplt = fgrt.add\_subplot(2,2,3)

NAxplt.plot(t, NAx, color = 'orange')

NAxplt.set\_title('NAx(t)')

NAyplt = fgrt.add\_subplot(2,2,4)

NAyplt.plot(t, NAy, color = 'orange')

NAyplt.set\_title('NAy(t)')

fgrt.show()

*#координаты точки А*

Xa = (R - r) \* n.cos(phi)

Ya = (R - r) \* n.sin(phi)

*#координаты точки Б*

Xb = R \* n.cos(n.pi/2+theta)

Yb = R \* n.sin(n.pi/2+theta)

*#окно и грaфик*

fgr = plt.figure()

grf = fgr.add\_subplot(1, 1, 1)

grf.axis('equal')

grf.set(xlim = [-3, 3], ylim = [-3, 3])

grf.set\_aspect( 1 )

*#треугольник*

grf.plot([0,0.1,-0.1,0], [0,-0.2,-0.2,0], color = 'black')

*#точки на графике*

p1 = grf.plot(O, O, marker = 'o', color = 'black')[0]

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

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

*#прямая ОА*

O1 = grf.plot([Xa[0], O],[Ya[0], O],color = 'black')[0]

*#большая окружность*

circle = plt.Circle(( O, O ), R , fill = False)

grf.add\_artist(circle)

*#маленькая окружность*

circleA = plt.Circle(( Xa[0], Ya[0]), r , fill = False)

grf.add\_artist(circleA)

*#спиральная пружина*

Ns = 2

numpnts = n.linspace(0, 1, 50\*Ns+1)

Betas = numpnts\*(Ns \* 2\*n.pi/ + phi[0])

Xs = ((r2-r1)\*numpnts)\*n.cos(Betas + n.pi/2)

Ys = ((r2-r1)\*numpnts)\*n.sin(Betas + n.pi/2)

SpPruzh = grf.plot(Xs, Ys, color = 'black')[0]

*#анимация*

def run(i):

    pA.set\_data([Xa[i]], [Ya[i]])

    pB.set\_data([Xb[i]], [Yb[i]])

    O1.set\_data([Xa[i], O], [Ya[i], O])

    circleA.center = (Xa[i], Ya[i])

    Betas = numpnts \* (Ns \* 2 \* n.pi - phi[i])

    Xs = -((r2 - r1) \* numpnts) \* n.cos(Betas + n.pi)

    Ys = ((r2 - r1) \* numpnts) \* n.sin(Betas + n.pi)

    SpPruzh.set\_data(Xs, Ys)

    return

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

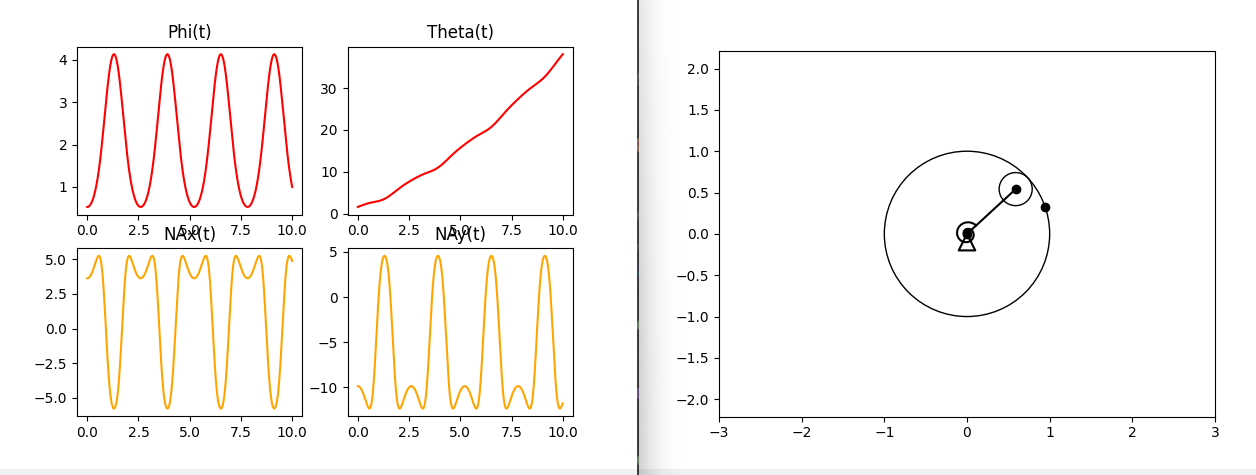
fgr.show()

*Скриншот:*

**Случай 1: Исходные параметры**

* **m1 = 2, m2 = 1, m3 = 1, r = 0.2, R = 1, c = 2, M1 = 0.5, M2 = 0.2**
* Начальные условия: phi = π/6, theta = π/2, phi' = 0, theta' = 2

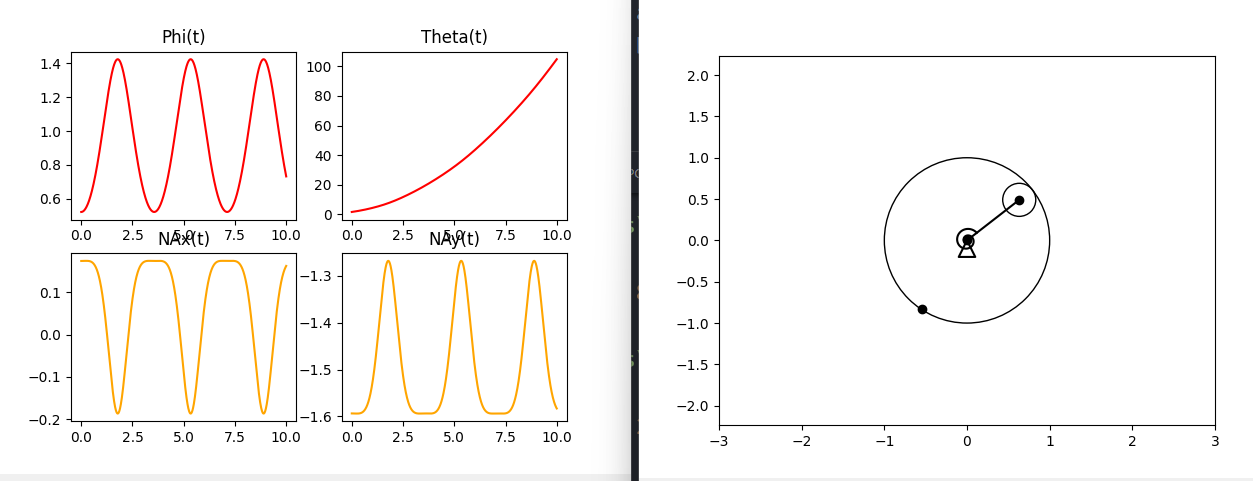
**Описание:** В данном случае мы использовали исходные параметры программы. Ожидается плавное движение системы без резких изменений. Колебания сохраняют устойчивый характер.



**Случай 2: Уменьшение масс m1, m2, m3 в 5 раз**

* **m1 = 0.4, m2 = 0.2, m3 = 0.2**
* Остальные параметры оставлены без изменений.

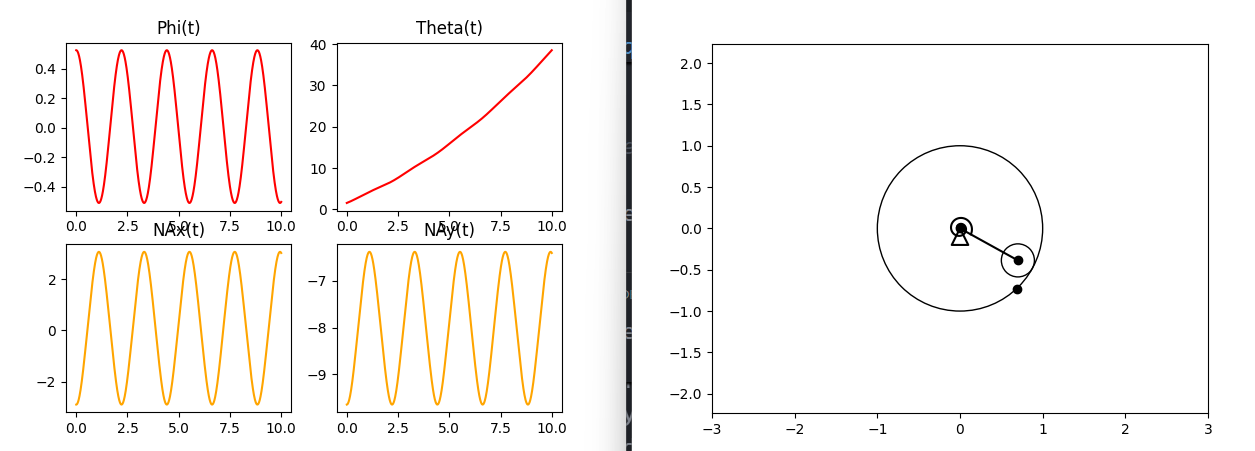
**Описание:** При уменьшении массы система становится более чувствительной к внешним воздействиям, что может привести к более резким колебаниям. Амплитуды движений ожидаются выше из-за меньшей инерции.



**Случай 3: Увеличение жесткости пружины в 10 раз**

* **c = 20**
* Остальные параметры оставлены без изменений.

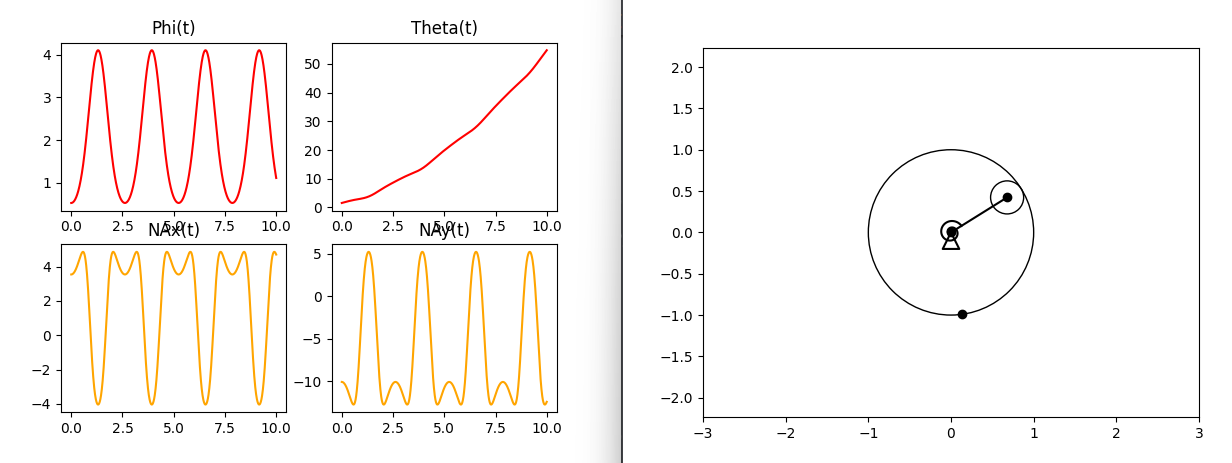
**Описание:** Увеличение жесткости приводит к снижению амплитуды колебаний. Движения становятся более сглаженными, система быстрее стремится к равновесию.



**Случай 4: Увеличение момента M1 в 2 раза**

* **M1 = 1.0**
* Остальные параметры оставлены без изменений.

**Описание:** Увеличение момента приводит к усилению вращательного эффекта, что влияет на траекторию движения точек и увеличивает динамичность системы.



*Вывод:*Я успешно выполнил лабораторную работу по теоретической механике. С помощью языка программирования Python и библиотек matplotlib и numpy я схематично проанимировал движение двух стержней и пружины и решил систему дифференциальных уравнений.

Благодаря этой лабораторной работе, я научился работать с 2д анимацией в matplotlib и реализовал основание для выполнения следующей лабораторной работы.

В моей программе используются реальные законы движения, благодаря чему можно посмотреть, как эта система будет вести себя в реальной жизни.