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 №4**

Выполнил(а) студент группы M8O-201Б-23

Вельма А.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Ст. преп. каф. 802 Волков Е.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

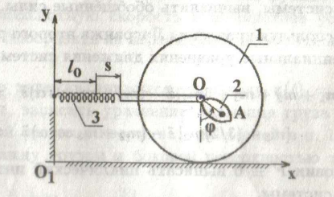
с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2024

**Задание:**

Используя средства языка программирования Python, проинтегрировать систему дифференциальных уравнений движения механической системы, включающей призму, пружину и балку-маятник. Построить анимацию движения системы, а также графики законов движения системы и указанных в задании зависимостей для различных параметров системы.

**Механическая система:**



**Текст программы**

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from scipy.integrate import odeint

import sympy as sp

import math

from sympy.series import O

# Функция для поворота координат вокруг точки (XC, YC) на угол Alpha

def Rot(X, Y, Alpha, XC, YC):

    RX = (X - XC) \* np.cos(Alpha) - (Y - YC) \* np.sin(Alpha) + XC  # Новая координата X

    RY = (X - XC) \* np.sin(Alpha) + (Y - YC) \* np.cos(Alpha) + YC  # Новая координата Y

    return RX, RY

# Функция для составления системы дифференциальных уравнений

def formY(y, t, fV, fOm):

    y1, y2, y3, y4 = y  # y1 = s (смещение), y2 = alpha (угол), y3 = ds/dt (скорость), y4 = d(alpha)/dt (угловая скорость)

    dydt = [y3, y4, fV(t, y1, y2, y3, y4), fOm(t, y1, y2, y3, y4)]  # Формирование системы уравнений

    return dydt

# Размеры и параметры системы

R = 1  # Радиус призмы

r = 0.1  # Длина балки

m1 = 10  # Масса призмы

m2 = 5  # Масса балки

g = 9.81  # Ускорение свободного падения

M\_0 = 0  # Амплитуда момента

w = math.pi  # Угловая частота момента

c = 35  # Жесткость пружины

# Символьные переменные и функции

t = sp.Symbol("t")  # Время

s = sp.Function("s")(t)  # Смещение призмы

alpha = sp.Function("alpha")(t)  # Угол наклона балки

V = sp.Function("V")(t)  # Скорость призмы (ds/dt)

om = sp.Function("om")(t)  # Угловая скорость балки (d(alpha)/dt)

M = sp.Function("M")(t)  # Момент силы

# Проверка вычислений производной

print(sp.diff(5 \* V\*\*2, V))  # Производная для проверки работы SymPy

# Кинетическая энергия

TTR = m1 \* V\*\*2 / 2 + m1 \* V\*\*2 / 4  # Кинетическая энергия призмы

Vc2 = om\*\*2 \* r\*\*2 + V\*\*2 - 2 \* om \* V \* r \* sp.cos(alpha)  # Квадрат скорости центра масс балки

TTr = (m2 \* Vc2) / 2 + (m2 \* r\*\*2) \* om\*\*2 / 2  # Кинетическая энергия балки

TT = TTR + TTr  # Полная кинетическая энергия системы

# Потенциальная энергия

Pi1 = -m2 \* g \* r \* sp.cos(alpha)  # Потенциальная энергия балки в поле тяжести

Pi2 = c \* (s - 1) \*\* 2 / 2  # Энергия сжатия пружины

Pi = Pi1 + Pi2  # Полная потенциальная энергия системы

# Лагранжева функция

M = M\_0 \* sp.cos(V) / R  # Момент силы, зависящий от скорости

L = TT - Pi  # Лагранжева функция (разность энергий)

# Уравнения движения Лагранжа

ur1 = sp.diff(sp.diff(L, V), t) - sp.diff(L, s) - M  # Уравнение движения по смещению

ur2 = sp.diff(sp.diff(L, om), t) - sp.diff(L, alpha)  # Уравнение движения по углу

# Решение уравнений методом Крамера

a11 = ur1.coeff(sp.diff(V, t), 1)  # Коэффициент при dV/dt в первом уравнении

a12 = ur1.coeff(sp.diff(om, t), 1)  # Коэффициент при dom/dt в первом уравнении

a21 = ur2.coeff(sp.diff(V, t), 1)  # Коэффициент при dV/dt во втором уравнении

a22 = ur2.coeff(sp.diff(om, t), 1)  # Коэффициент при dom/dt во втором уравнении

b1 = (

    -(ur1.coeff(sp.diff(V, t), 0))

    .coeff(sp.diff(om, t), 0)

    .subs([(sp.diff(s, t), V), (sp.diff(alpha, t), om)])  # Свободный член в первом уравнении

)

b2 = (

    -(ur2.coeff(sp.diff(V, t), 0))

    .coeff(sp.diff(om, t), 0)

    .subs([(sp.diff(s, t), V), (sp.diff(alpha, t), om)])  # Свободный член во втором уравнении

)

detA = a11 \* a22 - a12 \* a21  # Определитель матрицы системы

detA1 = b1 \* a22 - b2 \* a21  # Определитель подматрицы для dV/dt

detA2 = a11 \* b2 - b1 \* a21  # Определитель подматрицы для dom/dt

dVdt = detA1 / detA  # Ускорение призмы (d^2s/dt^2)

domdt = detA2 / detA  # Угловое ускорение балки (d^2(alpha)/dt^2)

countOfFrames = 200  # Количество кадров для анимации

# Численное решение системы

T = np.linspace(0, 12, countOfFrames)  # Временной интервал

fV = sp.lambdify([t, s, alpha, V, om], dVdt, "numpy")  # Функция для вычисления dV/dt

fOm = sp.lambdify([t, s, alpha, V, om], domdt, "numpy")  # Функция для вычисления dom/dt

y0 = [1, 3.14/9, 0, 0]  # Начальные условия: s(0), alpha(0), V(0), om(0)

sol = odeint(formY, y0, T, args=(fV, fOm))  # Решение системы дифференциальных уравнений

# Вычисление координат

XsprS = sp.lambdify(s, s + 2)  # Положение пружины вдоль оси X

xASPhi = sp.lambdify([s, alpha], XsprS(s) - r / 2 \* sp.sin(alpha))  # Координата X балки

yASPhi = sp.lambdify([s, alpha], 0.9 - r / 2 \* sp.cos(alpha))  # Координата Y балки

XC = XsprS(sol[:, 0])  # Центр призмы

Alpha = sol[:, 1]  # Угол наклона балки

XA = xASPhi(sol[:, 0], sol[:, 1])  # Координата X точки на балке

YA = yASPhi(sol[:, 0], sol[:, 1])  # Координата Y точки на балке

# Визуализация

fig = plt.figure(figsize=(17, 8))

ax1 = fig.add\_subplot(1, 2, 1)

ax1.axis("equal")

# Построение осей

(liney,) = ax1.plot([0, 0], [0, 5], "black")  # Вертикальная ось

(linex,) = ax1.plot([0, 5], [0, 0], "black")  # Горизонтальная ось

ArrowX = np.array([-0.2, 0, -0.2])  # Координаты стрелки оси X

ArrowY = np.array([0.1, 0, -0.1])  # Координаты стрелки оси Y

(ArrowOY,) = ax1.plot(ArrowY, ArrowX + 5, "black")  # Стрелка оси Y

(ArrowOX,) = ax1.plot(ArrowX + 5, ArrowY, "black")  # Стрелка оси X

# Построение окружности

(P,) = ax1.plot(XC[0], R, marker="o", color="black")  # Центр окружности призмы

Phi = np.linspace(0, 2 \* math.pi, 200)  # Углы для параметризации окружности

(Circ,) = ax1.plot(XC[0] + R \* np.cos(Phi), R + R \* np.sin(Phi), "black")  # Окружность призмы

# Построение балки-маятника

Mayatnik = ax1.plot(

    XA[0] + r / 2 \* np.cos(Phi), YA[0] + r \* np.sin(Phi), color="black"

)[0]  # Балка, движущаяся относительно призмы

# Построение графиков зависимостей

ax2 = fig.add\_subplot(4, 2, 2)

ax2.plot(T, sol[:, 0])  # График смещения s(t)

ax2.set\_title("Смещение $s(t)$")

ax2.set\_xlabel("Время, t")

ax2.set\_ylabel("Смещение, $s(t)$")

ax3 = fig.add\_subplot(4, 2, 4)

ax3.plot(T, sol[:, 1], color="orange")  # График угла phi(t)

ax3.set\_title("Угол $\\varphi(t)$")

ax3.set\_xlabel("Время, t")

ax3.set\_ylabel("Угол, $\\varphi(t)$")

ax4 = fig.add\_subplot(4, 2, 6)

Rx = sol[:, 0] + R \* np.cos(sol[:, 1])  # Координата R\_x(t)

ax4.plot(T, Rx, color="green")

ax4.set\_title("Координата $R\_x(t)$")

ax4.set\_xlabel("Время, t")

ax4.set\_ylabel("Координата, $R\_x(t)$")

ax5 = fig.add\_subplot(4, 2, 8)

Ry = R \* np.sin(sol[:, 1])  # Координата R\_y(t)

ax5.plot(T, Ry, color="red")

ax5.set\_title("Координата $R\_y(t)$")

ax5.set\_xlabel("Время, t")

ax5.set\_ylabel("Координата, $R\_y(t)$")

plt.subplots\_adjust(hspace=0.9)  # Коррекция пространства между графиками

# Параметры пружины

spring\_segments = 20  # Количество сегментов пружины

spring\_height = 0.3  # Высота каждого сегмента

# Создаём линию для пружины

spring\_line, = ax1.plot([], [], color="purple", linewidth=2)  # Фиолетовая пружина

def create\_spring(start\_x, end\_x, start\_y, end\_y, segments, height):

    spring\_x = np.linspace(start\_x, end\_x, segments \* 2)

    spring\_y = np.zeros\_like(spring\_x)

    for i in range(len(spring\_x)):

        if i % 2 == 0:

            spring\_y[i] = start\_y + (i / (segments \* 2)) \* (end\_y - start\_y) - height

        else:

            spring\_y[i] = start\_y + (i / (segments \* 2)) \* (end\_y - start\_y) + height

    return spring\_x, spring\_y

# Функция анимации

def anima(i):

    NewX = []

    NewY = []

    # Обновление данных окружности

    P.set\_data([XC[i]], [R])  # Центр призмы

    Circ.set\_data(XC[i] + R \* np.cos(Phi), R + R \* np.sin(Phi))  # Окружность призмы

    spring\_x, spring\_y = create\_spring(0, XC[i], R, R, spring\_segments, spring\_height)

    spring\_line.set\_data(spring\_x, spring\_y)

    # Обновление балки-маятника

    for phi in Phi:

        newx, newy = Rot(

            XC[i] + 0.15 \* math.cos(phi),

            0.1 + 0.3 \* math.sin(phi) + 0.6,

            Alpha[i],

            XC[i],

            R,

        )

        NewX.append(newx)

        NewY.append(newy)

    Mayatnik.set\_data(NewX, NewY)

    # Возврат изменяемых объектов

    return Circ, P, spring\_line, Mayatnik

# Создание анимации

anim = FuncAnimation(fig, anima, frames=countOfFrames, interval=60, blit=True)

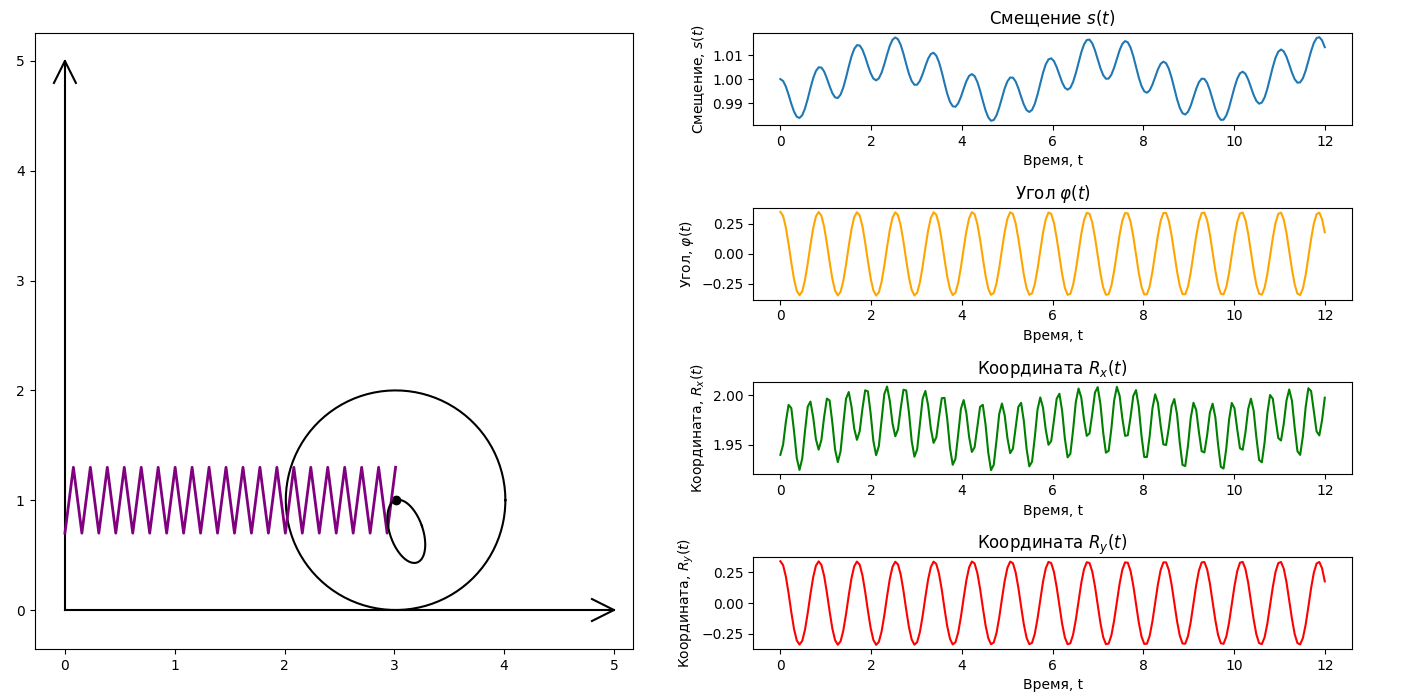
plt.show()

**Результат работы:**

**Случай 1: Исходные параметры**

* **m1 = 10, m2 = 5**
* **R = 1, r = 0.1**
* **c = 35, M₀ = 0**
* **Начальные условия: s(0) = 1, α(0) = π/9, ds/dt(0) = 0, dα/dt(0) = 0**

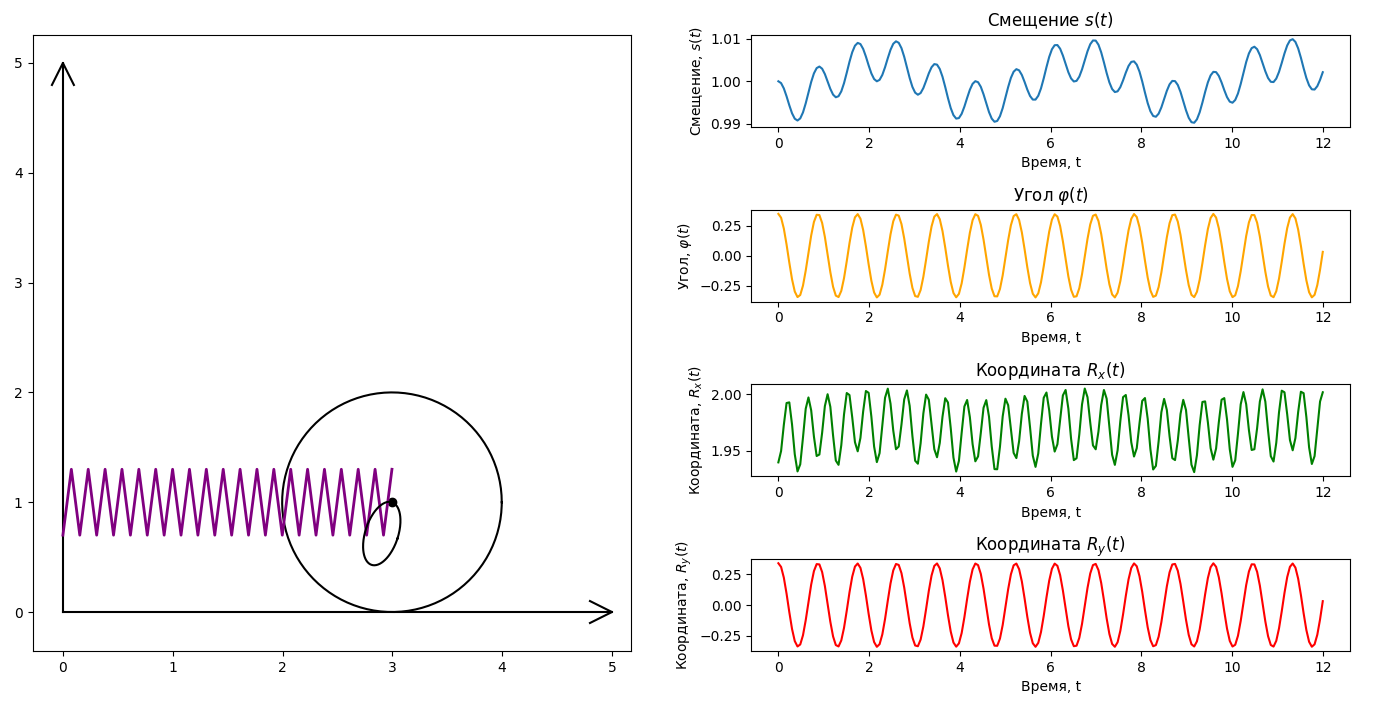
**Описание:** В данном случае были использованы исходные параметры системы. Ожидается плавное движение без значительных изменений амплитуды колебаний. Колебания сохраняют устойчивый характер, а система демонстрирует равномерное распределение энергии между поступательным и вращательным движением.

****

### Случай 2: Уменьшение массы балки (m₂) в 2 раза

* **m1 = 10, m2 = 2.5**
* **R = 1, r = 0.1**
* **c = 35, M₀ = 0**
* **Начальные условия: s(0) = 1, α(0) = π/9, ds/dt(0) = 0, dα/dt(0) = 0**

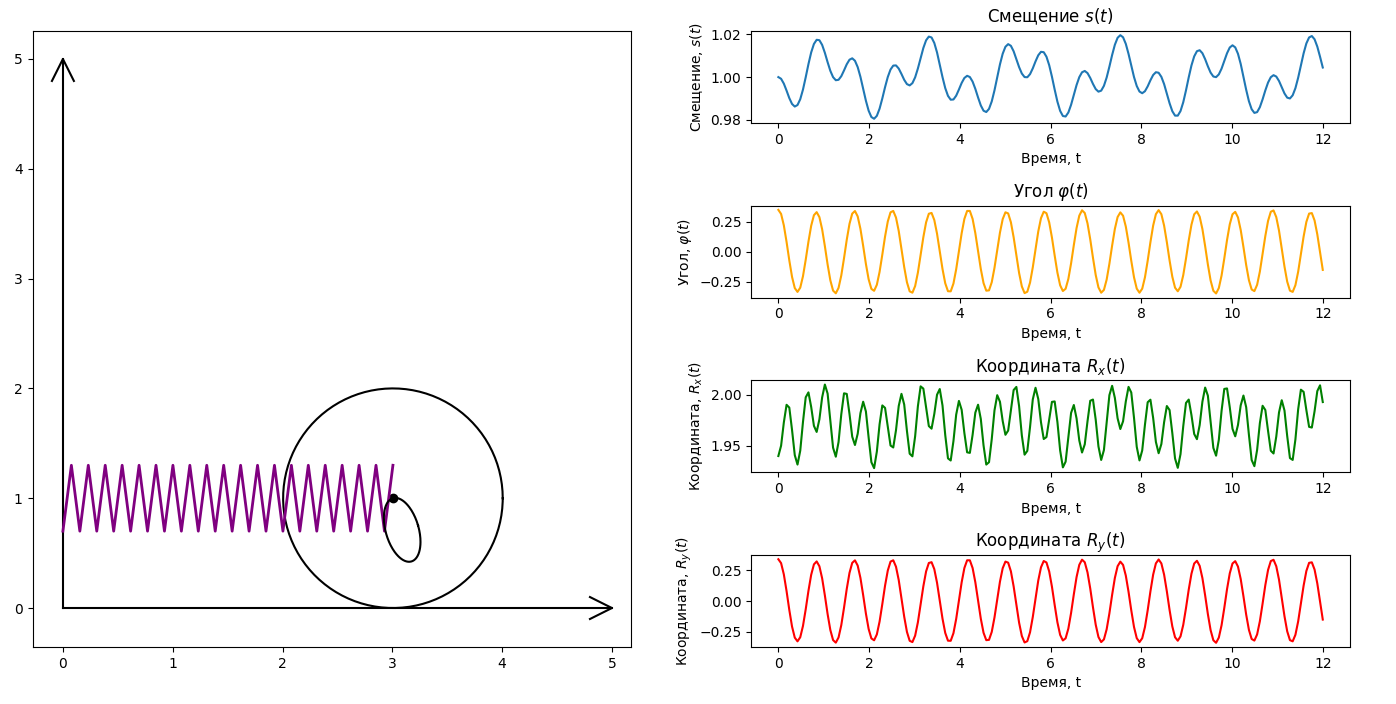
**Описание:** Уменьшение массы балки делает систему более чувствительной к внешним воздействиям. Это приводит к увеличению амплитуды колебаний и большей динамичности системы. Балка начинает сильнее реагировать на малейшие изменения начальных условий.

****

### Случай 3: Увеличение жесткости пружины в 5 раз

* **m1 = 10, m2 = 5**
* **R = 1, r = 0.1**
* **c = 175, M₀ = 0**
* **Начальные условия: s(0) = 1, α(0) = π/9, ds/dt(0) = 0, dα/dt(0) = 0**

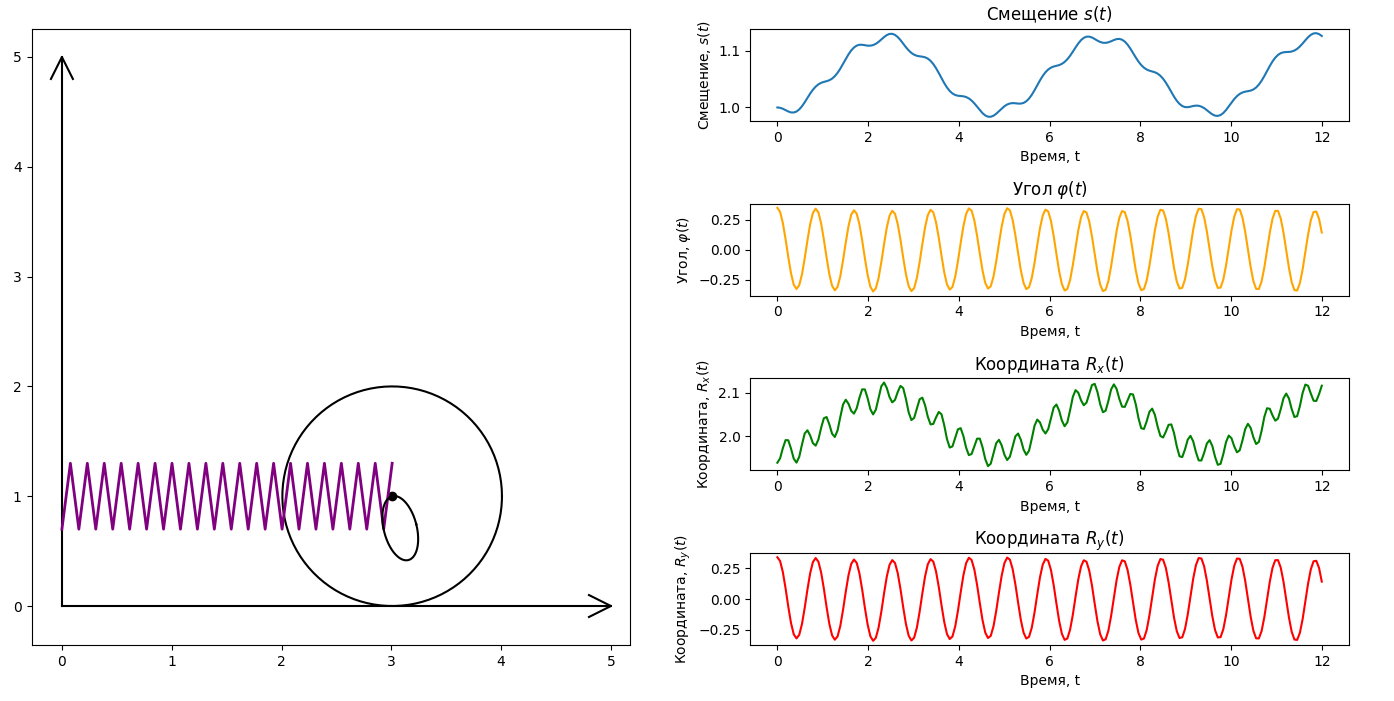
**Описание:** При увеличении жесткости пружины система становится более устойчивой. Амплитуда колебаний уменьшается, а движение быстрее затухает. Система быстрее возвращается в положение равновесия, что свидетельствует о снижении влияния вращательного движения балки.



### Случай 4: Увеличение амплитуды момента (M₀) в 2 раза

* **m1 = 10, m2 = 5**
* **R = 1, r = 0.1**
* **c = 35, M₀ = 2**
* **Начальные условия: s(0) = 1, α(0) = π/9, ds/dt(0) = 0, dα/dt(0) = 0**

**Описание:** Увеличение амплитуды момента приводит к значительному усилению вращательных эффектов в системе. Балка демонстрирует более высокую угловую скорость, а общая динамика системы становится сложнее. Система демонстрирует резкие изменения в траектории движения.

****

**Вывод:**

В ходе выполнения лабораторной работы я смоделировал движение системы, состоящей из призмы, пружины и балки-маятника, используя уравнения Лагранжа второго рода. Были построены графики зависимостей смещения, углового перемещения и координат центра массы от времени, а также анимация движения системы. Благодаря выполнению работы я углубил свои знания в области численного решения систем дифференциальных уравнений и научился визуализировать их результаты с помощью анимации в библиотеке matplotlib.