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 №24**

Выполнил(а) студент группы М8О-201Б-23

Фокин Л.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

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

подпись, дата

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

Москва, 2024

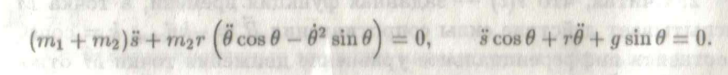
**Вариант №24**

**Задание:**

Реализовать численное интегрирование системы дифференциальных уравнений, описывающих движение системы с двумя степенями свободы, с использованием Python. Создать анимацию, демонстрирующую динамику системы, а также построить графики, отражающие законы движения и значения заданных реакций для различных конфигураций системы.

**Задание системы:**

Невесомая трубка, выгнутая в форме круглого кольца радиуса r, закреплена на платформе, которая имеет массу m₁ и может скользить без трения по горизонтальной плоскости (рис. 24). В трубке движется без трения точечный груз m₂ массы. Дифференциальные уравнения движения системы имеют вид:



**Код программы:**

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from scipy.integrate import solve\_ivp

# Функция для построения трапеции

def trapezoid(*x0*, *y0*, *width*=10, *height*=17.5):

    px = [*x0* - *width*, *x0* - *width* / 3, *x0* + *width* / 3, *x0* + *width*, *x0* - *width*]

    py = [*y0* - *height* / 2, *y0* + *height* / 2, *y0* + *height* / 2, *y0* - *height* / 2, *y0* - *height* / 2]

    return px, py

# Функция для решения уравнений движения

def equations(*t*, *y*, *m1*, *m2*, *r*, *g*):

    s, s\_dot, theta, theta\_dot = *y*

    a11 = *m1* + *m2*

    a12 = *m2* \* *r* \* np.cos(theta)

    a21 = np.cos(theta)

    a22 = *r*

    b1 = *m2* \* *r* \* theta\_dot\*\*2 \* np.sin(theta)

    b2 = -*g* \* np.sin(theta)

    det = a11 \* a22 - a12 \* a21

    s\_ddot = (a22 \* b1 - a12 \* b2) / det

    theta\_ddot = (-a21 \* b1 + a11 \* b2) / det

    return [s\_dot, s\_ddot, theta\_dot, theta\_ddot]

# Ввод параметров

m1 = float(input("Введите m1: "))

m2 = float(input("Введите m2: "))

r = float(input("Введите радиус r: "))

phi = float(input("Введите начальный угол phi (в радианах): "))

g = 9.81

# Временные параметры

t\_span = (0, 20)  # от 0 до 20 секунд

t\_eval = np.linspace(0, 20, 1000)

# Начальные условия [s, s\_dot, theta, theta\_dot]

initial\_conditions = [0, 0, phi, 0]

# Решение уравнений с помощью solve\_ivp

solution = solve\_ivp(equations, t\_span, initial\_conditions, *t\_eval*=t\_eval, *args*=(m1, m2, r, g))

s = solution.y[0]

s\_dot = solution.y[1]

theta = solution.y[2]

theta\_dot = solution.y[3]

# Координаты центра и точки

x\_center = s + 0.8

y\_center = np.ones\_like(t\_eval) \* 7.5

x\_A = x\_center - r \* np.sin(theta)

y\_A = y\_center + r \* np.cos(theta)

# Настройка фигуры и осей для анимации

fig, ax = plt.subplots(*figsize*=(10, 6))

ax.axis('equal')

ax.set\_xlim(x\_center.min() - 5, x\_center.max() + 5)

ax.set\_ylim(y\_center.min() - 10, y\_center.max() + 10)

# Отображаемые элементы

trap\_x, trap\_y = trapezoid(x\_center[0], y\_center[0])

trap, = ax.plot(trap\_x, trap\_y, 'r')  # Трапеция

radius\_line, = ax.plot([x\_center[0], x\_A[0]], [y\_center[0], y\_A[0]], 'k')  # Радиус

theta\_points = np.linspace(0, 2 \* np.pi, 50)

point\_circle, = ax.plot([], [], 'b')  # Точка на радиусе

# Анимация

def update(*frame*):

    trap\_x, trap\_y = trapezoid(x\_center[*frame*], y\_center[*frame*])

    trap.set\_data(trap\_x, trap\_y)

    radius\_line.set\_data([x\_center[*frame*], x\_A[*frame*]], [y\_center[*frame*], y\_A[*frame*]])

    point\_circle.set\_data(x\_A[*frame*] + 0.2 \* np.cos(theta\_points),

                          y\_A[*frame*] + 0.2 \* np.sin(theta\_points))

    return trap, radius\_line, point\_circle

# Создаем анимацию

ani = FuncAnimation(fig, update, *frames*=len(t\_eval), *interval*=20, *blit*=True)

# Создание подграфиков для показателей

fig2, axs = plt.subplots(4, 1, *figsize*=(10, 10))

axs[0].plot(t\_eval, s, *label*='s (перемещение)')

axs[0].set\_title('Перемещение от времени')

axs[0].set\_xlabel('Время (с)')

axs[0].set\_ylabel('s (м)')

axs[0].grid()

axs[0].legend()

axs[1].plot(t\_eval, s\_dot, *label*='s\_dot (скорость)', *color*='orange')

axs[1].set\_title('Скорость от времени')

axs[1].set\_xlabel('Время (с)')

axs[1].set\_ylabel('s\_dot (м/с)')

axs[1].grid()

axs[1].legend()

axs[2].plot(t\_eval, theta, *label*='theta (угол)', *color*='green')

axs[2].set\_title('Угол от времени')

axs[2].set\_xlabel('Время (с)')

axs[2].set\_ylabel('theta (рад)')

axs[2].grid()

axs[2].legend()

axs[3].plot(t\_eval, theta\_dot, *label*='theta\_dot (угловая скорость)', *color*='red')

axs[3].set\_title('Угловая скорость от времени')

axs[3].set\_xlabel('Время (с)')

axs[3].set\_ylabel('theta\_dot (рад/с)')

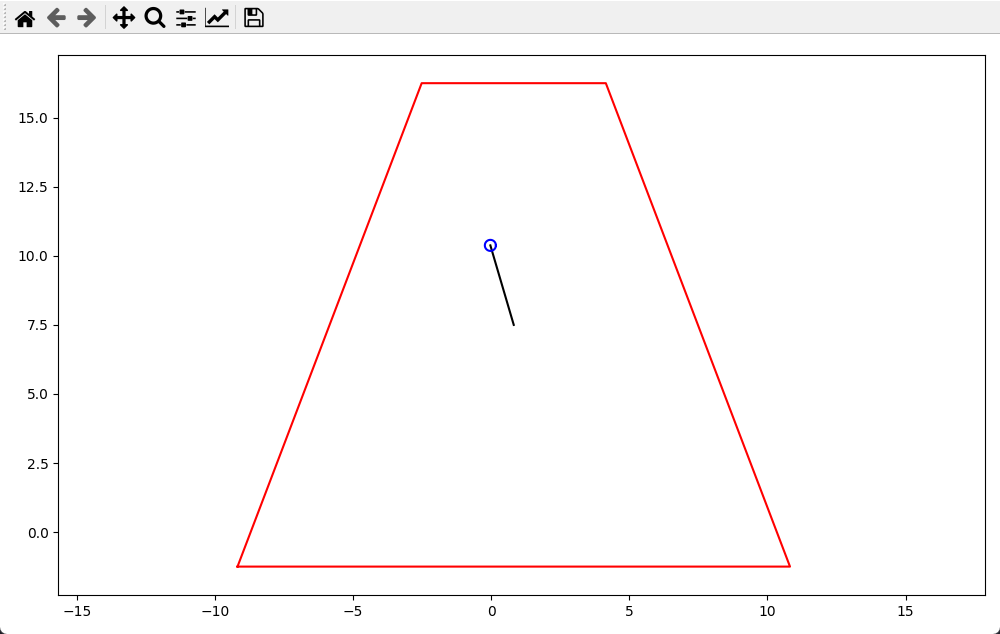
axs[3].grid()

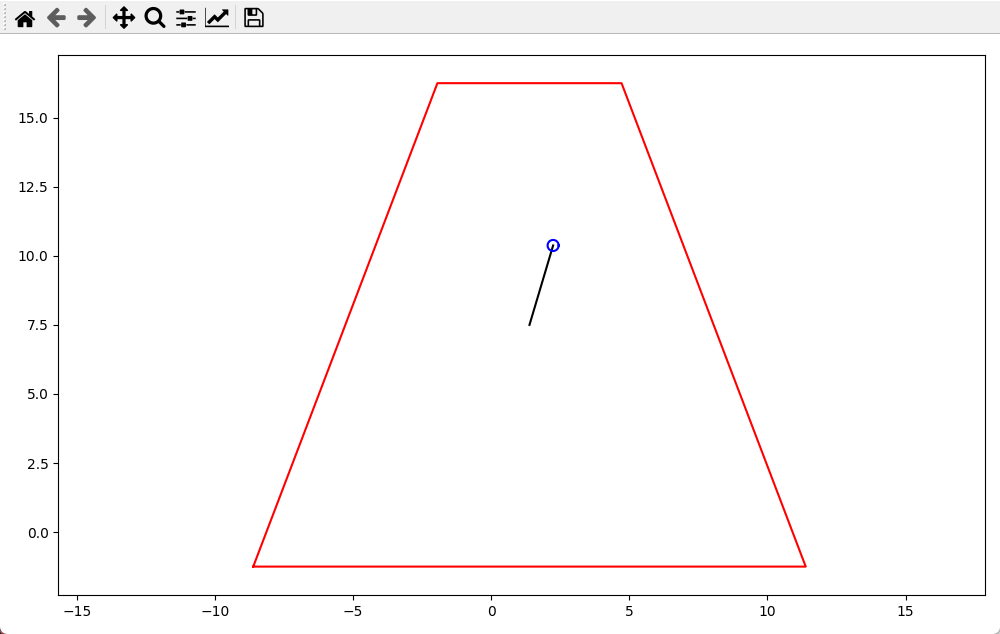
axs[3].legend()

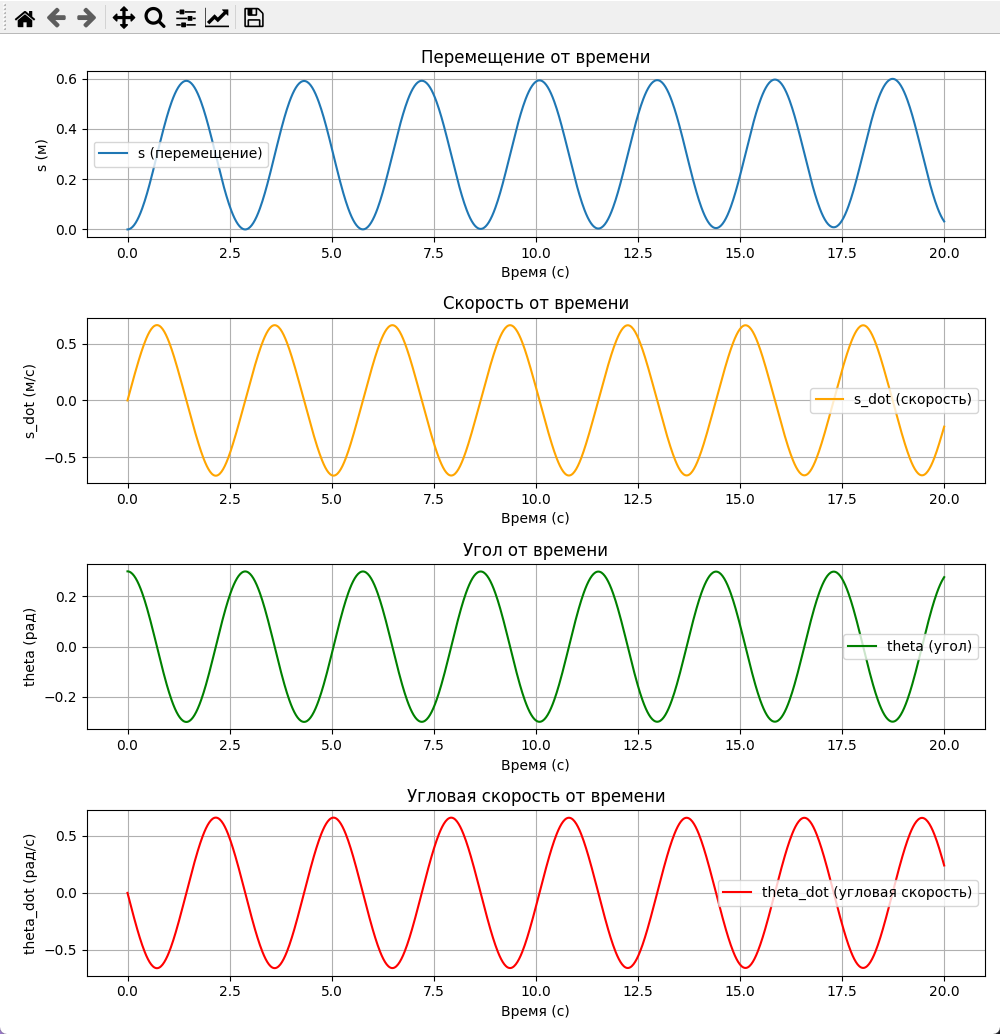
plt.tight\_layout()

plt.show()

**Результат работы при :**







**Вывод:** в ходе выполнения лабораторной работы была реализована численная модель системы с двумя степенями свободы, состоящей из платформы массой и точечного груза ​, движущегося по кольцевой трубке радиуса . С использованием Python и библиотеки *scipy.integrate.solve\_ivp* было проведено численное интегрирование системы дифференциальных уравнений, описывающих движение системы. Была создана анимация, демонстрирующая динамику системы, а также построены графики зависимости перемещения, скорости, угла и угловой скорости от времени. Результаты моделирования показали, что система демонстрирует сложное поведение, зависящее от начальных условий и параметров системы.Начало формы