ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ ДВИЖЕНИЯ МЕХАНИЧЕСКОЙ СИСТЕМЫ»**

**ПО ДИСЦИПЛИНЕ «ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ ДИНАМИЧЕСКИХ СИСТЕМ»**

**ВАРИАНТ ЗАДАНИЯ № 1**

Выполнил(а) студент группы М8О-203Б-23

Арусланов К.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

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

подпись, дата

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

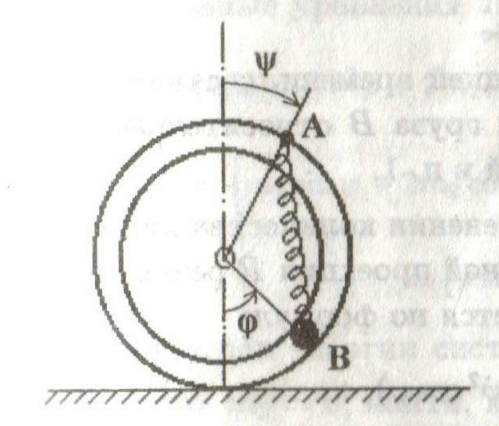
Москва, 2024

**Вариант № 1**

**Задание:**

Реализовать анимацию движения механической системы.

**Механическая система:**



**Текст программы**

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

import math

# --- Параметры системы ---

R = 2.0        # Радиус колеса (м)

r = 1.0        # Радиус внутренней трубки (м)

c = 20.0       # Жесткость пружины (Н/м)

m = 1.0        # Масса шарика (кг)

g = 9.81       # Ускорение свободного падения (м/с^2)

v = 1.0        # Скорость движения колеса (м/с)

phi\_eq = 0.0       # Равновесное положение угла шарика (рад)

phi0 = np.pi / 6   # Начальное отклонение угла (рад)

phi\_dot0 = 0.0     # Начальная угловая скорость (рад/с)

psi0 = 0.0         # Начальный угол точки A

ball\_radius = 0.2  # Радиус шарика (м)

# --- Время и дискретизация ---

Steps = 1001               # Количество кадров

t\_fin = 20                 # Конечное время (с)

t = np.linspace(0, t\_fin, Steps)  # Массив времени

dt = t[1] - t[0]           # Шаг по времени

# --- Связь между угловой и линейной скоростью ---

X\_O = v \* t                # Горизонтальное положение центра колеса (м)

Y\_O = R                    # Вертикальное положение центра колеса (м)

# Угловая скорость вращения колеса

omega\_wheel = v / R        # Угловая скорость колеса (рад/с)

# --- Угловое положение точек ---

psi = omega\_wheel \* t + psi0  # Угол точки A относительно вертикали

# --- Решение уравнения движения шарика ---

phi = np.zeros(Steps)      # Угловое положение шарика

phi\_dot = np.zeros(Steps)  # Угловая скорость шарика

phi[0] = phi0

phi\_dot[0] = phi\_dot0

omega\_0 = np.sqrt(c / m)   # Собственная угловая частота пружинного маятника

for i in range(1, Steps):

    phi\_ddot = -omega\_0\*\*2 \* (phi[i-1] - phi\_eq)  # Ускорение

    phi\_dot[i] = phi\_dot[i-1] + phi\_ddot \* dt     # Скорость

    phi[i] = phi[i-1] + phi\_dot[i] \* dt           # Положение

# --- Координаты точки A ---

X\_A = X\_O + R \* np.sin(psi)

Y\_A = Y\_O - R \* np.cos(psi)

# --- Координаты шарика B (исправлено для точного положения между окружностями) ---

center\_radius = R - ball\_radius  # Центр шарика учитывает внешний радиус минус его радиус

X\_B = X\_O + center\_radius \* np.sin(phi)

Y\_B = Y\_O - center\_radius \* np.cos(phi)

# --- Параметры пружины ---

spring\_segments = 20       # Количество сегментов пружины

# --- Координаты внутренней трубки ---

inner\_tube\_radius = R - ball\_radius \* 2  # Внутренняя трубка корректируется относительно радиуса шарика

X\_Tube = inner\_tube\_radius \* np.cos(np.linspace(0, 2 \* np.pi, 100))

Y\_Tube = inner\_tube\_radius \* np.sin(np.linspace(0, 2 \* np.pi, 100))

# Функция для создания координат пружины

def create\_spring\_segments(x\_start, y\_start, x\_end, y\_end, segments):

    X\_spring = np.zeros(segments)

    Y\_spring = np.zeros(segments)

    for i in range(segments):

        fraction = i / (segments - 1)

        X\_spring[i] = x\_start + fraction \* (x\_end - x\_start)

        Y\_spring[i] = y\_start + fraction \* (y\_end - y\_start)

        if 0 < i < segments - 1:

            if i % 2 == 0:

                X\_spring[i] += 0.1 \* (y\_end - y\_start)

            else:

                X\_spring[i] -= 0.1 \* (y\_end - y\_start)

    return X\_spring, Y\_spring

# --- Настройка графика ---

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

ax.set\_xlim(X\_O.min() - R - 1, X\_O.max() + R + 1)

ax.set\_ylim(-R - 1, Y\_O + R + 1)

ax.set\_aspect('equal')

ax.set\_xlabel('Горизонтальная позиция (м)')

ax.set\_ylabel('Вертикальная позиция (м)')

ax.set\_title('Анимация катящегося колеса с шариком и пружиной')

# --- Рисование статичных элементов ---

# Рисуем горизонтальную направляющую

ax.plot([X\_O.min() - R - 1, X\_O.max() + R + 1], [0, 0], 'k-', linewidth=2)

# --- Рисование динамических элементов ---

# Рисуем колесо

wheel\_outline = 100

psi\_circle = np.linspace(0, 2 \* np.pi, wheel\_outline)

X\_Wheel = R \* np.cos(psi\_circle)

Y\_Wheel = R \* np.sin(psi\_circle)

wheel, = ax.plot([], [], 'b-', linewidth=2)

# Рисуем внутреннюю трубку

tube, = ax.plot([], [], 'gray', linestyle='--', linewidth=1)

# Рисуем пружину

spring, = ax.plot([], [], 'r-', linewidth=2)

# Рисуем шарик

ball = plt.Circle((X\_B[0], Y\_B[0]), ball\_radius, color='g')

ax.add\_patch(ball)

# Рисуем радиус-векторы

radius\_vector\_A, = ax.plot([], [], 'g--', linewidth=1)

radius\_vector\_B, = ax.plot([], [], 'b--', linewidth=1)

# --- Функция инициализации ---

def init():

    wheel.set\_data([], [])

    tube.set\_data([], [])

    spring.set\_data([], [])

    ball.center = (X\_B[0], Y\_B[0])

    radius\_vector\_A.set\_data([], [])

    radius\_vector\_B.set\_data([], [])

    return wheel, tube, spring, ball, radius\_vector\_A, radius\_vector\_B

# --- Функция анимации ---

def anima(i):

    # Обновление положения колеса

    current\_X\_Wheel = X\_Wheel + X\_O[i]

    current\_Y\_Wheel = Y\_Wheel + Y\_O

    wheel.set\_data(current\_X\_Wheel, current\_Y\_Wheel)

    # Обновление положения внутренней трубки

    current\_X\_Tube = X\_Tube + X\_O[i]

    current\_Y\_Tube = Y\_Tube + Y\_O

    tube.set\_data(current\_X\_Tube, current\_Y\_Tube)

    # Обновление положения пружины

    X\_spring, Y\_spring = create\_spring\_segments(X\_A[i], Y\_A[i], X\_B[i], Y\_B[i], spring\_segments)

    spring.set\_data(X\_spring, Y\_spring)

    # Обновление положения шарика

    ball.center = (X\_B[i], Y\_B[i])

    # Обновление радиус-векторов

    radius\_vector\_A.set\_data([X\_O[i], X\_A[i]], [Y\_O, Y\_A[i]])

    radius\_vector\_B.set\_data([X\_O[i], X\_B[i]], [Y\_O, Y\_B[i]])

    return wheel, tube, spring, ball, radius\_vector\_A, radius\_vector\_B

# --- Создание анимации ---

anim = FuncAnimation(fig, anima, init\_func=init,

                     frames=Steps, interval=40, blit=True, repeat=False)

plt.show()

**Результат работы программы:**

|  |  |
| --- | --- |
|  |  |
|  |  |

**Вывод:**

В ходе выполнения лабораторной работы была разработана программа на языке Python, которая осуществляет анимацию движения системы. Мои навыки программирования на Python улучшились.