ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ СИСТЕМЫ»**

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

**ВАРИАНТ ЗАДАНИЯ №12**

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

Ермеков Георгий Александрович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

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

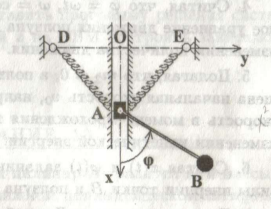
подпись, дата

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

Москва, 2024

**Задание:** построить анимацию движения системы с помощью Python.

**Механическая система:**

**Текст пр****ограммы:**

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from matplotlib.gridspec import GridSpec

a = 1.0  # расстояние между центрами пружин D и E

ab\_l = 2.0  # длина стержня AB

g = 9.81  # ускорение свободного падения

m2 = 1  # масса груза

slider\_height = 1.4

slider\_y = -0.5  # высота ползуна

# точки

point\_D = [a, 0]

point\_O = [0, 0]

point\_E = [-a, 0]

wall\_D = [a + 0.05, 0]

wall\_E = [-a - 0.05, 0]

fi0 = np.pi / 4  # начальный угол от вертикали

t\_max = 5.0

dt = 0.08

t\_values = np.arange(0, t\_max, dt)

fig = plt.figure()

gs = GridSpec(5, 1, figure=fig)

ax = fig.add\_subplot(gs[:5, :])

(line,) = ax.plot([], [], "c-")  # стержень AB

(ball,) = ax.plot([], [], "bo", markersize=10)  # груз B

(slider,) = ax.plot([], [], "go", markersize=15)  # ползун A

(point\_d,) = ax.plot([], [], "co", markersize=3)  # точка D

(point\_o,) = ax.plot([], [], "mo", markersize=3)  # точка O

(point\_e,) = ax.plot([], [], "co", markersize=3)  # точка E

(wall\_d,) = ax.plot([], [], "bo", markersize=8)  # стенка D

(wall\_e,) = ax.plot([], [], "bo", markersize=8)  # стенка E

(spring\_DA,) = ax.plot([], [], "g", lw=2)  # пружина DA

(spring\_EA,) = ax.plot([], [], "g", lw=2)  # пружина EA

ax.set\_xlim(-10, 10)

ax.set\_ylim(-3, 1)

ax.axvline(0, linestyle="--", color="k")  # пунктиры

ax.axhline(point\_D[1], linestyle="--", color="k")

point\_d.set\_data([point\_D[0]], [point\_D[1]])

point\_o.set\_data([point\_O[0]], [point\_O[1]])

point\_e.set\_data([point\_E[0]], [point\_E[1]])

wall\_d.set\_data([wall\_D[0]], [wall\_D[1]])

wall\_e.set\_data([wall\_E[0]], [wall\_E[1]])

# подписи точек

ax.text(point\_D[0], point\_D[1], "D", ha="right", va="bottom")

ax.text(point\_O[0], point\_O[1], "O", ha="right", va="bottom")

ax.text(point\_E[0], point\_E[1], "E", ha="right", va="bottom")

# буквы A и B

text\_A = ax.text(0, slider\_y, "A", ha="right", va="bottom")

text\_B = ax.text(ab\_l \* np.sin(fi0), slider\_y - ab\_l \* np.cos(fi0), "B", ha="right", va="bottom")

# рассчет координат пружин

def create\_springs(start, end, num\_segments=8, amplitude=0.1):

    x\_vals = np.linspace(start[0], end[0], num\_segments)

    y\_vals = np.linspace(start[1], end[1] - 0.2, num\_segments)

    dist = np.sqrt((end[0] - start[0]) \*\* 2 + (end[1] - start[1]) \*\* 2)

    factor = amplitude \* (1 + 0.5 \* dist)

    for i in range(1, num\_segments, 2):

        y\_vals[i] += factor

    return x\_vals, y\_vals

# координаты ломаных пружины

spring\_DA\_x, spring\_DA\_y = create\_springs(point\_D, [0, slider\_y])

spring\_EA\_x, spring\_EA\_y = create\_springs(point\_E, [0, slider\_y])

# задаем пружины

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

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

def animate(i):

    global slider\_y

    t = i \* dt

    # меняем положение ползуна

    slider\_y = -ab\_l \* np.cos(fi0 \* np.cos(np.sqrt(g / ab\_l) \* t)) + slider\_height

    # уравнение маятника

    fi = fi0 \* np.cos(np.sqrt(g / ab\_l) \* t)

    x\_values = [0, ab\_l \* np.sin(fi)]

    y\_values = [slider\_y, slider\_y - ab\_l \* np.cos(fi)]

    line.set\_data(x\_values, y\_values)

    # положение груза B

    x\_ball = ab\_l \* np.sin(fi)

    y\_ball = slider\_y - ab\_l \* np.cos(fi)

    ball.set\_data([x\_ball], [y\_ball])

    # обновляем пружины

    spring\_DA\_x, spring\_DA\_y = create\_springs(point\_D, [0, slider\_y])

    spring\_EA\_x, spring\_EA\_y = create\_springs(point\_E, [0, slider\_y])

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

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

    # положение ползуна

    slider.set\_data([0], [slider\_y - 0.1])

    # обновляем координаты букв

    text\_A.set\_position((0, slider\_y))

    text\_B.set\_position((x\_ball, y\_ball))

    return line, ball, slider, spring\_EA, spring\_DA, point\_d, point\_o, point\_e, text\_A, text\_B, wall\_d, wall\_e

ani = FuncAnimation(fig, animate, frames=len(t\_values), blit=True, interval=100)

plt.show()

**Пример работы:**

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

**Вывод:** В результате работы была создана программа, которая отражает работу данной установки в реальном времени с помощью средств библиотеки matplotlib.