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 № 18**

Выполнил(а) студент группы М8О-203Б-22

Ибрагимов Роман Рифхатович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

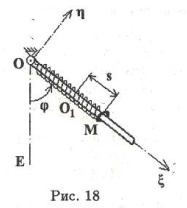
Авдюшкин А.Н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

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

Москва, 2023

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



Код программы

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

def spring(start, end, nodes, width):

nodes = max(int(nodes), 1)

start, end = np.array(start).reshape((2,)), np.array(end).reshape((2,))

if (start == end).all():

return start[0], start[1]

length = np.linalg.norm(np.subtract(end, start))

u\_t = np.subtract(end, start) / length

u\_n = np.array([[0, -1], [1, 0]]).dot(u\_t)

spring\_coords = np.zeros((2, nodes + 2))

spring\_coords[:,0], spring\_coords[:,-1] = start, end

normal\_dist = np.sqrt(max(0, width\*\*2 - (length\*\*2 / nodes\*\*2))) / 2

for i in range(1, nodes + 1):

spring\_coords[:,i] = (

start

+ ((length \* (2 \* i - 1) \* u\_t) / (2 \* nodes))

+ (normal\_dist \* (-1)\*\*i \* u\_n))

return spring\_coords[0,:], spring\_coords[1,:]

def anim(i):

pM.set\_data([M\_x[i], M\_y[i]])

OE.set\_data([O\_x, E\_x[i]], [O\_y, E\_y[i]])

Spr.set\_data(\*spring((O\_x, O\_y), (M\_x[i], M\_y[i]), 10, 0.3))

return

steps = 1000

fgr = plt.figure()

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

gr.axis('equal')

gr.set(xlim=[0, 5], ylim=[-1, 5])

gr.plot([0, 0, 4], [4, 0, 0], linewidth=3)

gr.plot([1.75, 2.25], [3, 3], 'black', linewidth=3)

t = np.linspace(-np.pi, 0, steps)

x0 = 2

y0 = 3

L = 1.5

O\_x = x0

O\_y = y0

E\_x = x0 + L \* np.cos(t)

E\_y = y0 + L \* np.sin(t)

L\_sping\_max = 1.2

L\_sping\_curr = L\_sping\_max \* np.sin(t)

M\_x = x0 - L\_sping\_curr \* np.cos(t)

M\_y = y0 - L\_sping\_curr \* np.sin(t)

pO = gr.plot(O\_x, O\_y)[0]

pE = gr.plot(E\_x[0], E\_y[0])[0]

pM = gr.plot(M\_x[0], M\_y[0], 'yellow', marker='o')[0]

OE = gr.plot([O\_x, E\_x[0]], [O\_y, E\_y[0]], 'grey')[0]

Spr = gr.plot(\*spring((O\_x, O\_y), (M\_x[0], M\_y[0]), 10, 0.3), 'red')[0]

an = FuncAnimation(fgr, anim, frames=steps, interval=1)

plt.show()

Результат выполнения программы

