ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ СИСТЕМЫ»**

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

**ВАРИАНТ ЗАДАНИЯ №3**

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

Бакач Анна Андреевна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

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

подпись, дата

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

Москва, 2024

**Вариант №3**

**Задание:**

Реализовать анимацию движения механической системы используя язык программирования Python.  
Условие:  


**Работа программы:**



**Код программы:**

import matplotlib.pyplot as plt

from matplotlib.patches import Circle, Polygon

from matplotlib.animation import FuncAnimation

from scipy.integrate import solve\_ivp

import numpy as np

def create\_wheels(axis1X, axis2X):

w = []

for i in [axis1X, axis2X]:

wheel\_front = Circle((0, 2), radius=0.8, fill=False, edgecolor='black')

wheel\_back = Circle((0 + 0.05, 2), radius=0.8, fill=False, edgecolor='black')

grf.add\_patch(wheel\_front)

grf.add\_patch(wheel\_back)

w += [wheel\_front, wheel\_back]

return w

def create\_axis(x, y):

mesh = [(x - 1, y), (x + 1, y), (x, y - 2)]

axis = Polygon(mesh, closed=True, fill=False, edgecolor='green')

grf.add\_patch(axis)

return [mesh, axis]

m1, m2, m3, m4 = 10, 3, 0.1, 1

g = 9.81

alpha = np.radians(15)

F0, omega = 3, 0.5

speed\_multiplier = 3.5

theta = np.arctan2(-3, 4)

t\_eval = np.linspace(0, 20, 500)

A = 2.0

omega = 0.5

x\_vals = A \* np.sin(omega \* t\_eval)

s\_vals =A \* np.sin(omega \* t\_eval)

fgr, grf = plt.subplots()

[wheel1, wheel2, wheel3, wheel4] = create\_wheels(0, 5)

plane\_mesh = [(-100, 1.2), (100, 1.2)]

plane = Polygon(plane\_mesh, closed=False, fill=False, edgecolor='black')

grf.add\_patch(plane)

[axis1\_mesh, axis1] = create\_axis(0, 4)

[axis2\_mesh, axis2] = create\_axis(5, 4)

pillar\_mesh = [(2.6, 8.4), (2.4, 8), (2.8, 8)]

pillar = Polygon(pillar\_mesh, closed=True, fill=False, edgecolor='yellow')

grf.add\_patch(pillar)

body\_mesh = [(-3, 4), (-3, 8), (3, 8), (7, 5), (7, 4)]

body = Polygon(body\_mesh, closed=True, fill=False, edgecolor='green')

grf.add\_patch(body)

little\_circle = Circle((2.6, 8.6), radius=0.4, fill=False, edgecolor='blue')

big\_circle = Circle((-1, 8.8), radius=0.8, fill=False, edgecolor='red')

grf.add\_patch(little\_circle)

grf.add\_patch(big\_circle)

cylinder = Polygon([(0, 0), (0.6, 0), (0.6, 0.45), (0, 0.45)], closed=True, fill=False, edgecolor='#E1592F')

grf.add\_patch(cylinder)

[line1] = grf.plot([0, 0], [0, 0], color='black')

[line2] = grf.plot([0, 0], [0, 0], color='black')

direction\_vector = np.array([-4, 3]) / 5

def rotate\_cylinder(base\_center, width, height, theta):

local\_mesh = np.array([

[-width / 2, 0],

[width / 2, 0],

[width / 2, height],

[-width / 2, height]

])

rotation\_matrix = np.array([

[np.cos(theta), -np.sin(theta)],

[np.sin(theta), np.cos(theta)]

])

rotated\_mesh = local\_mesh @ rotation\_matrix.T + base\_center

return rotated\_mesh

def move\_cart(dx, ds):

wheel1.center = (dx, 2)

wheel2.center = (dx + 5, 2)

wheel3.center = (dx + 0.1, 2)

wheel4.center = (dx + 0.1 + 5, 2)

body\_xy = np.array(body\_mesh) + [dx, 0]

body.set\_xy(body\_xy)

axis1\_xy = np.array(axis1\_mesh) + [dx, 0]

axis1.set\_xy(axis1\_xy)

axis2\_xy = np.array(axis2\_mesh) + [dx, 0]

axis2.set\_xy(axis2\_xy)

pillar\_xy = np.array(pillar\_mesh) + [dx, 0]

pillar.set\_xy(pillar\_xy)

little\_circle.center = (2.6 + dx, 8.4)

def update\_rope(cylinder\_mesh):

cylinder\_center = cylinder\_mesh.mean(axis=0)

line1.set\_xdata([little\_circle.center[0], big\_circle.center[0]])

line1.set\_ydata([little\_circle.center[1] + 0.4, big\_circle.center[1]])

line2.set\_xdata([little\_circle.center[0], cylinder\_center[0]])

line2.set\_ydata([little\_circle.center[1] + 0.4, cylinder\_center[1]])

def anim(i):

dx = x\_vals[i]

ds = s\_vals[i]

move\_cart(dx, ds)

horizontal\_offset = -s\_vals[i]

big\_circle.center = (-0.8 + dx + horizontal\_offset, 8.8)

rect\_base\_center = np.array([3 + dx, 8]) + (ds - 3) \* direction\_vector

cylinder\_mesh\_transformed = rotate\_cylinder(rect\_base\_center, 1.2, 0.6, theta)

cylinder.set\_xy(cylinder\_mesh\_transformed)

update\_rope(cylinder\_mesh\_transformed)

return wheel1, wheel2, body, axis1, axis2, pillar, little\_circle, big\_circle, cylinder

grf.set\_xlim(-5, 10)

grf.set\_ylim(0, 10)

grf.set\_aspect('equal')

ani = FuncAnimation(fgr, anim, frames=500, interval=50)

plt.show()

**Вывод:**

В процессе выполнения лабораторной работы была разработана анимация механической системы с применением библиотек numpy и matplotlib. Основные этапы включали задание параметров системы, вычисление координат точек и создание визуализации движения.