ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ ТОЧКИ»**

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

**ВАРИАНТ ЗАДАНИЯ №21**

Выполнил(а) студент группы  М80-201Б-22

Парфенов Михаил Максимович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Зав. каф. 802, Бардин Б.С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

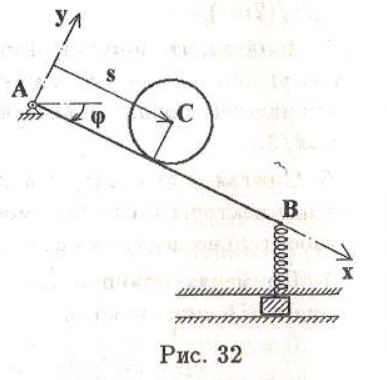
подпись, дата

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

Москва, 2023

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

**Механическая система:**



**Текст программы:**

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from scipy.integrate import odeint

step = 1000

t = np.linspace(0, 10, step)

#перевод полярных координат в Декартовы

s = (np.cos(6\*t))\*np.cos(t + 0.2\*np.cos(3\*t))

phi = t + 0.2\*np.cos(3\*t)

l0 = 5 #длина недеформированной пружины в начальном состоянии с учетом phi\_

l = 10

r = 0.5

n = 13

h = 0.05

xA = 0 #точки крепления шарнира и балки

yA = 5

xB = l #точки крепления пружины и балки

yB = 0

xP = np.zeros(2\*n + 1)

yP = np.linspace(0, 1, 2\*n + 1)

ss = 0

# делаем пружину пружиной

for i in range(2\*n+1):

xP[i] = h\*np.sin(ss)

ss += np.pi/2

yB = l0 - l \* np.sin(phi)

#prizuna

fig = plt.figure(figsize=[15,7])

ax = fig.add\_subplot(1,1,1)

ax.axis('equal')

ax.set(xlim=[-3,12], ylim=[-3,12])

Pruzzhina = ax.plot(xP + xB, yP\*(yB[0] + l0), color='black')[0]

#enviroment

X\_Ground = [0, 0, 12]

Y\_Ground = [9, 0, 0]

ax.plot(X\_Ground,Y\_Ground,color='black',linewidth=3)

ax.plot([0], [5], '-ro', markersize=10, color='blue')

#балка

Stick = ax.plot([xA, xB], [yA, yB[0]], linewidth=7, color='brown')[0]

#точка крепления пружины и балки

B = ax.plot(xB, yB[0], 'o',color=[1,0,0])[0]

# диск по точкам, используя уравнение окружности

def disk(x, y, r):

cx = [x + r \* np.sin(i / 100) for i in range(0, 628)]

cy = [y + r \* np.cos(i / 100) for i in range(0, 628)]

return (cx, cy)

x\_disk = s \* np.cos(phi)

# l - s - r \* np.tan(phi) - вертикальное расстояние от точки B до центра диска

# sin(phi) - учитывает горизонтальное перемещение диска в результате поворота балки.

# r/cos(phi) - компенсация вертикального перемещения диска вдоль наклоненной балки, что учитывает наклон балки.

y\_disk = (l - s - r \* np.tan(phi)) \* np.sin(phi) + r / np.cos(phi)

disk\_ = ax.plot(disk(x\_disk[0], y\_disk[0] + yB[0], r)[0], disk(x\_disk[0], y\_disk[0] + yB[0], r)[1], 'green')[0]

def kadr(i):

print(yB[i])

B.set\_data(xB, np.abs(yB[i]))

Pruzzhina.set\_data(xP + xB, np.abs(yP\*(yB[i])))

Stick.set\_data([xA, xB], [yA, np.abs(yB[i])])

disk\_.set\_data(disk(x\_disk[i], y\_disk[i] + yB[i], r)[0], disk(x\_disk[i], y\_disk[i] + yB[i], r)[1])

return [B, Pruzzhina, Stick, disk]

kino = FuncAnimation(fig,kadr,interval = 10,frames=len(t))

plt.show()

**Результат работы:**

