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 20**

Выполнил студент группы М8О-212Б-22

Корнев Максим Сергеевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

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

подпись, дата

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

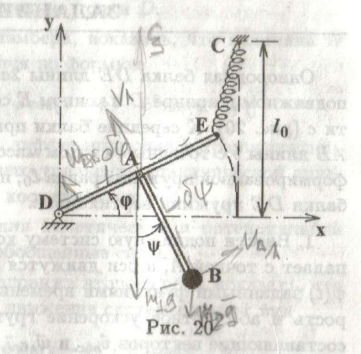
Москва, 2023

**Вариант № «20»**

**Задание:**

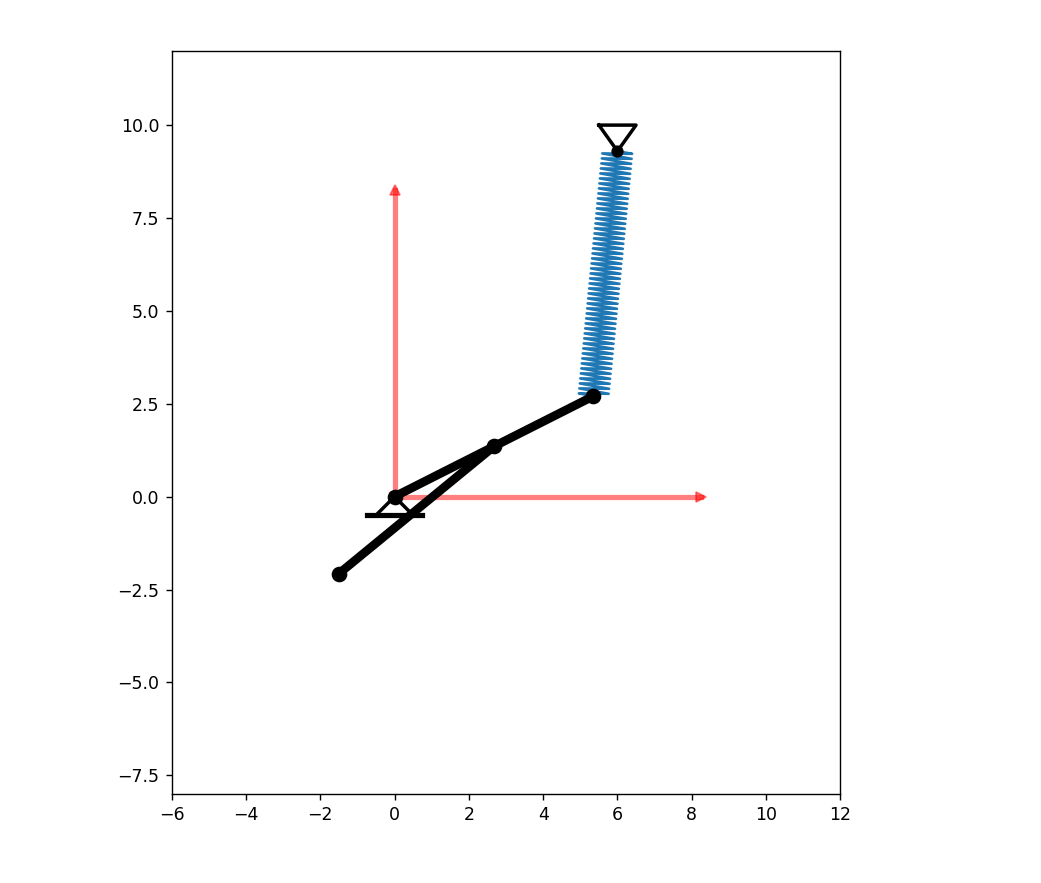
Реализовать анимацию движения механической системы используя язык программирования Python.

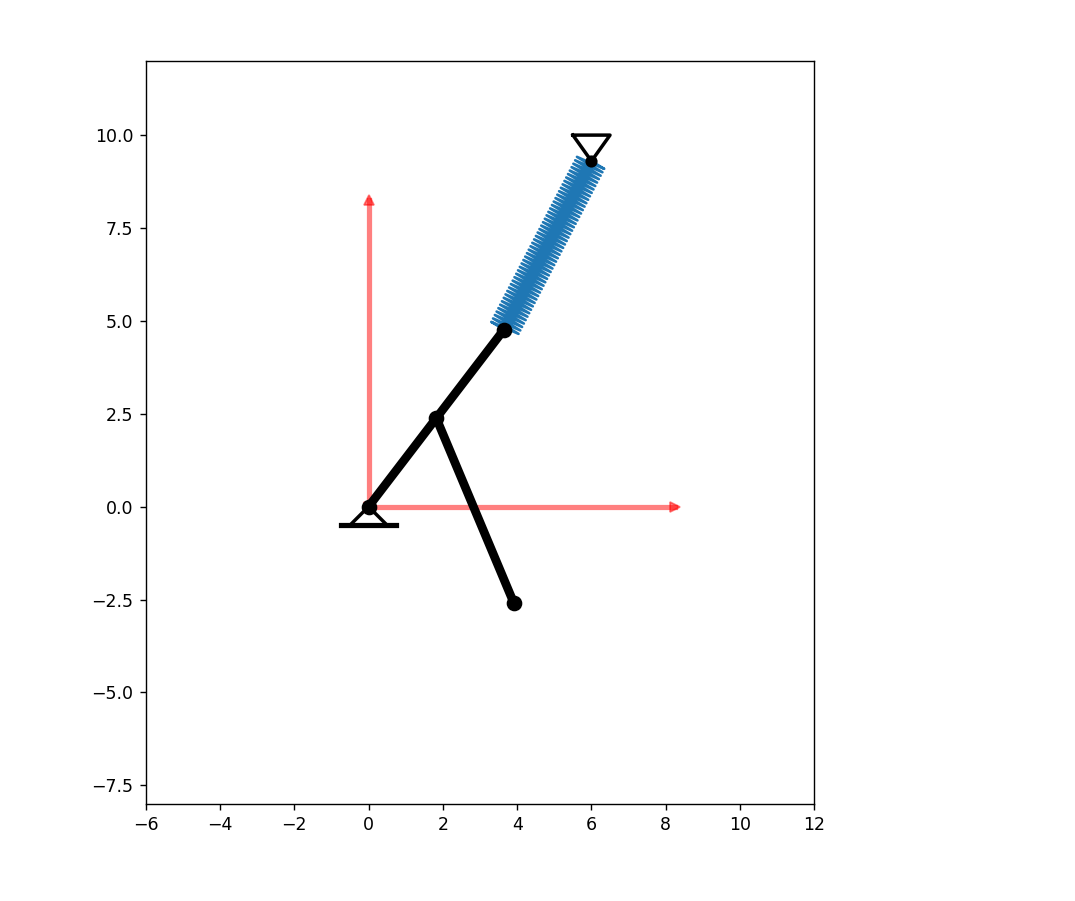
**Механическая система:**

****

Код программы:

import numpy as np  
import matplotlib.pyplot as plt  
from matplotlib.animation import FuncAnimation  
import math  
  
# Функция для поворота  
def Rot2D(X, Y, Alpha):  
 RX = X\*np.cos(Alpha) - Y\*np.sin(Alpha)  
 RY = X\*np.sin(Alpha) + Y\*np.cos(Alpha)  
 return RX, RY  
  
# Вспомогательные данные  
Steps = 500  
t\_fin = 20  
t = np.linspace(0, t\_fin, Steps)  
phi = np.sin(t)  
ksi = np.cos(t)  
  
a = 3  
lenDE = 2\*a # Длина стержня DE  
l = a \* 1.8 # Длина стержня АВ  
l0 = 1.55 \* lenDE # Высота, на которой закреплена пружина  
D = np.array([0, 0])  
  
# Прорисовка пружины  
K = 100  
Sh = 0.4  
b = 1/(K-2)  
X\_Spr = np.zeros(K)  
Y\_Spr = np.zeros(K)  
X\_Spr[0] = 0  
Y\_Spr[0] = 0  
X\_Spr[K-1] = 1  
Y\_Spr[K-1] = 0  
  
for i in range(K-2):  
 X\_Spr[i+1] = b\*((i+1) - 1/2)  
 Y\_Spr[i+1] = Sh\*(-1)\*\*i  
  
# Координаты точки E  
Ex = lenDE \* np.cos(phi)  
Ey = lenDE \* np.sin(phi)  
  
# Смещение точки B относительно точки A  
Bx = l \* np.sin(ksi)  
By = l \* np.cos(ksi)  
  
# Подсчитываю отрезки, необходимые для вычисления угла наклона пружины  
Spr\_x = lenDE - Ex  
Spr\_y = l0 - Ey  
  
#длина пружины в каждый момент времени  
length\_Spr = (Spr\_x\*\*2 + Spr\_y\*\*2)\*\*(0.5)  
  
# Отрисовка окна (с указанием параметров)  
fig = plt.figure(figsize=[20, 10])  
ax = fig.add\_subplot(1, 1, 1)  
ax.set\_aspect('equal') # для равности осей  
ax.set\_ylim([-8, 12])  
ax.set\_xlim([-6, 12])  
  
# ПРУЖИНА  
# Получаю координаты пружины после поворота  
Spr\_x\_L\_fi, Spr\_y\_fi = Rot2D(X\_Spr, Y\_Spr, -(math.pi/2 + abs(math.atan2(Spr\_x[0], Spr\_y[0]))))  
  
# Задаю пружину уже после поворота, перемещая её в конечную позицию и растягивая на длину  
WArrow, = ax.plot(Spr\_x\_L\_fi + lenDE, (Spr\_y\_fi\*length\_Spr[0]) + l0)  
  
# Крепёж для пружины  
ax.plot(2\*a, l0, color='black', linewidth=5, marker='o')  
ax.plot([2\*a-0.5, 2\*a+0.5, 2\*a, 2\*a-0.5], [l0+0.7, l0+0.7, l0, l0+0.7], color='black', linewidth=2, )  
  
ax.plot([-0.5, 0.5, 0, -0.5], [-0.5, -0.5, 0, -0.5], color='black', linewidth=2)  
ax.plot([-0.75, 0.75], [-0.5, -0.5], color='black', linewidth=3)  
  
# Рисую оси  
ax.plot([0, 0], [0, 8.25], color='red', linewidth=3, linestyle='solid', alpha=0.5, marker='^')  
ax.plot([0, 8.25], [0, 0], color='red', linewidth=3, linestyle='solid', alpha=0.5, marker='>')  
  
# Рисую DE  
Drawed\_DE = ax.plot(np.array([0, Ex[0]]), np.array([0, Ey[0]]), color='black', linewidth=5, marker='o', markersize=8)[0]  
  
# Рисую АВ  
Drawed\_AB = ax.plot(np.array([Ex[0]/2, Ex[0]/2 + Bx[0]]), np.array([Ey[0]/2, Ey[0]/2 - By[0]]), color='black', linewidth=5, marker='o', markersize=8)[0]  
  
def anima(i):  
 # ---  
 # Отрисовываю отрезок DE  
 Drawed\_DE.set\_data(np.array([0, Ex[i]]), np.array([0, Ey[i]]))  
 # Получаю координаты пружины после поворота  
 Spr\_x\_L\_fi, Spr\_y\_fi = Rot2D(X\_Spr\*length\_Spr[i], Y\_Spr, -(math.pi/2 + abs(math.atan2(Spr\_x[i], Spr\_y[i]))))  
 # Задаю пружину уже после поворота, причём сразу перемещаю её в конечную позицию и растягиваю на длину  
 WArrow.set\_data(Spr\_x\_L\_fi + lenDE, (Spr\_y\_fi) + l0)  
  
 Drawed\_AB.set\_data(np.array([Ex[i]/2, Ex[i]/2 + Bx[i]]), np.array([Ey[i]/2, Ey[i]/2 - By[i]]))  
 return Drawed\_DE  
  
anim = FuncAnimation(fig, anima, frames=len(t), interval=100, repeat=False)  
  
plt.show()

Работа программы: 



Вывод:

В ходе лабораторной работы была создана анимация механической системы с использованием библиотек numpy и matplotlib. Шаги включали в себя определение параметров системы, расчет координат точек, и создание анимации. Комментарии были добавлены для понимания кода, и использовались различные параметры для улучшения визуального восприятия. Также рассматривались вопросы управления скоростью анимации и решения проблем с её остановкой.

Начало формы