ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ ДВИЖЕНИЯ МЕХАНИЧЕСКОЙ СИСТЕМЫ»**

**ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА»**

**ВАРИАНТ ЗАДАНИЯ №22**

Выполнил студент группы М80-203Б-23

Салихов Р.Р.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

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

подпись, дата

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

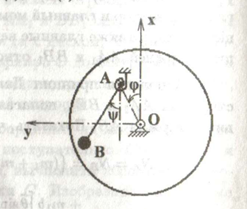
Москва, 2024

**Вариант №22**

**Задание:**

Реализовать анимацию движения механической системы.

**Механическая система:**



**Текст программы**

|  |
| --- |
| import numpy as np # для работы с массивами  import matplotlib.pyplot as plt # ля построения графиков (matplotlib)  from matplotlib.animation import FuncAnimation #для создания анимаций  import math  #функция для поворота точки на плоскости  def Rot2D(X, Y, Alpha):  RX = X \* np.cos(Alpha) - Y \* np.sin(Alpha) # поворот по оси X  RY = X \* np.sin(Alpha) + Y \* np.cos(Alpha) # поворот по оси Y  return RX, RY # возвращаем новые координаты после поворота  # определяю константы и функции для угловых перемещений  R = 2 # радиус диска  a = 1 # расстояние от центра O до точки A  l = 2.5 # ллина стержня AB  phi\_t = np.sin # функция углового перемещения \phi(t)  psi\_t = np.cos # функция углового перемещения \psi(t)  steps = 500  T = np.linspace(0, 40, steps) # массив времени от 0 до 40 с 500 шагами  #функция для вычисления координат точек в зависимости от времени  def compute\_positions(t):  phi = phi\_t(t) # угол для точки A  psi = psi\_t(t) # угол для точки B    #позиция точки O (центр диска, фиксированная)  X\_O, Y\_O = 0, 0    #позиция точки A (на диске)  X\_A = X\_O + a \* np.sin(phi)  Y\_A = Y\_O - a \* np.cos(phi)    #позиция точки B (конец стержня AB)  X\_B = X\_A - l \* np.sin(psi)  Y\_B = Y\_A - l \* np.cos(psi)  return X\_O, Y\_O, X\_A, Y\_A, X\_B, Y\_B #возвращаем координаты всех точек  #подготовим массивы для хранения позиций  X\_O, Y\_O = [], [] #списки для хранения координат точки O  X\_A, Y\_A = [], [] #списки для хранения координат точки A  X\_B, Y\_B = [], [] #списки для хранения координат точки B  #для каждого времени из массива T вычисляем позиции  for t in T:  xo, yo, xa, ya, xb, yb = compute\_positions(t)  X\_O.append(xo) #добавляю координаты точки O в список  Y\_O.append(yo) #добавляю координаты точки O в список  X\_A.append(xa) #добавляю координаты точки A в список  Y\_A.append(ya) #добавляю координаты точки A в список  X\_B.append(xb) #добавляю координаты точки B в список  Y\_B.append(yb) #добавляю координаты точки B в список  #настраиваю график  fig, ax = plt.subplots(figsize=(8, 8)) #создаю фигуру и оси для графика  ax.axis('equal') #устанавливаю равные масштабы по осям  ax.set\_xlim(-R - l, R + l) #устанавливаю пределы оси X  ax.set\_ylim(-R - l, R + l) #устанавливаю пределы оси Y  ax.set\_xlabel("x") # подписываб ось X  ax.set\_ylabel("y") # подписываю ось Y  #строю диск, стержень и точки  phi = np.linspace(0, 2 \* np.pi, 100) #угол для рисования окружности  Disk, = ax.plot(R \* np.cos(phi), R \* np.sin(phi), 'gray') #рисую диск серым цветом  Rod, = ax.plot([], [], 'b-', linewidth=2) #Стержень, который будет обновляться  Point\_O, = ax.plot([], [], 'ro', markersize=8) #центр диска)  Point\_A, = ax.plot([], [], 'go', markersize=8) # Точка A  Point\_B, = ax.plot([], [], 'mo', markersize=8) # Точка B  #обновление анимации  def animate(i):  # обновляю позиций на графике  Disk.set\_data(R \* np.cos(phi) + X\_O[i], R \* np.sin(phi) + Y\_O[i]) # обноволяю диск  Rod.set\_data([X\_A[i], X\_B[i]], [Y\_A[i], Y\_B[i]]) # обнволяю стержень  Point\_O.set\_data([X\_O[i]], [Y\_O[i]]) # обноволяю точку O  Point\_A.set\_data([X\_A[i]], [Y\_A[i]]) # обноволяю точку A  Point\_B.set\_data([X\_B[i]], [Y\_B[i]]) # обноволяю точку B  return Disk, Rod, Point\_O, Point\_A, Point\_B # вернул обновленные объекты  # Запуск анимации  ani = FuncAnimation(fig, animate, frames=steps, interval=50, blit=True)  plt.show() #вывожу анимацию |

**Результат работы программы:**

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

**Вывод:** в ходе выполнения лабораторной работы я освоил использование библиотек Python, таких как NumPy и Matplotlib, для моделирования и визуализации динамических систем. Научился создавать функции для вычисления координат элементов системы во времени, а также реализовывать анимации сложного движения с помощью FuncAnimation. Понял, как интегрировать математические модели движения с графическими инструментами для наглядного отображения поведения системы. Теперь вместо произвольных зависимостей можно внедрять конкретные законы движения, что позволяет более точно моделировать и анализировать реальное движение заданных элементов системы.