ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕ-НИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ОТЧЕТ

О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ №2

«АНИМАЦИЯ СИСТЕМЫ»

ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА И ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ»

ВАРИАНТ ЗАДАНИЯ №15

Выполнил(а) студент группы М8О-208Б-23

Денисов Константин Дмитриевич_____
подпись, дата

Проверил и принял

Ст. преп. каф. 802 Волков Е.В_____
подпись, дата

с оценкой _____

Москва, 2024

Вариант №15

Задание:

Реализовать анимацию движения механической системы в среде Python

Механическая система:

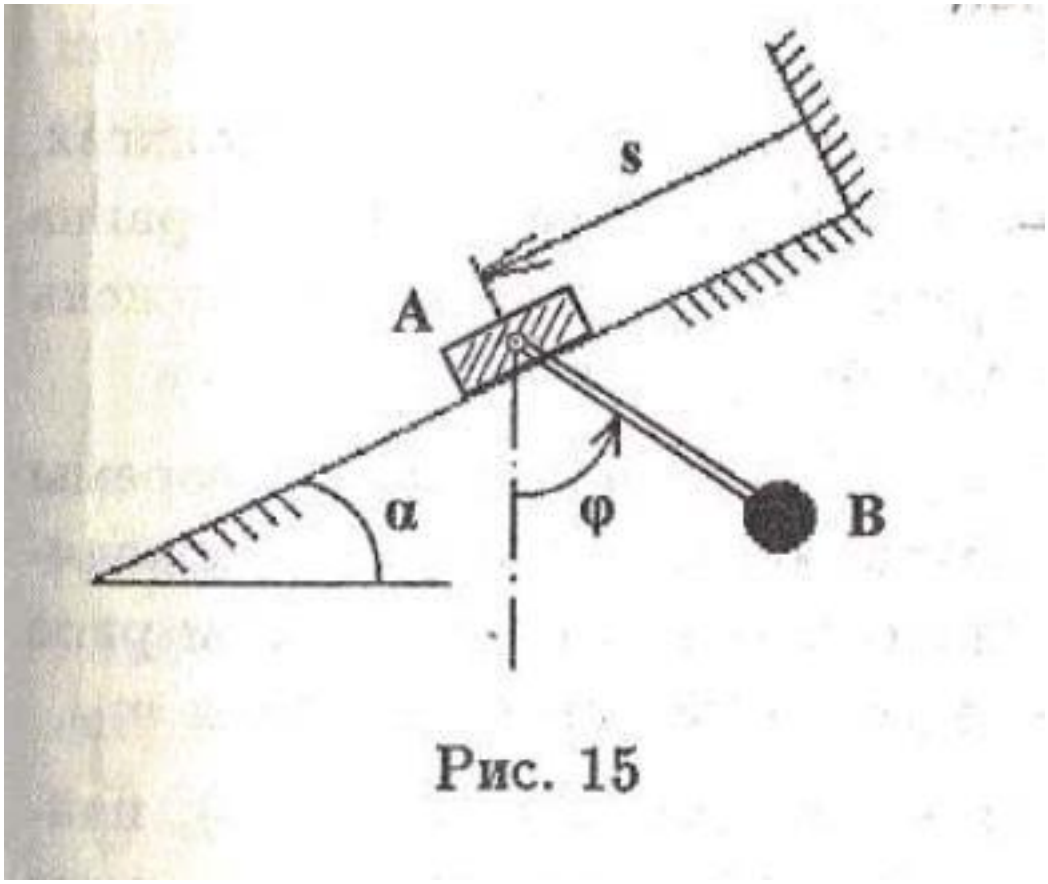


Рис. 15

Текст программы

```
import matplotlib              # Импортируем библиотеку matplotlib для визуализации данных
import numpy as np             # Импортируем библиотеку numpy для работы с массивами и математическими операциями
import matplotlib.pyplot as plt  # Импортируем модуль pyplot из matplotlib для построения графиков
from matplotlib.animation import FuncAnimation  # Импортируем модуль анимации для создания движущихся графиков
import sympy as sp             # Импортируем библиотеку sympy для символьных вычислений (в данном коде не используется)
import math                    # Импортируем библиотеку math для работы с математическими функциями

matplotlib.use("TkAgg")        # Указываем, что matplotlib будет использовать интерфейс Tk для отображения графиков

# Создаем массив времени от 1 до 20, состоящий из 1001 равномерно распределенных точек
t = np.linspace(1, 20, 1001)
```

```
# Определяем параметры движения: координата x изменяется по косинусу, угол phi по синусу
x = np.cos(t)
phi = np.sin(2 * t)
alpha = math.pi / 6            # Угол наклона системы (30 градусов)
X_0 = 4                        # Начальная координата объекта
a = 2.5                        # Параметр ширины коробки
b = 3                          # Параметр высоты коробки
l = 3                          # Длина маятника

# Вычисляем координаты точек A и B
X_A = a / 2 * x                # Горизонтальная координата точки A
Y_A = X_A                      # Вертикальная координата точки A (для симметрии)
X_B = X_A - l * np.sin(phi)    # Горизонтальная координата точки B, зависимость от угла phi
Y_B = Y_A - l * np.cos(phi)    # Вертикальная координата точки B, зависимость от угла phi

# Определяем форму коробки с помощью массивов координат её вершин
X_Box = np.array([-1.5, -3, 0, 1.5, -1.5])
Y_Box = np.array([-1.5, -0.5, 2.5, 1.5, -1.5])

# Координаты для вспомогательной прямой на графике
X_Straight = [-10, 0, 10]
Y_Straight = [-10, 0, 10]

# Создаем фигуру для отображения графиков
fig = plt.figure(figsize=[9, 5])       # Устанавливаем размер фигуры
ax = fig.add_subplot(1, 2, 1)          # Создаем область для первого графика
ax.axis('equal')                       # Задаем равное масштабирование осей
ax.set(xlim=[-5, 5], ylim=[-5, 5])     # Устанавливаем пределы отображения по осям

# Рисуем статические элементы
ax.plot(X_Straight, Y_Straight)        # Вспомогательная диагональная прямая
Drawed_Box = ax.plot(X_A[0] + X_Box, Y_A[0] + Y_Box)[0]  # Рисуем коробку в начальном положении
Line_AB = ax.plot([X_A[0], X_B[0]], [Y_A[0], Y_B[0]])[0]  # Линия от точки A до точки B
Point_A = ax.plot(X_A[0], Y_A[0], marker='o')[0]          # Отображаем точку A
Point_B = ax.plot(X_B[0], Y_B[0], marker='o', markersize=10)[0]  # Отображаем точку B

# Создаем дополнительные графики для отображения зависимостей
ax2 = fig.add_subplot(4, 2, 2)
ax2.plot(t, X_A)                       # График X_A(t)
plt.xlabel('t values')                 # Подпись оси времени
plt.ylabel('x values')                 # Подпись оси координат

ax3 = fig.add_subplot(4, 2, 4)
ax3.plot(t, Y_A)                       # График Y_A(t)
plt.xlabel('t values')
plt.ylabel('y values')

ax4 = fig.add_subplot(4, 2, 6)
ax4.plot(t, X_B)                       # График X_B(t)
plt.xlabel('t values')
plt.ylabel('x values')

ax5 = fig.add_subplot(4, 2, 8)
ax5.plot(t, Y_B)                       # График Y_B(t)
```

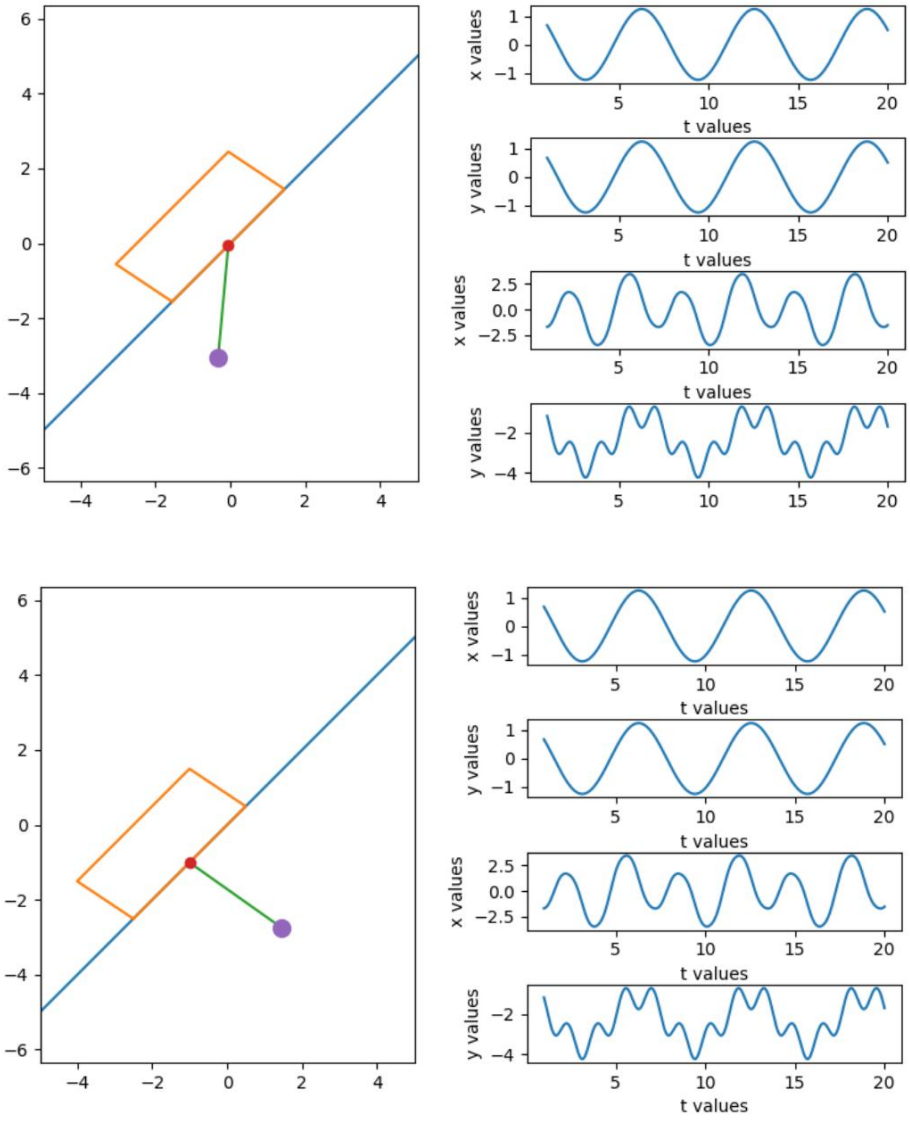
```
plt.xlabel('t values')
plt.ylabel('y values')

plt.subplots_adjust(wspace=0.3, hspace=0.7)  # Настраиваем отступы между графиками

# Функция для обновления данных на графике в каждом кадре анимации
def Kino(i):
    Point_A.set_data(X_A[i], Y_A[i])              # Обновляем положение точки A
    Point_B.set_data(X_B[i], Y_B[i])              # Обновляем положение точки B
    Line_AB.set_data([X_A[i], X_B[i]], [Y_A[i], Y_B[i]])  # Обновляем положение линии AB
    Drawed_Box.set_data(X_A[i] + X_Box, Y_A[i] + Y_Box)   # Обновляем положение коробки
    return [Point_A, Point_B, Line_AB, Drawed_Box]        # Возвращаем обновленные элементы

# Создаем анимацию
anima = FuncAnimation(fig, Kino, frames=1001, interval=10)  # Настраиваем 1001 кадр с
интервалом в 10 мс
plt.show()  # Отображаем график
```

Результат работы



Вывод

ходе лабораторной работы я реализовал анимацию движения механической системы в среде Python. Для этого пришлось изучить основы создания анимации на Питоне.