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 №12

Выполнил(а) студент группы М8О-207Б-22

Брюханов З. Д._____
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.

Рис. 11

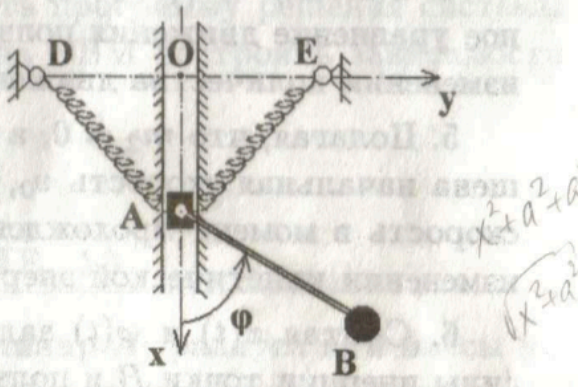


Рис. 12

ЗАДАНИЕ 12

К ползуну A массы m_1, помещенному между вертикальными направляющими, прикреплены две пружины жесткости c каждая, расположенные в вертикальной плоскости (рис. 12). Невесомый стержень AB длины ℓ, на конце которого укреплен груз B массы m_2, соединен шарнирно с ползуном A. Стержень AB и груз B движутся в вертикальной плоскости. Концы пружин D и E расположены на одной горизонтали, причем $DO = OE = a$. Когда ползун A находится на линии DE (в точке O), пружины не напряжены.

Текст программы

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from matplotlib.patches import Rectangle
import sympy as sp
"""Определение параметров"""
a = 10  # Длина маятника
lAB = 5  # Длина линии AB
sA = 1  # Размер объекта A
t = sp.Symbol('t')  # Время как символьная переменная
```

```
# Определение s и phi
xA = 4 * sp.cos(3 * t)
phi = 4 * sp.sin(t - 10)
# Движение объекта B
xB = xA + lAB * sp.sin(phi)
yB = lAB * sp.cos(phi)
# Модули скорости и ускорения объекта B
VmodB = sp.sqrt(sp.diff(xB, t) ** 2 + sp.diff(yB, t) ** 2)
WmodB = sp.sqrt(sp.diff(xB, t, 2) ** 2 + sp.diff(yB, t, 2) ** 2)
"""Построение функций"""
countOfFrames = 200
T_start, T_stop = 0, 12
T = np.linspace(T_start, T_stop, countOfFrames)
# Лямбда-функции для численных значений
XA_def = sp.lambdify(t, xA)
XB_def = sp.lambdify(t, xB)
YB_def = sp.lambdify(t, yB)
VmodB_def = sp.lambdify(t, VmodB)
WmodB_def = sp.lambdify(t, WmodB)
XA = XA_def(T)
XB = XB_def(T)
YB = YB_def(T)
VB = VmodB_def(T)
WB = WmodB_def(T)
"""Построение графика"""
fig = plt.figure(figsize=(10, 7))
# Один подграфик на всю ширину
ax1 = fig.add_subplot(1, 1, 1)
ax1.axis('equal')
ax1.set(ylim=[-a, XA.max() + a], xlim=[min(-lAB, -a), max(lAB, a)])
ax1.set_xlabel('ось y')
ax1.set_ylabel('ось x')
ax1.invert_yaxis()
# Исходные точки D и E
ax1.plot(-a, XA.min(), marker='o', color='black')
ax1.plot(a, XA.min(), marker='o', color='black')
# Линии, между которыми находится A
ax1.plot([-sA / 2, -sA / 2], [XA.min() - sA, XA.max() + sA], linestyle='-.',
color='black')
ax1.plot([sA / 2, sA / 2], [XA.min() - sA, XA.max() + sA], linestyle='-.',
color='black')
# Начальные положения
# A
PA = ax1.add_patch(Rectangle(xy=[-sA / 2, XA[0] - sA / 2], width=sA, height=sA,
color='green'))
```

```
# B
PB, = ax1.plot(YB[0], XB[0], marker='o', color='r', markersize=10)
# Линия AB
PAB, = ax1.plot([0, YB[0]], [XA[0], XB[0]], 'black')
# Линии DA и EA
PDA, = ax1.plot([-a, 0], [XA.min(), XA[0]], linestyle='--', color='m')
PEA, = ax1.plot([a, 0], [XA.min(), XA[0]], linestyle='--', color='m')
# Функция для обновления положения
def anima(i):
    PA.set(xy=[-sA / 2, XA[i] - sA / 2])
    PB.set_data(YB[i], XB[i])
    PAB.set_data([0, YB[i]], [XA[i], XB[i]])
    PDA.set_data([-a, 0], [XA.min(), XA[i]])
    PEA.set_data([a, 0], [XA.min(), XA[i]])
    return PAB, PDA, PEA, PA, PB
# Анимационная функция
anim = FuncAnimation(fig, anima, frames=countOfFrames, interval=100, blit=True)
plt.show()
```

Описание

Программа моделирует движение маятника, состоящего из объектов A и B, и визуализирует его анимацию. Основные шаги включают в себя определение параметров маятника, таких как длина и размеры объектов, а также символьное описание их движения в зависимости от времени.

С использованием библиотеки SymPy производится вычисление координат объектов, их скоростей и ускорений. Полученные выражения используются для построения лямбда-функций, которые численно оценивают значения координат, скоростей и ускорений в течение заданного времени.

Графическое представление осуществляется с использованием библиотеки Matplotlib. Анимация отображает движение объектов A и B, а также линии связи между ними. Визуализация позволяет наглядно изучить динамику маятника, его начальные положения и изменения во времени.

Рассмотрим, что делает каждый блок кода.

В первом блоке кода задаются параметры маятника, такие как длина, размеры объектов и время, представленное символьной переменной.

Во втором блоке выражается движение объекта A с использованием тригонометрических функций.

Третий блок описывает движение объекта B, который связан с объектом A и движется в соответствии с углом наклона маятника.

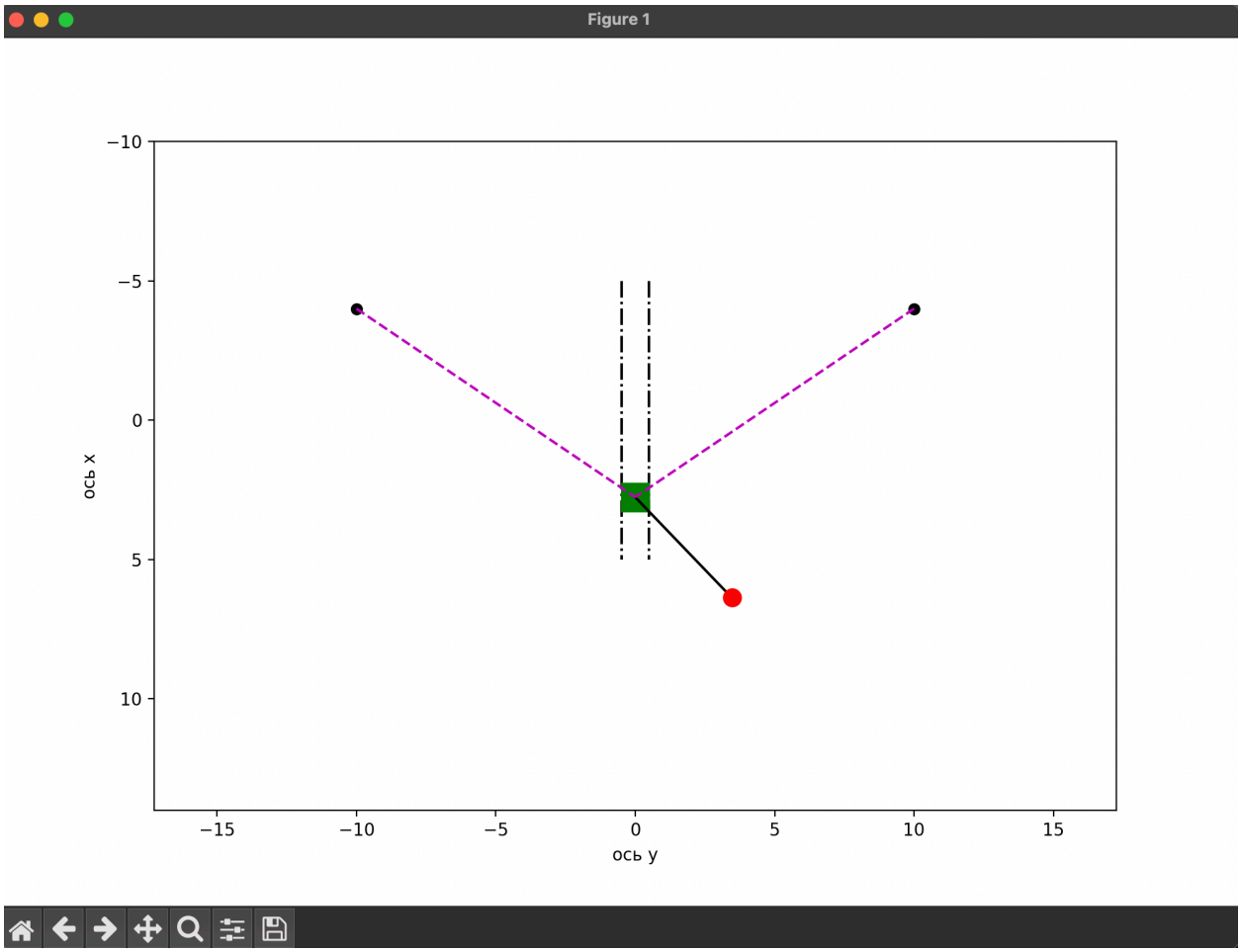
Четвертый блок выражает модули скорости и ускорения объекта B, используя производные по времени от координат.

Пятый блок создает лямбда-функции для численного вычисления значений координат, скоростей и ускорений на временной сетке.

Шестой блок инициализирует массивы значений координат, скоростей и ускорений объекта B на основе полученных лямбда-функций.

В седьмом блоке строится график, представляющий движение маятника и его динамические характеристики. Объекты A и B отображаются в начальных положениях, после чего начинают анимироваться. Линии связи и маятник анимируются с течением времени.

Результат работы программы



Вывод

В ходе выполнения этой работы я научился моделировать и визуализировать движение маятника с использованием символьных выражений и численных методов. Я познакомился с основами работы с библиотеками SymPy, NumPy и Matplotlib в контексте создания анимации для механической системы. Кроме того, я научились формулировать уравнения движения, рассчитывать скорость и ускорение объектов, а также строить анимации для визуализации динамических процессов.