ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«ДИНАМИКА СИСТЕМЫ»**

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

**ВАРИАНТ ЗАДАНИЯ 12**

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

Ганяк Александр Олегович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Зав. каф. 802, Волков Е.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

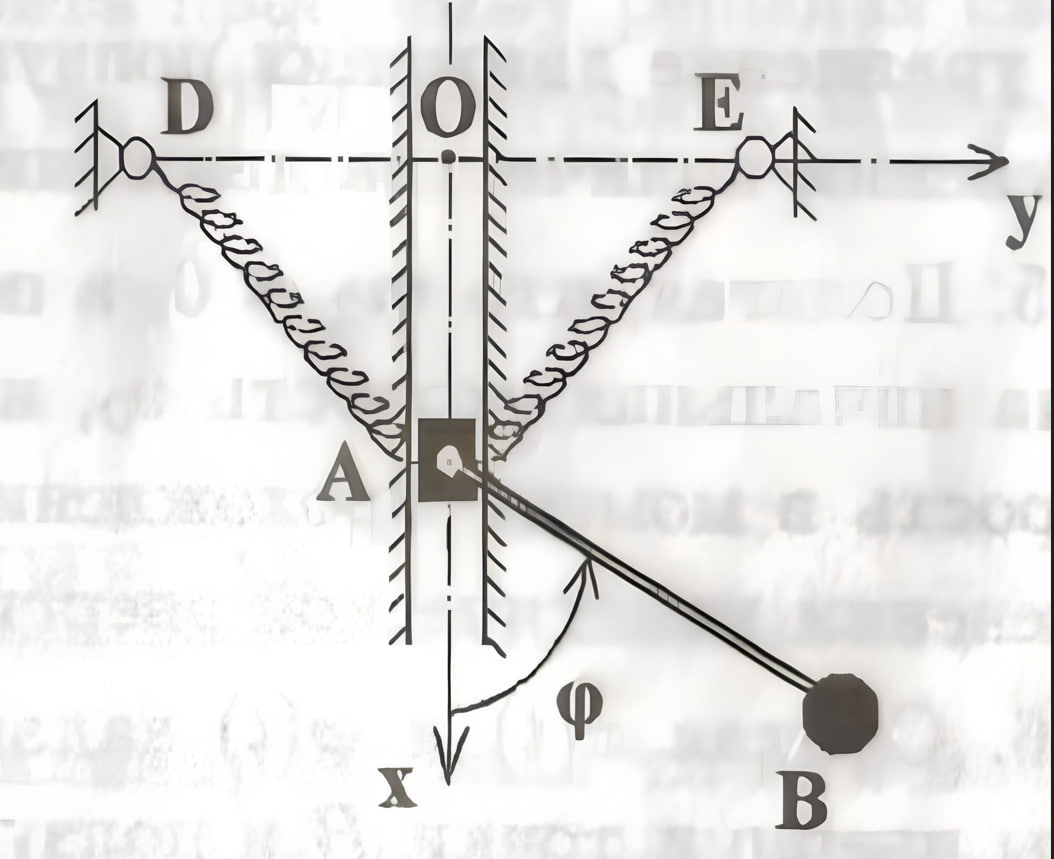
подпись, дата

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

Москва, 2024

**Задание:** Реализовать анимацию движения механической системы.

**Система:**



**Текст программы:**

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from matplotlib.patches import Rectangle, Circle

# КОНФИГ

rect\_width = 1 # Ширина прямоугольника

rect\_length = 2 # Длина прямоугольника

circle\_radius = 0.2 # Радиус круга

spring\_distance = 3 # Расстояние до пружин

rod\_length = 2 # Длина стержня

# Начальные позиции

AX = 0

AY = -0.5

BX = rod\_length \* np.sin(np.pi / 6)

BY = -(rod\_length \* np.cos(np.pi / 6) + AY)

# Начало построения графиков и анимации

fig = plt.figure(figsize=(12, 6))

plt.suptitle('Анимация', fontsize=16)

# Настройка оси для симуляции

ax\_sim = fig.add\_subplot(1, 2, 1)

ax\_sim.axis("equal")

ax\_sim.set\_xlim(-spring\_distance - rect\_width - 1, spring\_distance + rect\_width + 1)

ax\_sim.set\_ylim(-spring\_distance - rect\_length - 1, spring\_distance + rect\_length + 1)

# Подготовка данных для окружения

wall\_left\_x = [-rect\_width / 2, -rect\_width / 2]

wall\_right\_x = [rect\_width / 2, rect\_width / 2]

wall\_y = [ -spring\_distance - rect\_length, spring\_distance + rect\_length]

# Построение окружения

ax\_sim.plot(0, 0, marker=".", color="red") # Красная точка O

ax\_sim.plot(wall\_left\_x, wall\_y, linestyle='--', color="grey") # Левая стена

ax\_sim.plot(wall\_right\_x, wall\_y, linestyle='--', color="grey") # Правая стена

# Инициализация пружин с пустыми данными

spring\_left, = ax\_sim.plot([], [], color="grey") # Левая пружина

spring\_right, = ax\_sim.plot([], [], color="grey") # Правая пружина

# Соединения

ax\_sim.plot(-spring\_distance, 0, marker=".", color="black") # Левое соединение

ax\_sim.plot(spring\_distance, 0, marker=".", color="black") # Правое соединение

ax\_sim.axhline(0, linestyle=':', color='k') # Горизонтальная пунктирная линия

# Создание прямоугольника и круга

rect = Rectangle((-rect\_width / 2, AY - rect\_length / 2), rect\_width, rect\_length, color="black")

circ = Circle((BX, BY), circle\_radius, color="grey")

# Построение радиус-вектора точки B

radius\_vector, = ax\_sim.plot([AX, BX], [AY, BY], color="grey")

# Добавление патчей для прямоугольника и круга

ax\_sim.add\_patch(rect)

ax\_sim.add\_patch(circ)

# Добавление подписей к точкам, грузу и ползунку

ax\_sim.text(-spring\_distance - 0.2, 0, 'D', ha='right', va='bottom')

ax\_sim.text(spring\_distance + 0.2, 0, 'E', ha='left', va='bottom')

text\_A = ax\_sim.text(AX, AY - rect\_length / 2 - 0.2, 'A', ha='right', va='bottom')

text\_B = ax\_sim.text(BX + 0.2, BY, 'B', ha='left', va='bottom')

# Функция для создания пружины

def spring(start, end, num\_segments=12, amplitude=0.1):

x\_vals = np.linspace(start[0], end[0], num\_segments)

y\_vals = np.linspace(start[1], end[1], num\_segments)

dist = np.sqrt((end[0] - start[0])\*\*2 + (end[1] - start[1])\*\*2)

amp\_factor = amplitude \* (2 + 0.5 \* dist) # Изменение амплитуды пружины в зависимости от расстояния

for i in range(1, num\_segments, 2):

y\_vals[i] += amp\_factor

return x\_vals, y\_vals

# Функция для инициализации позиций

def init():

rect.set\_y(AY - rect\_length / 2)

circ.center = (BX, BY)

radius\_vector.set\_data([AX, BX], [AY, BY])

spring\_left.set\_data([], [])

spring\_right.set\_data([], [])

return rect, circ, radius\_vector, spring\_left, spring\_right, text\_A, text\_B

# Функция для обновления позиций на каждом кадре анимации

def animate(i):

t = np.linspace(0, 2 \* np.pi, 500)[i]

# Плавное движение прямоугольника (вверх-вниз по синусоиде)

rect\_y = 2 \* np.sin(t)

rect.set\_y(rect\_y - rect\_length / 2)

# Координата центра прямоугольника

AX = 0

AY = rect\_y

# Плавное вращение круга вокруг центра прямоугольника

angle = t

circ\_x = AX + rod\_length \* np.cos(angle)

circ\_y = AY + rod\_length \* np.sin(angle)

circ.center = (circ\_x, circ\_y)

# Обновление радиус-вектора

radius\_vector.set\_data([AX, circ\_x], [AY, circ\_y])

# Обновление пружин с новыми координатами

spring\_left\_x, spring\_left\_y = spring((-spring\_distance, 0), (-rect\_width / 2, rect\_y), num\_segments=12, amplitude=0.1)

spring\_right\_x, spring\_right\_y = spring((spring\_distance, 0), (rect\_width / 2, rect\_y), num\_segments=12, amplitude=0.1)

spring\_left.set\_data(spring\_left\_x, spring\_left\_y)

spring\_right.set\_data(spring\_right\_x, spring\_right\_y)

# Обновление подписей

text\_A.set\_position((AX, AY - rect\_length / 2 - 0.2))

text\_B.set\_position((circ\_x + 0.2, circ\_y))

return spring\_left, spring\_right, rect, radius\_vector, circ, text\_A, text\_B

anim = FuncAnimation(

fig,

animate,

init\_func=init,

frames=len(np.linspace(0, 2 \* np.pi, 500)),

blit=True,

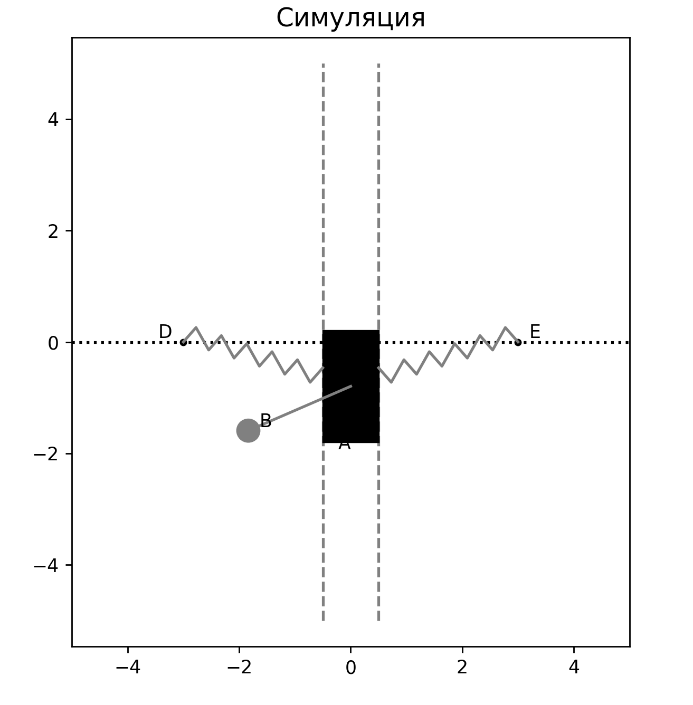
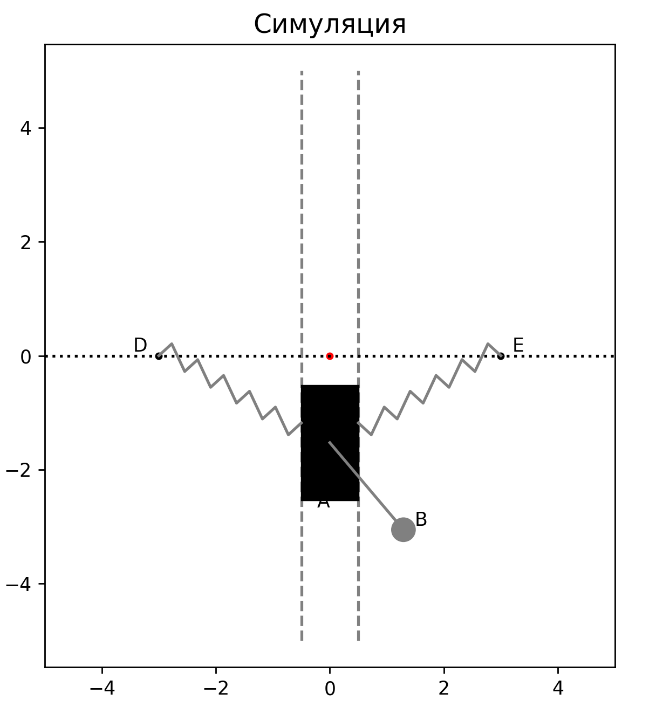
interval=20, # Интервал между кадрами в миллисекундах

repeat=True

)

plt.show()

**Результат работы программы:**



**Вывод:**

В ходе данной лабораторной работы мною была реализована анимация движения механической системы, представляющей собой более сложную структуру, чем просто точки и векторы. Созданная программа позволила визуализировать динамику системы, что значительно упростило анализ её поведения и взаимосвязей между элементами.

Работа над этим этапом помогла мне улучшить навыки программирования и визуализации сложных движений. Полученные знания и опыт станут основой для совершенствования моего кода в следующей лабораторной работе, где я планирую дополнить функционал программы и углубить её анализ.