ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ОТЧЕТ

О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«АНИМАЦИЯ СИСТЕМЫ»

ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА И ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ»

ВАРИАНТ ЗАДАНИЯ №31

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

Пинчук Михаил Сергеевич _____
подпись, дата

Проверил и принял

Ст. преп. каф. 802 Волков Е.В._____
подпись, дата

с оценкой _____

Москва, 2024

Вариант №31

Задание:

Реализовать анимацию движения механической системы.

Механическая система:

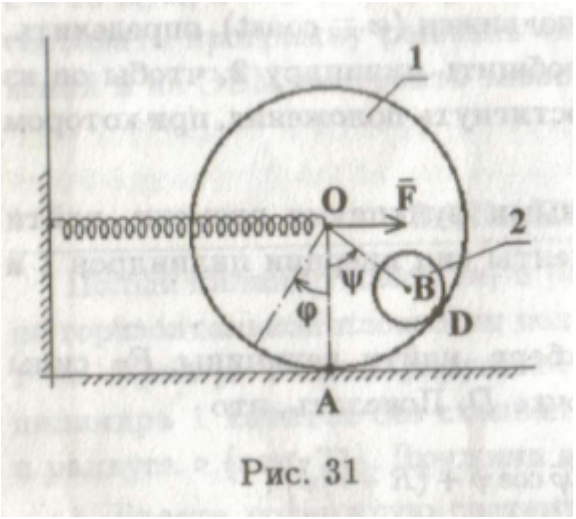


Рис. 31

Текст программы:

```
import numpy as np
import math
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

# Количество шагов анимации и финальное время
num_steps = 100
time_final = 2 * np.pi
time = np.linspace(0, time_final, num_steps)

# Горизонтальное смещение и угол вращения
x_displacement = np.sin(time)  # Смещение по горизонтали
phi_angle = np.sin(-time)      # Угол для вращения второго цилиндра

# Константы для цилиндров и пружины
initial_x = 1.5          # Начальное смещение пружины
main_radius = 0.45       # Радиус основного цилиндра
sub_radius = 0.1         # Радиус дополнительного цилиндра
spring_length = 0.35     # Длина пружины

# Положение центра основного цилиндра (точка A)
x_A = initial_x + x_displacement
y_A = main_radius
```

```
# Положение центра дополнительного цилиндра (точка B)
x_B = x_A + spring_length * np.sin(phi_angle)
y_B = y_A - spring_length * np.cos(phi_angle)

# Координаты для построения окружностей (основной и вспомогательный цилиндры)
circle_angle = np.linspace(0, 2 * math.pi, 100)
x_circle_main = main_radius * np.sin(circle_angle)
y_circle_main = main_radius * np.cos(circle_angle)
x_circle_sub = sub_radius * np.sin(circle_angle)
y_circle_sub = sub_radius * np.cos(circle_angle)

# Создание пружины
num_spring_points = 20  # Количество витков пружины
spring_height = 0.06    # Высота пружины
spring_step = 1 / (num_spring_points - 2)  # Шаг между витками

x_spring = np.zeros(num_spring_points)
y_spring = np.zeros(num_spring_points)
x_spring[0] = 0
y_spring[0] = 0
x_spring[-1] = 1
y_spring[-1] = 0

# Заполняем координаты пружины
for i in range(num_spring_points - 2):
    x_spring[i + 1] = spring_step * ((i + 1) - 1 / 2)
    y_spring[i + 1] = spring_height * (-1) ** i

# Параметры графика
fig = plt.figure(figsize=[8, 6])
ax = fig.add_subplot(1, 1, 1)
ax.axis('equal')  # Равномерный масштаб по осям
ax.set(xlim=[-1, 4], ylim=[-1, 3])

# Построение базовых линий (основание)
ground_x = [0, 0, 4]
ground_y = [2, 0, 0]
ax.plot(ground_x, ground_y, color='black', linewidth=2)

# Построение начальных элементов (обновляемых в анимации)
main_cylinder_drawn = ax.plot(x_A[0] + x_circle_main, y_A + y_circle_main)[0]
sub_cylinder_drawn = ax.plot(x_B[0] + x_circle_sub, y_B[0] + y_circle_sub)[0]

spring_drawn = ax.plot(x_spring * (initial_x + x_displacement[0]), y_spring +
y_A)[0]
point_A = ax.plot(x_A[0], y_A, marker='o', color='red')[0]

# Функция анимации
def animate(frame):
    # Обновляем положение точки A
```

```
    point_A.set_data([x_A[frame]], [y_A])

    # Обновляем основной цилиндр (движение вместе с точкой A)
    main_cylinder_drawn.set_data(x_A[frame] + x_circle_main, y_A +
y_circle_main)

    # Обновляем дополнительный цилиндр (вращается вокруг точки B)
    sub_cylinder_drawn.set_data(x_B[frame] + x_circle_sub, y_B[frame] +
y_circle_sub)

    # Обновляем пружину
    spring_drawn.set_data(x_spring * (initial_x + x_displacement[frame]),
y_spring + y_A)

    return [main_cylinder_drawn, sub_cylinder_drawn, spring_drawn, point_A]

# Создаем анимацию
animation = FuncAnimation(fig, animate, frames=num_steps, interval=50,
repeat=False)

# Отображаем график с анимацией
plt.show()
```

Результат работы программы:

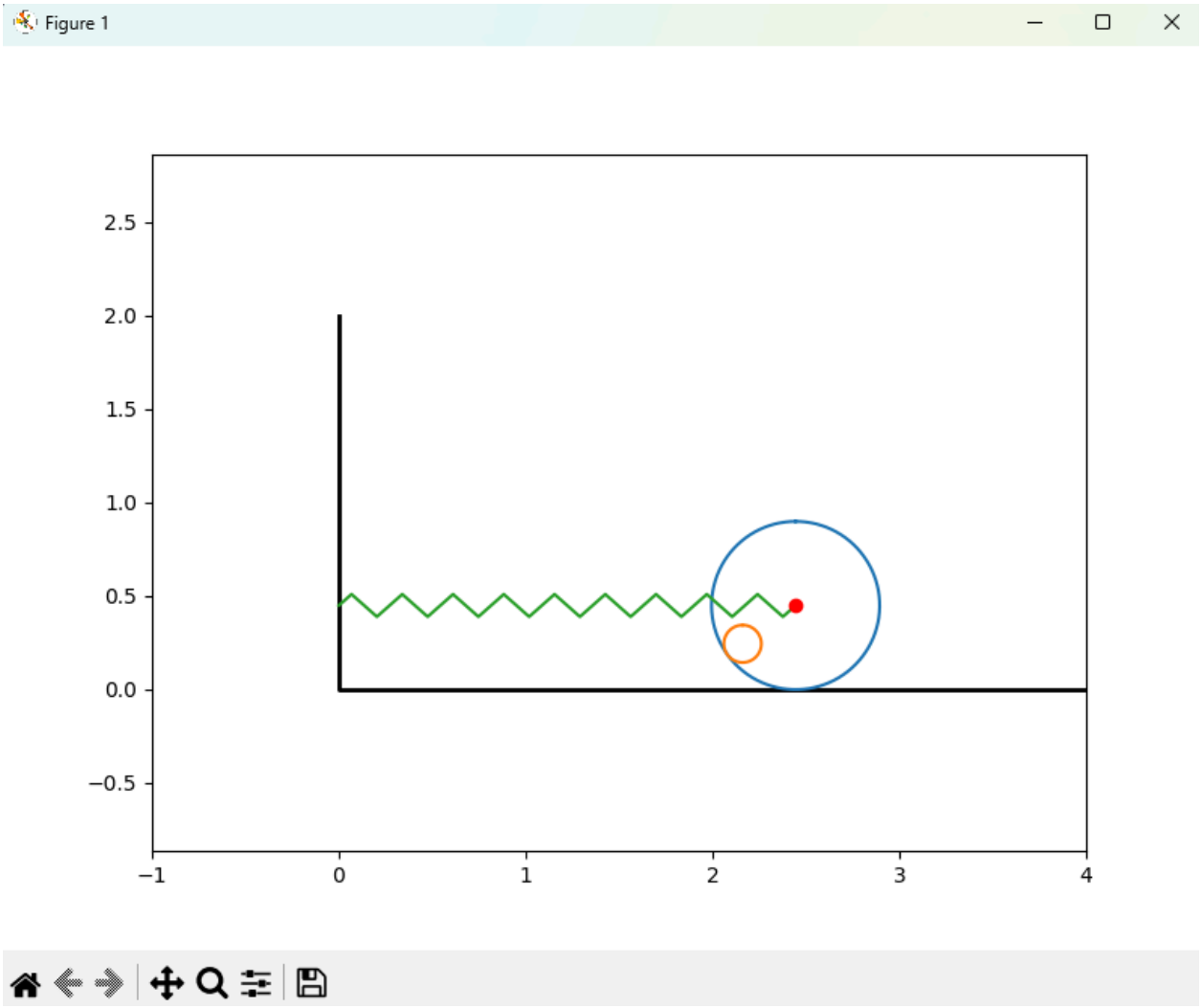
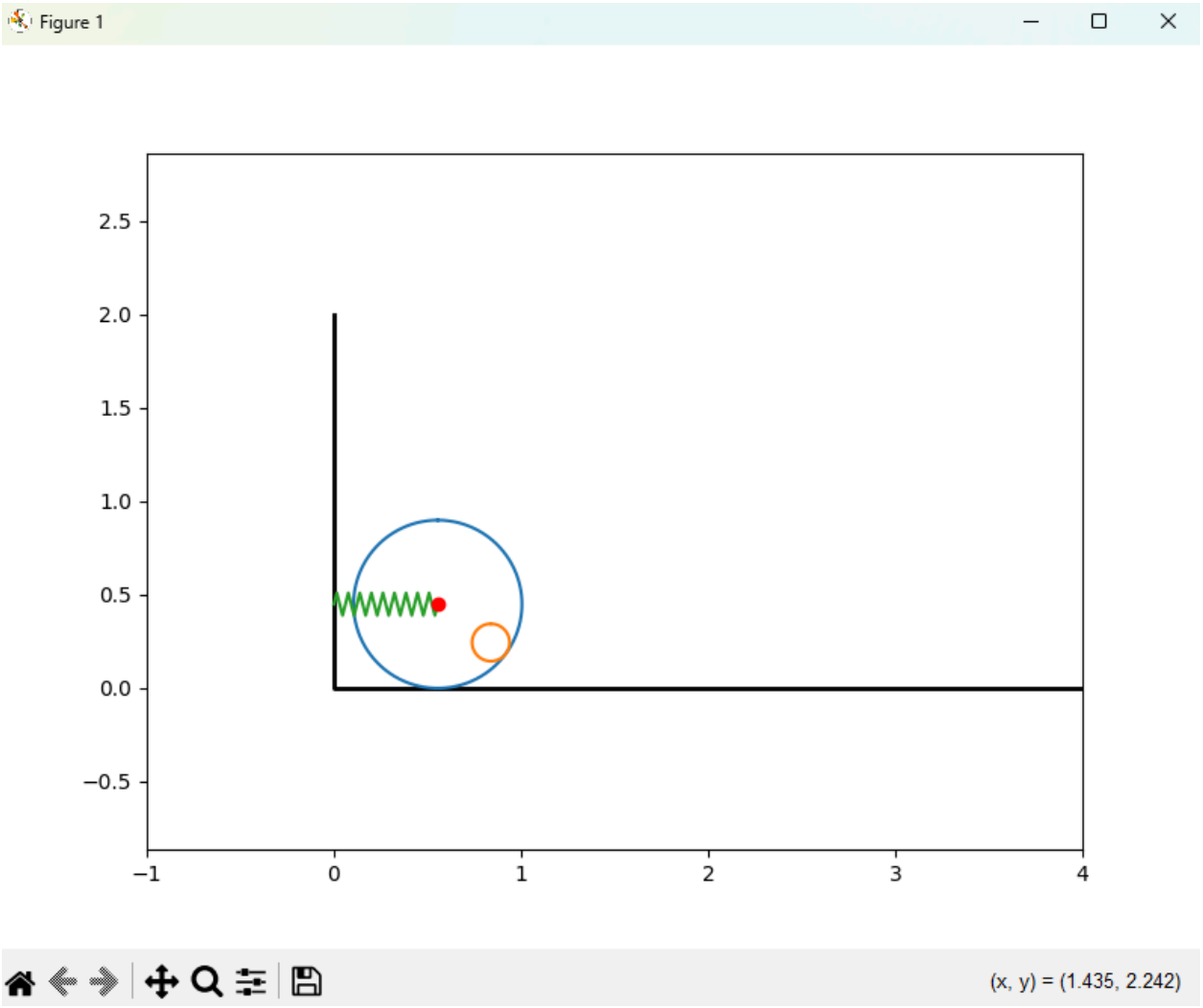


Figure 1
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1
0
1
2
3
4



Вывод:

В ходе выполнения этой лабораторной работы я написал Python код, который строит анимацию движения системы. Я реализовал два цилиндра, один из которых находится внутри другого, пружину, которая сжимается и разжимается в соответствии с данным законом, а так же продемонстрировал шарнир, за который пружина крепится к цилиндру.

Я узнал, как с помощью Matplotlib и NumPy задавать и анимировать движение системы, используя данные законы и средства языка.