

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ОТЧЕТ
О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ
«АНИМАЦИЯ СИСТЕМЫ»
ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА»
ВАРИАНТ ЗАДАНИЯ 20

Выполнил студент группы М8О-209Б-22

Концебалов О.С. _____
подпись, дата

Проверил и принял

Авдюшкин А.Н. _____
подпись, дата

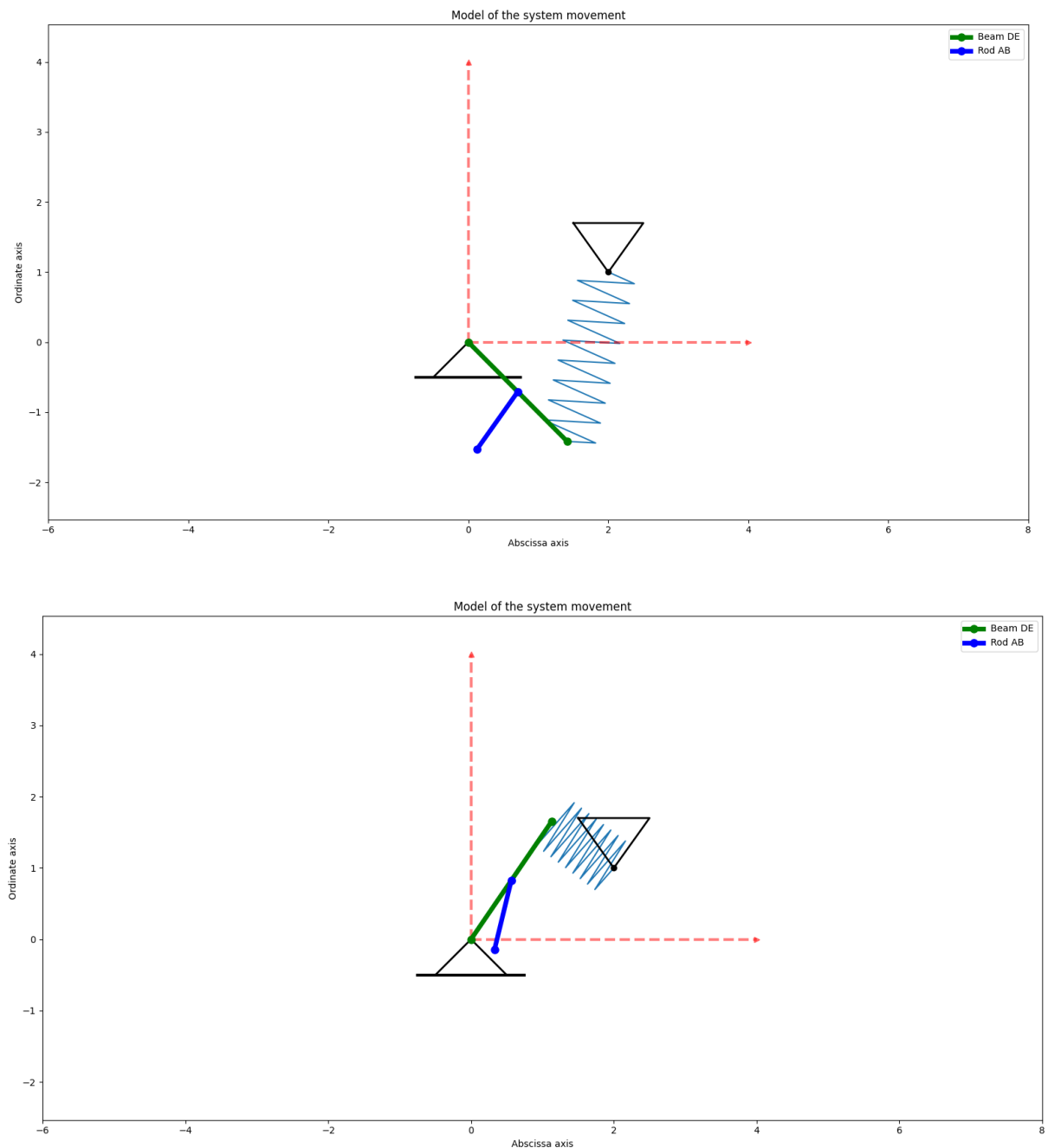
Москва, 2023

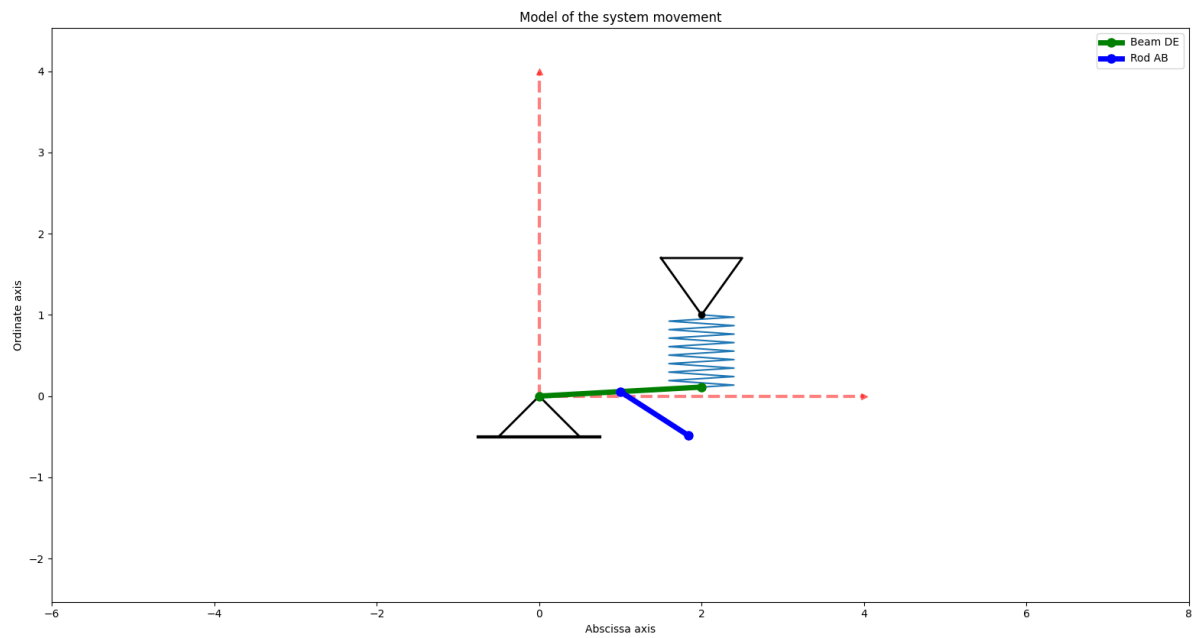
Задание: построить анимацию движения системы с помощью Python.

Задание системы 20 варианта формулируется следующим образом:

Однородная балка DE длины $2a$ и массы m_1 закреплена в неподвижном шарнире D и концом E соединена с пружиной жесткости c . К середине балки прикреплен невесомый стержень AB длины b с точечным грузом массы m_2 на конце B . Длина недеформированной пружины равна l_0 ; при горизонтальном положении балки DE пружина вертикальна.

Рисунок получившейся анимации движения системы:





Код программы

```
import numpy as np
```

```
import math
```

```
import matplotlib.pyplot as plt
```

```
from matplotlib.animation import FuncAnimation
```

```
from matplotlib.lines import Line2D
```

```
t = np.linspace(0, 120, 1000)
```

```
phi = np.sin(t)
```

```
ksi = np.cos(t)
```

```
a = 1
```

```
b = a
```

```
len_DE = 2 * a
```

```
len_AB = b
```

```
spring_height = 1
```

```
D = np.array([0, 0])
```

```
def rotate_2D(X, Y, angle) -> tuple:
```

```
    rotated_X = X * np.cos(angle) - Y * np.sin(angle)
```

```
    rotated_Y = X * np.sin(angle) + Y * np.cos(angle)
```

```
    return rotated_X, rotated_Y
```

```
K = 19
```

```
Sh = 0.4
```

```
b = 1 / (K - 2)
```

```
x_spring = np.zeros(K)
```

```
y_spring = np.zeros(K)
```

```
x_spring[0] = 0
```

```
y_spring[0] = 0
```

```
x_spring[K - 1] = 1
```

```
y_spring[K - 1] = 0
```

```
for i in range(K - 2):
```

```
    x_spring[i + 1] = b * ((i + 1) - 1 / 2)
```

```
    y_spring[i + 1] = Sh * (-1) ** i
```

```
x_E = len_DE * np.cos(phi)
```

```
y_E = len_DE * np.sin(phi)
```

```
x_B = len_AB * np.sin(ksi)
```

```
y_B = len_AB * np.cos(ksi)
```

```
x_spr = len_DE - x_E
```

```
y_spr = spring_height - y_E
```

```
spring_length = (x_spr * x_spr + y_spr * y_spr) ** 0.5
```

```
fig = plt.figure(figsize=(9, 8))
```

```
graph = fig.add_subplot(1, 1, 1)
```

```
graph.axis('equal')
```

```
graph.set_title("Model of the system movement")
```

```
graph.set_xlabel("Abscissa axis")
```

```
graph.set_ylabel("Ordinate axis")
```

```
graph.set(xlim=[-3, 5], ylim=[-3, 5])
```

```
spring_x_cord_rotated, spring_y_cord_rotated = rotate_2D(x_spring, y_spring,  
                                                         -(math.pi / 2 + abs(math.atan2(x_spr[0], y_spr[0]))))
```

```
set_spring_after_rotate, = graph.plot(spring_x_cord_rotated + len_DE,  
                                       (spring_y_cord_rotated * spring_length[0]) + spring_height)
```

```
graph.plot(2 * a, spring_height, color='black', linewidth=5, marker='o')
```

```
graph.plot([2 * a - 0.5, 2 * a + 0.5, 2 * a, 2 * a - 0.5],  
           [spring_height + 0.7, spring_height + 0.7, spring_height, spring_height + 0.7],  
           color='black', linewidth=2)
```

```
graph.plot([-0.5, 0.5, 0, -0.5], [-0.5, -0.5, 0, -0.5], color='black', linewidth=2)
```

```
graph.plot([-0.75, 0.75], [-0.5, -0.5], color='black', linewidth=3)
```

```
graph.plot([0, 0], [0, 4], color='red', linewidth=3, linestyle='dashed', alpha=0.5, marker='^')
```

```
graph.plot([0, 4], [0, 0], color='red', linewidth=3, linestyle='dashed', alpha=0.5, marker='>')
```

```
draw_DE = graph.plot(np.array([0, x_E[0]]), np.array([0, y_E[0]]),  
                    color='green', linewidth=5, label="Beam DE", marker='o', markersize=8)[0]
```

```
draw_AB = graph.plot(np.array([x_E[0] / 2, x_E[0] / 2 + x_B[0]]),  
                    np.array([y_E[0] / 2, y_E[0] / 2 - y_B[0]]),  
                    color='blue', linewidth=5, label="Rod AB", marker='o', markersize=8)[0]
```

```
graph.legend()
```

```
def animation(i) -> Line2D:
```

```
    draw_DE.set_data(np.array([0, x_E[i]]), np.array([0, y_E[i]]))  
  
    spring_x_cord_rotated, spring_y_cord_rotated = rotate_2D(x_spring * spring_length[i],  
y_spring,  
                    -(math.pi / 2 + abs(math.atan2(x_spr[i], y_spr[i]))))  
  
    set_spring_after_rotate.set_data(spring_x_cord_rotated + len_DE, spring_y_cord_rotated +  
spring_height)  
  
    draw_AB.set_data(np.array([x_E[i] / 2, x_E[i] / 2 + x_B[i]]), np.array([y_E[i] / 2, y_E[i] / 2 -  
y_B[i]]))  
  
    return draw_DE
```

```
show_movement = FuncAnimation(fig, animation, frames=len(t), interval=20, repeat=True)
```

```
animation_running = True
```

```
def animation_pause(event) -> None:
```

```
    global animation_running
```

```
    if animation_running:
```

```
        show_movement.event_source.stop()
```

```
        animation_running = False
```

```
    else:
```

```
        show_movement.event_source.start()
```

```
        animation_running = True
```

```
if __name__ == "__main__":
```

```
    fig.canvas.mpl_connect('button_press_event', animation_pause)
```

```
    plt.show()
```

Пояснения

В программе есть 3 основные функции:

1. `rotate_2D(X, Y, angle)` – выполняет поворот объекта на угол `angle` и возвращает измененные координаты. Поворот происходит с помощью аппарата линейной алгебра, а именно матрицы поворота.
2. `animation(i)` – функция смены кадра. На каждой итерации меняет положение балки и стержня, а также изменяет состояние пружины (ее угол поворота и растяжение).

3. `animation_pause(event)` – ставит анимацию на паузу, если нажать кнопку мыши.

Вывод

Я успешно выполнил лабораторную работу по теоретической механике. С помощью языка программирования Python и библиотек `matplotlib` и `numpy` я схематично проанимировал движение балки, прикрепленной к пружине, и невесомого стержня с грузом на конце, который прикреплен к середине балки.

Благодаря этой лабораторной работе, я научился работать с 2D анимацией в `matplotlib`, иллюстрировать движение системы с помощью Python и реализовал основание для выполнения следующей лабораторной работы. Моя программа полностью готова к тому, чтобы поместить в нее реальные законы движения.