ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ОТЧЕТ

О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«ДИНАМИКА 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: проинтегрировать систему дифференциальных уравнений движения системы с двумя степенями свободы с помощью средств Python. Построить графики законов движения системы и указанных в задании реакций для разных случаев системы.

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.

Дифференциальные уравнения движения системы имеют вид:

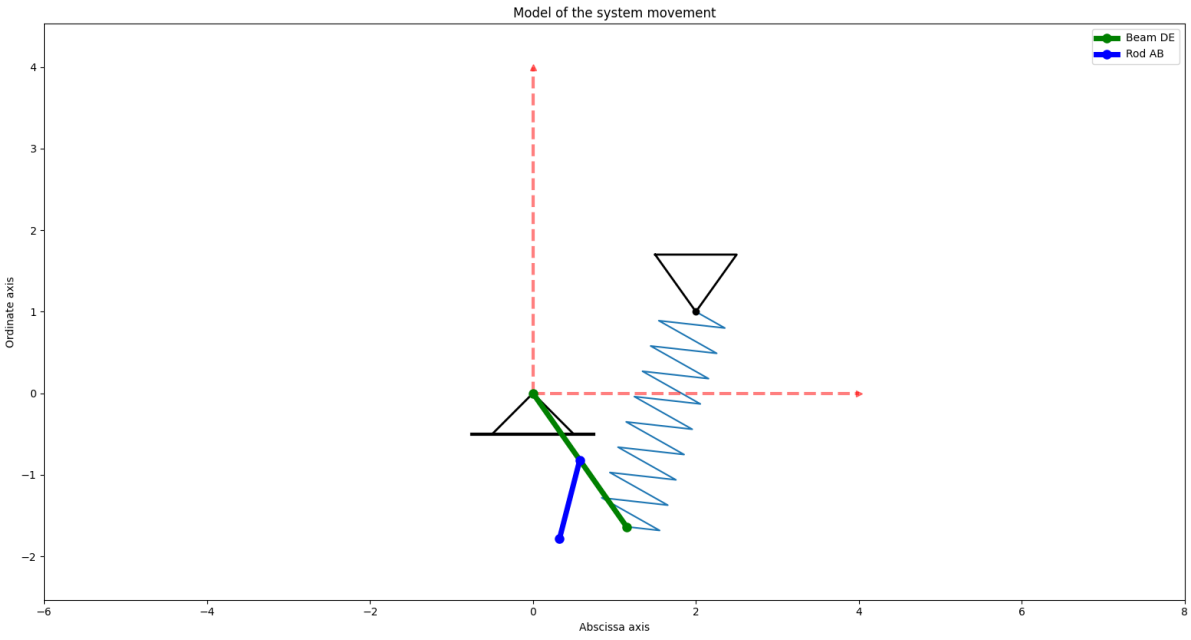
$$[(4/3)m_1 + m_2]\, a\ddot{\varphi} + m_2 b\left[\ddot{\psi}\sin(\psi-\varphi) + \dot{\psi}^2\cos(\psi-\varphi)\right] =$$

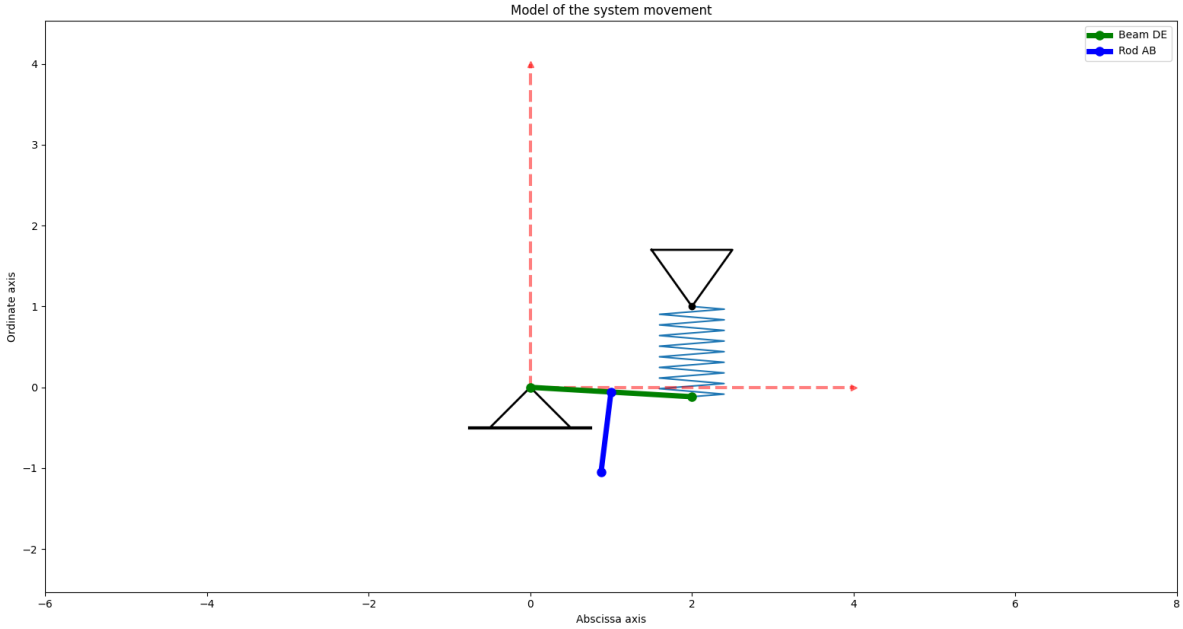
$$= -(m_1+m_2)g\cos\varphi + c\,[(\ell_0/\ell) - 1]\,(4a\sin\varphi - 2\ell_0\cos\varphi),$$

$$a\left[\ddot{\varphi}\sin(\psi-\varphi) - \dot{\varphi}^2\cos(\psi-\varphi)\right] + b\ddot{\psi} = -g\sin\psi,$$

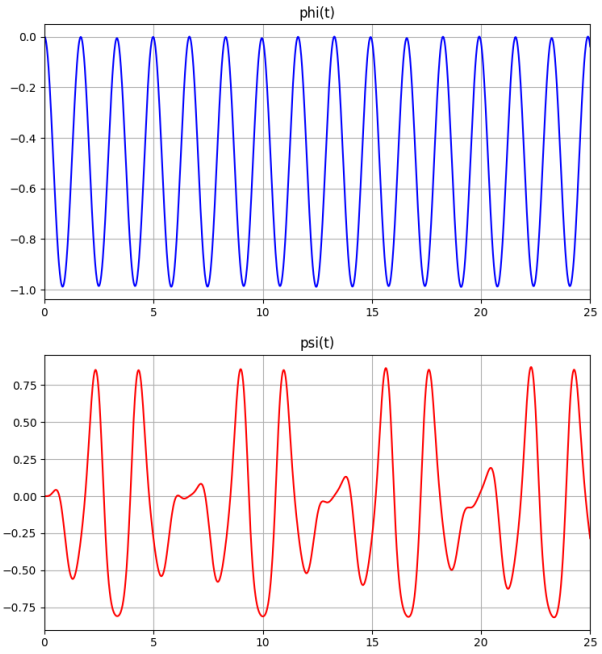
где $\ell = \sqrt{8a^2(1-\cos\varphi) + \ell_0(\ell_0 - 4a\sin\varphi)}.$

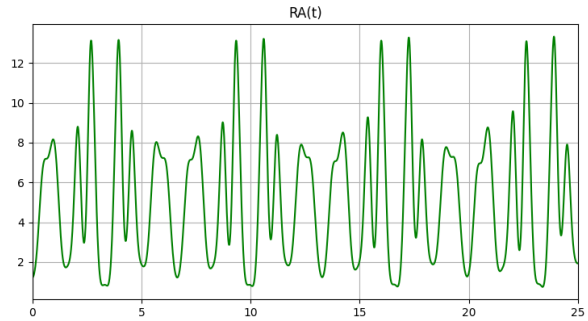
Рисунок получившейся анимации движения:





Графики φ, Ψ и сил реакции:





Код программы

```
import numpy as np
import matplotlib.pyplot as plt
import math

from matplotlib.animation import FuncAnimation
from matplotlib.lines import Line2D
from numpy import ndarray
from scipy.integrate import odeint
```

```
t_fin = 25
t = np.linspace(0, t_fin, 2001)

m1 = 50
m2 = 0.5
a = b = l0 = 1
c = 250
g = 9.8

phi0 = 0
psi0 = 0
dphi0 = dpsi0 = 0
y0 = [phi0, psi0, dphi0, dpsi0]


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


def odesys(y, t, m1, m2, a, b, l0, c, g) -> ndarray:
    l = ((8 * (a ** 2) * (1 - np.cos(y[0]))) + (l0 * (l0 - 4 * a * np.sin(y[0])))) ** 0.5
    dy = np.zeros(4)
    dy[0] = y[2]
    dy[1] = y[3]

    a11 = a * ((4 / 3) * m1 + m2)
    a12 = m2 * b * np.sin(y[1] - y[0])
    a21 = a * np.sin(y[1] - y[0])
    a22 = b

    b1 = c * ((l0 / l) - 1) * (4 * a * np.sin(y[0]) - 2 * l0 * np.cos(y[0])) - \
        ((m1 + m2) * g * np.cos(y[0])) - (m2 * b * ((y[3]) ** 2) * np.cos(y[1] - y[0]))

    b2 = (a * ((y[3]) ** 2) * np.cos(y[1] - y[0])) - (g * np.sin(y[1]))

    dy[2] = (b1 * a22 - b2 * a12) / (a11 * a22 - a12 * a21)
    dy[3] = (b2 * a11 - b1 * a21) / (a11 * a22 - a12 * a21)

    return dy


Y = odeint(odesys, y0, t, (m1, m2, a, b, l0, c, g))
```

```
phi = Y[:, 0]
ksi = Y[:, 1]
dphi = Y[:, 2]
dpsi = Y[:, 3]
ddphi = [odesys(y, t, m1, m2, a, b, l0, c, g)[2] for y, t in zip(Y, t)]
ddpsi = [odesys(y, t, m1, m2, a, b, l0, c, g)[3] for y, t in zip(Y, t)]

RA = m2 * (g * np.cos(ksi) + b * (dpsi ** 2) + a * (ddphi * np.cos(ksi - phi) + (dphi ** 2) *
np.sin(ksi - phi)))

fig_for_graphs = plt.figure(figsize=[13, 7])

ax_for_graphs = fig_for_graphs.add_subplot(2, 2, 1)
ax_for_graphs.plot(t, phi, color='Blue')
ax_for_graphs.set_title("phi(t)")
ax_for_graphs.set(xlim=[0, t_fin])
ax_for_graphs.grid(True)

ax_for_graphs = fig_for_graphs.add_subplot(2, 2, 3)
ax_for_graphs.plot(t, ksi, color='Red')
ax_for_graphs.set_title("psi(t)")
ax_for_graphs.set(xlim=[0, t_fin])
ax_for_graphs.grid(True)

ax_for_graphs = fig_for_graphs.add_subplot(2, 2, 2)
ax_for_graphs.plot(t, RA, color='Green')
ax_for_graphs.set_title("RA(t)")
ax_for_graphs.set(xlim=[0, t_fin])
ax_for_graphs.grid(True)

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 4 основные функции:

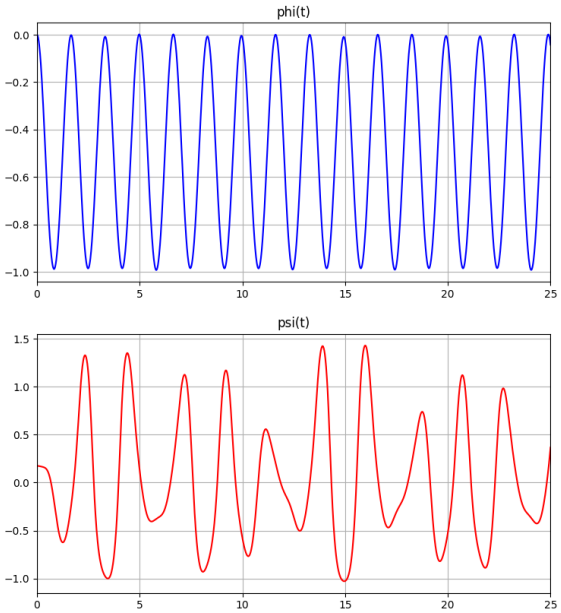
1. rotate_2D(X, Y, angle) – выполняет поворот объекта на угол angle и возвращает измененные координаты. Поворот происходит с помощью аппарата линейной алгебра, а именно матрицы поворота.
2. animation(i) – функция смены кадра. На каждой итерации меняет положение балки и стержня, а также изменят состояние пружины (ее угол поворота и растяжение).
3. animation_pause(event) – ставит анимацию на паузу, если нажать кнопку мыши.
4. odesys(y, t, m1, m2, a, b, l0, c, g) – функция для метода odesys, который решает дифференциальные уравнения

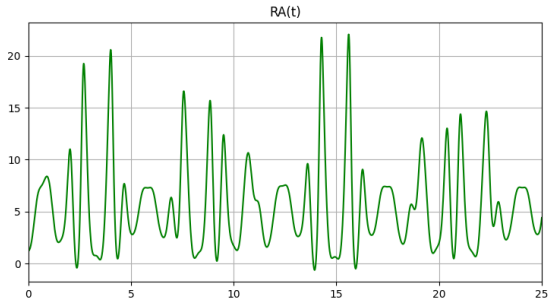
Примеры работы:

Приведу несколько примеров работы моей программы.

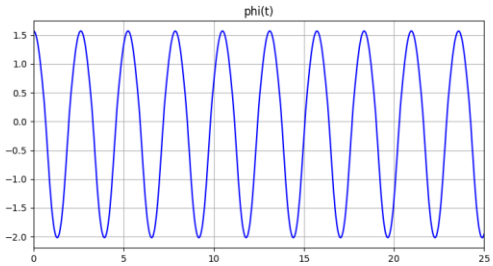
1. Начальные условия из учебника

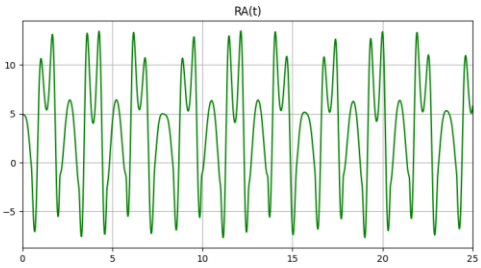
$m_1 = 50$ kg, $m_2 = 0.5$ kg, $a = b = l_0 = 1$ m, $c = 250$ N/m, $t_0 = 0$, $\varphi_0 = 0$, $\Psi_0 = \pi/18$, $\dot{\varphi}_0 = \dot{\Psi}_0 = 0$

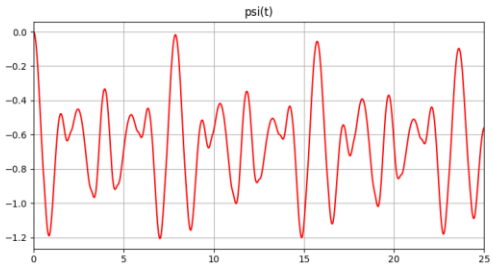




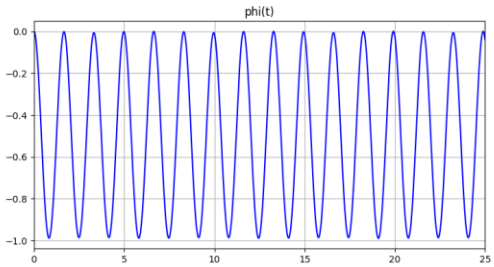
2. Все значения из учебника, кроме начального положения балки и начального угла стержня, он равен 0. Балка начинает движение с угла $\frac{\pi}{2}$

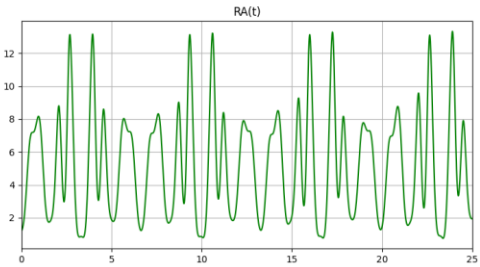


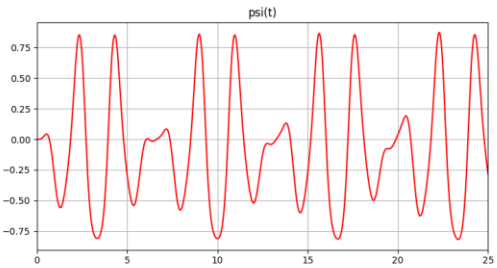




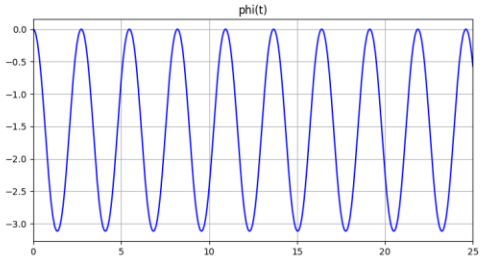
3. Все значения нулевые

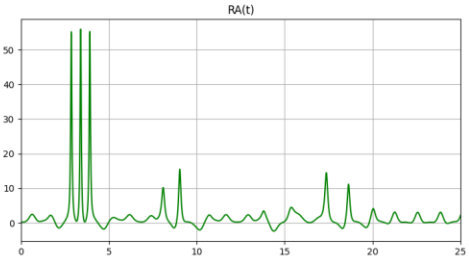


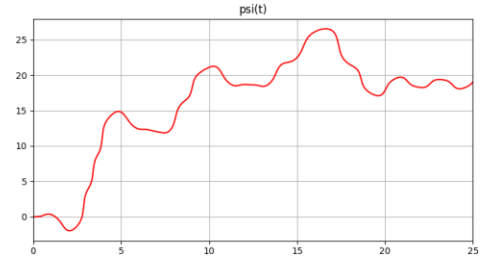




4. Масса балки 5000 кг, масса груза 0.1 кг







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.

В моей программе используются реальные законы движения, благодаря чему можно посмотреть, как эта система будет вести себя в реальной жизни.