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 №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:

Проинтегрировать систему дифференциальных уравнений движения системы с двумя степенями свободы с помощью средств Python. Построить анимацию движения системы, а также графики законов движения системы и указанных в задании реакций для разных случаев системы.

Механическая система:

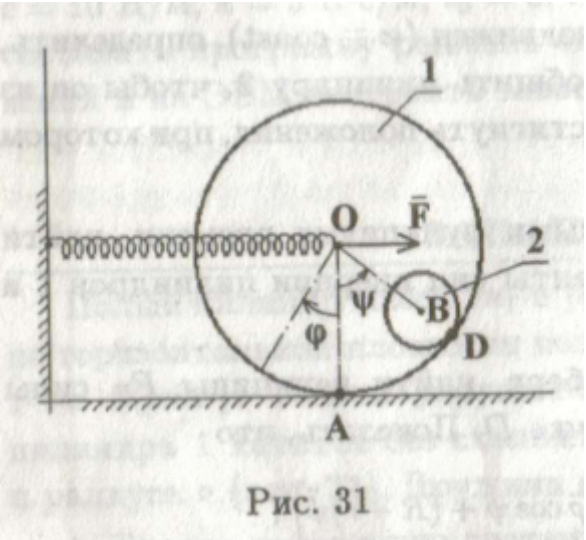


Рис. 31

Текст программы

```python
import numpy as np
import math
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from scipy.integrate import odeint

# ======================
# Функция для системы дифференциальных уравнений
# ======================
def system_of_equations(y, t, F0, M, m, c, gamma, g, R, r):
    """
    Система дифференциальных уравнений для модели.

    Параметры:
    ----------
```

```
    y      : массив состояния [phi, psi, phi_dot, psi_dot]
    t      : время
    F0     : Амплитуда внешней силы
    M      : Масса основного цилиндра (m1)
    m      : Масса вспомогательного цилиндра (m2)
    c      : Коэффициент c (по смыслу из уравнения может быть
«жёсткость»/«пружинный» коэффициент)
    gamma  : Частота внешней силы
    g      : Ускорение свободного падения
    R      : Радиус основного цилиндра
    r      : Радиус вспомогательного цилиндра
    """
    # Распакуем переменные для удобства
    phi     = y[0]
    psi     = y[1]
    phi_dot = y[2]
    psi_dot = y[3]

    # Вектор выходных значений
    dy = np.zeros_like(y)

    # Первые две компоненты — это просто производные углов
    dy[0] = phi_dot
    dy[1] = psi_dot

    # Коэффициенты матрицы (левая часть системы) согласно уравнениям
    a11 = 2 * (M + m) * R
    a12 = m * (R - r) * (1 + np.cos(psi))
    a21 = R * (1 + np.cos(psi))
    a22 = 2 * (R - r)

    # Правая часть первого уравнения:
    #  2(M + m)R * ddphi + m(R - r)[(1 + cos psi)*ddpsi - psi_dot^2 sin psi] +
cR * phi = F0 sin(gamma t)
    # => 2(M + m)R * ddphi + m(R - r)(1 + cos psi)*ddpsi
    #    = F0 sin(gamma t) - cR * phi + m(R - r)(psi_dot^2 sin psi)
    b1 = (F0 * np.sin(gamma * t)
          - c * R * phi
          + m * (R - r) * (psi_dot**2) * np.sin(psi))

    # Правая часть второго уравнения:
    #  R(1 + cos psi)*ddphi + 2(R - r)*ddpsi + g sin psi = 0
```

```
    # => R(1 + cos psi)*ddphi + 2(R - r)*ddpsi = -g sin psi
    b2 = -g * np.sin(psi)

    # Вычислим определитель матрицы
    det = a11 * a22 - a12 * a21
    if abs(det) < 1e-14:
        raise ValueError("Определитель матрицы близок к нулю — система
может не иметь уникального решения.")

    # Решаем систему:
    # [ a11  a12 ] [ ddphi ] = [ b1 ]
    # [ a21  a22 ] [ ddpsi ]   [ b2 ]
    ddphi = ( b1 * a22 - b2 * a12 ) / det
    ddpsi = ( b2 * a11 - b1 * a21 ) / det

    # Запишем найденные ускорения в dy
    dy[2] = ddphi
    dy[3] = ddpsi

    return dy


# ======================
# Константы системы
# ======================
F_amplitude = 0          # F0, Н
mass_main = 5             # M, кг
mass_sub  = 20            # m, кг
damping_coef = 0         # c, кг/с
frequency   = math.pi     # gamma, рад/с
gravity = 9.81            # g, м/с²
radius_main = 1           # R, м
radius_sub  = 0.3         # r, м

# Параметры времени
time_steps = 1000
time_final = 20
time_array = np.linspace(0, time_final, time_steps)

# Начальные условия (углы в радианах и соответствующие скорости)
initial_state = [0.0, math.pi/4, 0.0, 0.0]  # [угол_main, угол_sub, угловая
скорость_main, угловая скорость_sub]
```

```
# ======================
# Решение системы
# ======================
solution = odeint(system_of_equations, initial_state, time_array,
                  args=(F_amplitude, mass_main, mass_sub, damping_coef,
frequency, gravity, radius_main, radius_sub))

angle_main = solution[:, 0]    # Угол основного цилиндра (psi)
angle_sub  = solution[:, 1]    # Угол дополнительного цилиндра (phi)
ang_vel_main = solution[:, 2]  # Угловая скорость основного цилиндра
ang_vel_sub  = solution[:, 3]  # Угловая скорость дополнительного цилиндра

# Вычисление ускорений для графиков (по необходимости можно
использовать)
# ddx = [system_of_equations(y, t, F_amplitude, mass_main, mass_sub,
damping_coef, frequency, gravity, radius_main, radius_sub)[2] for y, t in
zip(solution, time_array)]
# ddphi = [system_of_equations(y, t, F_amplitude, mass_main, mass_sub,
damping_coef, frequency, gravity, radius_main, radius_sub)[3] for y, t in
zip(solution, time_array)]

# Вычисление дополнительных параметров для графиков
# Для соответствия второму коду, будем строить sin(angle_main) и
cos(angle_sub)
sin_angle_main = np.sin(angle_main)
cos_angle_sub = np.cos(angle_sub)

# Предположим, что "длина пружины" (или нечто аналогичное) меняется как
2.5 + angle_main + radius_main
spring_length = 2.5 + angle_main + radius_main

# Координаты основного цилиндра
X_main = spring_length
Y_main = np.full_like(X_main, radius_main)  # Y_main не меняется во времени

# Координаты дополнительного цилиндра
X_sub = X_main + (radius_main - radius_sub) * np.sin(angle_sub)
Y_sub = Y_main - (radius_main - radius_sub) * np.cos(angle_sub)

# ======================
# Графики решений
```

```python
# ======================
fig_graphs, axes = plt.subplots(2, 2, figsize=(13, 7))

# График угла основного цилиндра (psi)
axes[0, 0].plot(time_array, angle_main, color='blue')
axes[0, 0].set_title("Угол основного цилиндра (psi)")
axes[0, 0].set_xlabel("Время (s)")
axes[0, 0].set_ylabel("Угол (рад)")
axes[0, 0].grid(True)

# График угла дополнительного цилиндра (phi)
axes[1, 0].plot(time_array, angle_sub, color='red')
axes[1, 0].set_title("Угол дополнительного цилиндра (phi)")
axes[1, 0].set_xlabel("Время (s)")
axes[1, 0].set_ylabel("Угол (рад)")
axes[1, 0].grid(True)

# График синуса основного угла
axes[0, 1].plot(time_array, sin_angle_main, color='orange')
axes[0, 1].set_title("Синус основного угла (sin(psi))")
axes[0, 1].set_xlabel("Время (s)")
axes[0, 1].set_ylabel("sin(psi)")
axes[0, 1].grid(True)

# График косинуса дополнительного угла
axes[1, 1].plot(time_array, cos_angle_sub, color='black')
axes[1, 1].set_title("Косинус дополнительного угла (cos(phi))")
axes[1, 1].set_xlabel("Время (s)")
axes[1, 1].set_ylabel("cos(phi)")
axes[1, 1].grid(True)

fig_graphs.tight_layout()

# ======================
# Анимация системы
# ======================
fig_anim, ax_anim = plt.subplots(figsize=(10, 7))
ax_anim.axis('equal')
ax_anim.set(xlim=[0, 8], ylim=[-1, 7])
ax_anim.set_title("Анимация системы двух цилиндров с пружиной")

# Статические элементы (земля, опоры и т.д.)
```

```
ground = ax_anim.plot([0, 0, 6], [6, 0, 0], color='black', linewidth=2)[0]

# Подготовка данных для цилиндров
circle_theta = np.linspace(0, 2 * math.pi, 100)
X_circle_main = radius_main * np.sin(circle_theta)
Y_circle_main = radius_main * np.cos(circle_theta)
X_circle_sub  = radius_sub * np.sin(circle_theta)
Y_circle_sub  = radius_sub * np.cos(circle_theta)

# Создание объектов для анимации
main_cylinder, = ax_anim.plot([], [], color='blue', linewidth=3)  # Основной
цилиндр
sub_cylinder,  = ax_anim.plot([], [], color='red',  linewidth=2)  #
Дополнительный цилиндр
spring_line,   = ax_anim.plot([], [], color='green', linewidth=2)  # Пружина
point_main,    = ax_anim.plot([], [], 'bo', markersize=8)          # Точка
центра основного цилиндра

def create_spring(x1, y1, x2, y2, n_coils=20, amplitude=0.1, resolution=100):
    """
    Генерирует координаты пружины между двумя точками.

    Параметры:
    ----------
    x1, y1 : float
        Координаты начальной точки.
    x2, y2 : float
        Координаты конечной точки.
    n_coils : int
        Количество витков пружины.
    amplitude : float
        Амплитуда колебаний пружины.
    resolution : int
        Количество точек для построения пружины.

    Возвращает:
    ----------
    X_spring, Y_spring : ndarray
        Координаты пружины.
    """
    t = np.linspace(0, 1, resolution)
    X_straight = x1 + (x2 - x1) * t
```

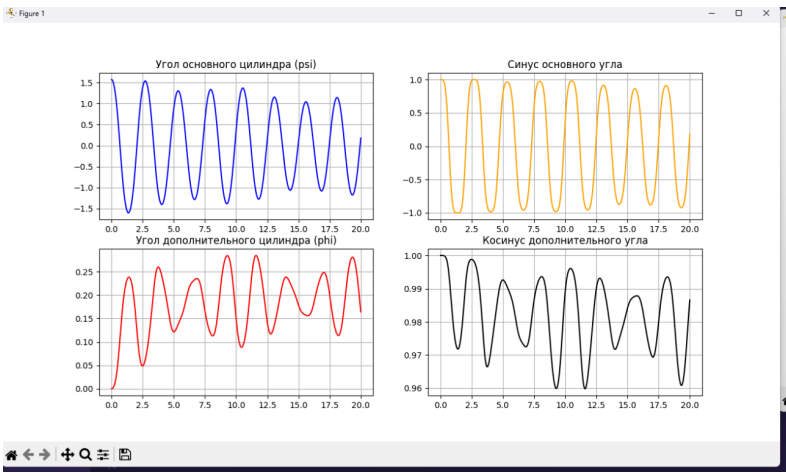
```
    Y_straight = y1 + (y2 - y1) * t

    dx = x2 - x1
    dy = y2 - y1
    length = np.hypot(dx, dy)

    perp_norm = np.array([-dy, dx])
    perp_len = np.hypot(perp_norm[0], perp_norm[1])
    if perp_len != 0:
        perp_norm /= perp_len
    else:
        perp_norm = np.array([0, 1])

    wave = amplitude * np.sin(2 * np.pi * n_coils * t)
    X_spring = X_straight + wave * perp_norm[0]
    Y_spring = Y_straight + wave * perp_norm[1]

    return X_spring, Y_spring

def animate(frame):
    """
    Анимирует кадр с номером frame.
    """
    # Координаты основного цилиндра на данном кадре
    current_X_main = X_main[frame]
    current_Y_main = Y_main[frame]

    # Координаты дополнительного цилиндра на данном кадре
    current_X_sub = X_sub[frame]
    current_Y_sub = Y_sub[frame]

    # --- Обновляем основной цилиндр ---
    main_cylinder.set_data(X_circle_main + current_X_main,
                           Y_circle_main + current_Y_main)

    # --- Обновляем дополнительный цилиндр ---
    sub_cylinder.set_data(X_circle_sub + current_X_sub,
                          Y_circle_sub + current_Y_sub)

    # --- Обновляем точку центра основного цилиндра ---
    point_main.set_data([current_X_main], [current_Y_main])
```

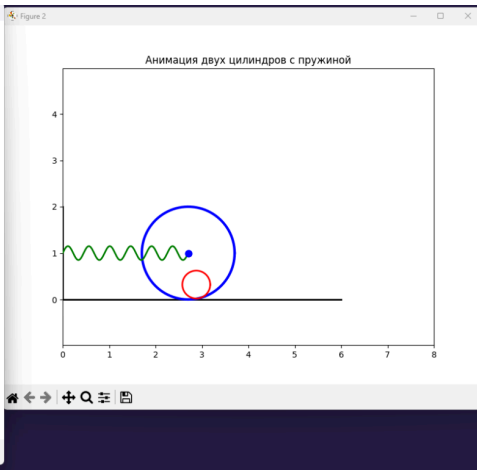
```
    # --- Генерируем координаты «пружины» ---
    spring_x, spring_y = create_spring(0, current_Y_main,
                                       current_X_main, current_Y_main,
                                       n_coils=20, amplitude=0.1,
resolution=200)
    spring_line.set_data(spring_x, spring_y)

    return main_cylinder, sub_cylinder, spring_line, point_main

# Создаем анимацию
anim = FuncAnimation(fig_anim, animate, frames=len(time_array), interval=30,
blit=True)

# Показываем графики и анимацию
plt.show()
```

Результат работы программы:

- mass_main = 5.0; mass_sub = 0.5; radius_main = 1.0; radius_sub = 0.3; F_amplitude = 3.0; frequency = math.pi; damping_coef = 5.0; gravity = 9.81;

initial_state = [math.pi / 2, 0, 0, 0]; – лёгкий дополнительный цилиндр, умеренное демпфирование:





Результат: Основной цилиндр (более тяжёлый) колеблется около вертикального положения, отклоняясь под действием внешней силы.

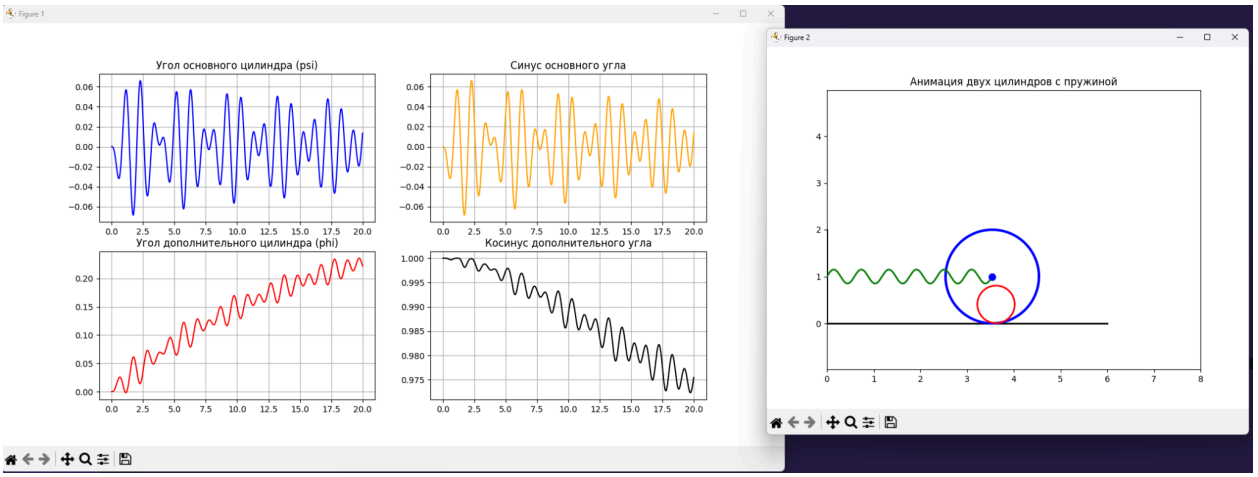
Лёгкий дополнительный цилиндр (mass_sub = 0.5) будет раскачиваться заметно сильнее и «гулять» по радиусу основного.

Демпфирование (damping_coef = 5) не слишком большое, поэтому колебания затухают медленно.

Пружина (модельно) чуть «трясет» основной цилиндр взад-вперед, создавая колебания в горизонтальном направлении.

- mass_main = 3.0; mass_sub = 5.0; radius_main = 1.0; radius_sub = 0.4; F_amplitude = 2.0; frequency = 2 * math.pi; damping_coef = 1.0; gravity = 9.81;

initial_state = [0, 0, 0, 0]; – тяжёлый «дополнительный» цилиндр, слабый демпфер:



Результат: Поскольку дополнительный цилиндр тяжелее основного, система становится нестабильной при минимальном внешнем воздействии.

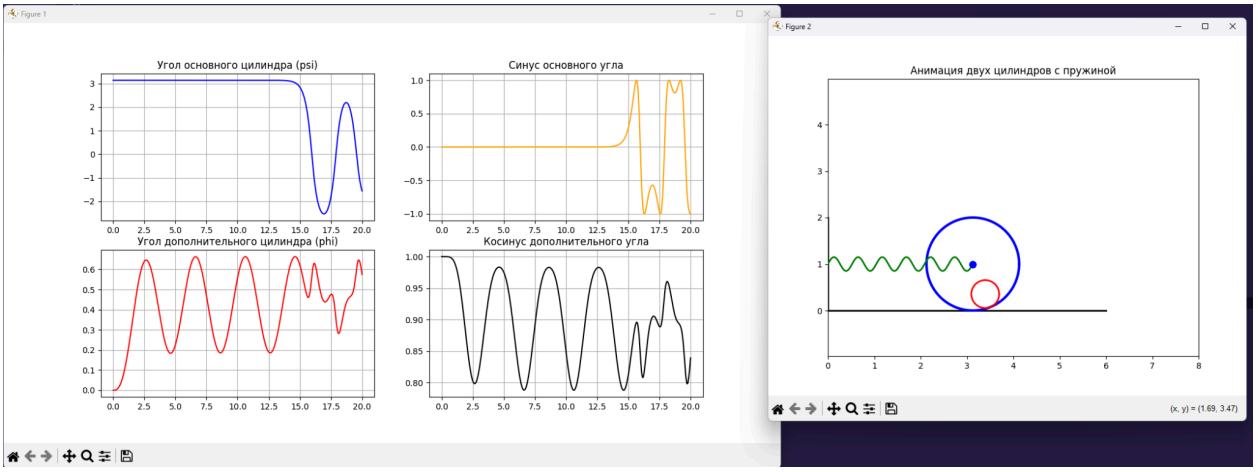
При слабом демпфировании (damping_coef = 1) пружинные колебания почти не гасятся, возможны крупные размахи.

Основной цилиндр будет стремиться вращаться, но значительная масса дополнительного может вывести систему из равновесия.

Из-за более высокой частоты внешней силы (2π) могут появиться эффекты резонанса, если система колебаний попадёт в подходящий диапазон.

- mass_main = 5.0; mass_sub = 2.0; radius_main = 1.0; radius_sub = 0.3; F_amplitude = 10.0; frequency = 2 * math.pi; damping_coef = 15.0; gravity = 9.81;

initial_state = [math.pi, 0, 0, 0] – сильное демпфирование, большая внешняя сила:



Результат: Из-за большой внешней силы (F_amplitude = 10) цилиндры будут пытаться сильно раскачиваться.

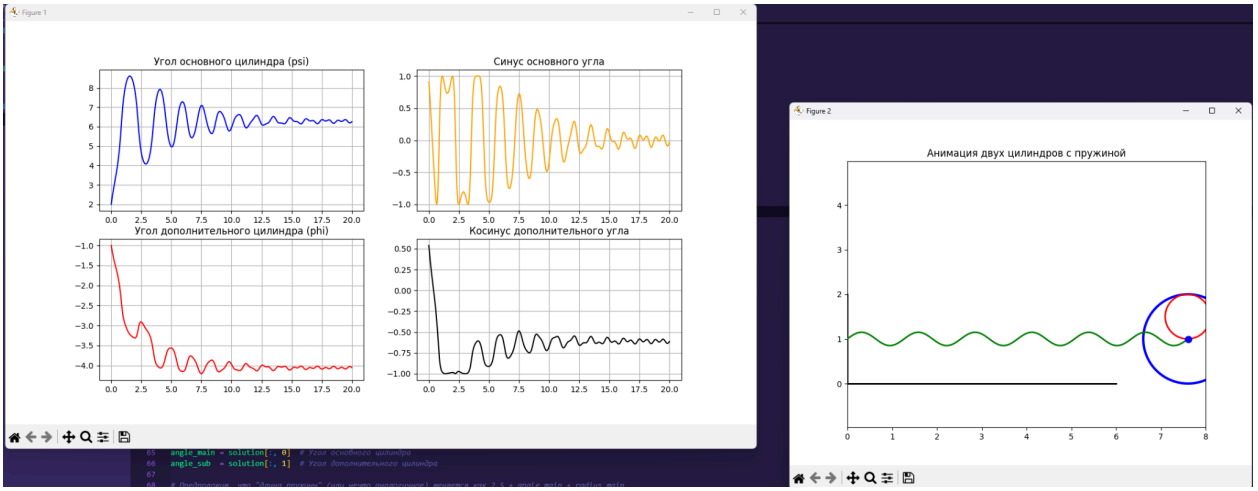
Однако высокое демпфирование (damping_coef = 15) будет быстро гасить любые колебания.

Основной цилиндр, начав с перевёрнутого положения (угол ~ 180°), будет пытаться вернуться в более устойчивое состояние (около 0 или π), совершая пару «рывков».

Дополнительный цилиндр может слегка «подвисать» или «подпрыгивать» в радиальной полости, но скоро тоже стабилизируется.

- mass_main = 1.0; mass_sub = 1.0; radius_main = 1.0; radius_sub = 0.5; F_amplitude = 4.0; frequency = 3 * math.pi; damping_coef = 2.0; gravity = 9.81;

initial_state = [2.0, -1.0, 5.0, -2.0]; – маленькая основная масса, сильные пружинные колебания, большие начальные углы:



Результат: Пример «бурных» колебаний: оба цилиндра легкие (одинаковые массы), начальные углы и скорости заданы так, что сразу возникает сильная динамика.

Высокая частота внешней силы (3π) может «разгонять» систему и приводить к резким пульсациям вокруг оси.

Небольшой коэффициент демпфирования (2.0) позволяет этим колебаниям довольно долго сохранять энергию, хотя и будет заметное затухание с течением времени.

Пружина визуально будет сильно «шевелиться», переходя из сжатого в растянутое состояние, цилиндры могут делать почти полный оборот вокруг своих центров.

Вывод:

В ходе выполнения этой лабораторной работы я написал Python код, строящий анимацию движения системы, уравнения движения которой могут быть модифицированы путем изменения коэффициентов (начальных значений).

Также я реализовал 4 графика, которые отображают изменение x(t), phi(t), FA(t) (силы трения) и NA(t) (силы давления системы на плоскость).