ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ОТЧЕТ

О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«ДИНАМИКА СИСТЕМЫ»

ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА И ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ»

ВАРИАНТ ЗАДАНИЯ № 14

Выполнил(а) студент группы М8О-208Б-20

Марков И. И. ______________________________

подпись, дата

Проверил и принял

Зав. каф. 802, Бардин Б.С.______________________________

подпись, дата

с оценкой ______________________________

Москва, 2022

Лабораторная работа 3.

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. Исследовать на устойчивость.

1. Схема программы

Для решения поставленных задач требуется сделать следующие шаги:

1. Отдельно от основной программы с помощью уравнений движения системы требуется сформировать функцию, которая будет принимать в себя значения $(q_1, q_2, \dot{q_1}, \dot{q_2})$, а на выход вернёт значения $(\dot{q_1}, \dot{q_2}, \ddot{q_1}, \ddot{q_2})$.
2. В основной программе требуется задать значения всех параметров, начальное положение системы, и запустить процедуру численного интегрирования системы.
3. Результаты численного интегрирования системы далее следует использовать при построении анимации движения системы.

2. Составление функции уравнений движения

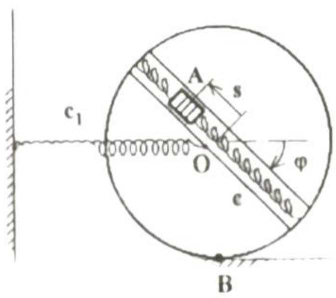


Рис. 1

Для того, чтобы узнать законы движения по координатам s, φ, необходимо определить функцию f , которая будет принимать на вход вектор состояния $(s, \varphi, \dot{s}, \dot{\varphi})$и момент времени t . Значением такой функции будет вектор $(\dot{s}, \dot{\varphi}, \ddot{s}, \ddot{\varphi})$.

Для определения такой функции воспользуемся уравнением движения системы, являющейся уравнением Лагранжа II рода:

$$\left[(2m_1 + m)r^2 + ms^2 + 2mrs \cdot sin(\varphi)\right]\ddot{\varphi} - mr(\ddot{s} - s\dot{\varphi}^2) + 2m(s + r \cdot sin(\varphi))\dot{s}\dot{\varphi} = -c_1 r^2 \varphi - mgs \cdot cos(\varphi)$$

$$-r \cdot cos(\varphi)\ddot{\varphi} + \ddot{s} - s\dot{\varphi}^2 = -2\frac{cs}{m} - g \cdot sin(\varphi)$$

Функция имеет вид:

```
def odesys(y, t, m1, m2, c, c1, R, g):
    dy = np.zeros(4)
    dy[0] = y[2]
    dy[1] = y[3]
```

```
    a11 = -m2 * R * np.cos(y[1])
    a12 = ((2 * m1 + m2) * R**2 + m2 * y[0] + 2 * R * m2 * y[0] * np.sin(y[1]))
    a21 = 1
    a22 = -R * np.cos(y[1])

    b1 = -m2 * R * y[0] * y[3]**2 * np.cos(y[1]) - 2 * m2 * (y[0] + R * np.sin(y[1])) * y[2] * y[3] - c1 *
R**2 * y[1] - m2 * g * y[0] * np.cos(y[1])
    b2 = y[0] * y[3]**2 - 2 * (c/m2) * 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
```

На вход функции `odesys` приходит вектор `y`, представленный в виде столбца, реализованного в библиотеке numpy. В качестве параметра `t` передается момент времени. Параметры `m1, m2, c, c1, R, g` являются константами системы и обозначают массу кольца, груза, жесткость пружин, радиус кольца и ускорение свободного падения.

3. Численное интегрирование системы уравнений

Следующим шагом является численное интегрирование функции odesys. Для этого я использовал функцию из библиотеки scipy под названием odeint. Она производит численное интегрирование функции по указанным параметрам. Итого, численное интегрирование выглядит следующим образом:

```
m1 = 1
m2 = 0.5
R = 1
g = 9.81
c = c1 = 5
t_fin = 20
t = np.linspace(0, t_fin, 1001)
s0 = 0
phi0 = np.pi/2
ds0 = 0
dphi0 = 1
y0 = [s0, phi0, ds0, dphi0]
Y = odeint(odesys, y0, t, (m1, m2, c, c1, R,  g))
s = Y[:, 0]
phi = Y[:, 1]
ds = Y[:, 2]
dphi = Y[:, 3]
```

Где `y0` содержит начальное состояние системы.

4. Построение графиков

Построим графики решения решения $s(t)$, $\varphi(t)$, $\dot{s}(t)$, $\dot{\varphi}(t)$, а также $\ddot{\varphi}(t), \ddot{\psi}(t)$. Для построения графиков я использовал библиотеку `pyplot`. Итого, код программы, отрисовывающей графики, следующий

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

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

ax_for_graphs = fig_for_graphs.add_subplot(2,2,4)
ax_for_graphs.plot(t,dphi,color='black')
ax_for_graphs.set_title('phi\'(t)')
ax_for_graphs.set(xlim=[0,t_fin])
ax_for_graphs.grid(True)
```

5. Построение анимации

Теперь построим анимацию движения системы, используя полученные результаты. Оформим графическое окно, создадим нарисованные объекты в начальном положении и запустим цикл, переставляющий объект в положения, соответствующим новым моментам времени

```
def spring(k, h, w):
    x = np.linspace(0, h, 100)
    return np.array([
        x,
        np.sin(2 * pi / (h / k) * x) * w
    ])

ring_dots_x_tmp = R * np.cos(angles)
ring_dots_y_tmp = R * np.sin(angles)

box_x_tmp = np.array([-box_h / 2, -box_h / 2, box_h / 2, box_h / 2, -box_h / 2])
box_y_tmp = np.array([-box_w / 2, box_w / 2, box_w / 2, -box_w / 2, -box_w / 2])

ring_dots_x = np.zeros([len(t), len(angles)])
ring_dots_y = np.zeros([len(t), len(angles)])

box_dots_x = np.zeros([len(t), 5])
box_dots_y = np.zeros([len(t), 5])

spring_a_x = np.zeros([len(t), 100])
spring_a_y = np.zeros([len(t), 100])

spring_b_x = np.zeros([len(t), 100])
spring_b_y = np.zeros([len(t), 100])

spring_c_x = np.zeros([len(t), 100])
spring_c_y = np.zeros([len(t), 100])

for i in range(len(t)):
```

```
    ring_x = x0 + phi[i] * R
    ring_y = R

    ring_dots_x[i] = np.cos(phi[i]) * ring_dots_x_tmp + np.sin(phi[i]) * ring_dots_y_tmp + ring_x
    ring_dots_y[i] = - np.sin(phi[i]) * ring_dots_x_tmp + np.cos(phi[i]) * ring_dots_y_tmp + ring_y

    bx = box_x_tmp - s[i]
    by = box_y_tmp
    box_dots_x[i] = np.cos(phi[i]) * bx + np.sin(phi[i]) * by + ring_x
    box_dots_y[i] = - np.sin(phi[i]) * bx + np.cos(phi[i]) * by + ring_y

    spring_a_x[i] = spring(5, ring_x, 0.2)[0]
    spring_a_y[i] = spring(5, ring_x, 0.2)[1] + ring_y

    b_x = R - spring(10, R + s[i] - box_h / 2, 0.2)[0]
    b_y = spring(10, R - s[i], 0.2)[1]
    spring_b_x[i] = np.cos(phi[i]) * b_x + np.sin(phi[i]) * b_y + ring_x
    spring_b_y[i] = -np.sin(phi[i]) * b_x + np.cos(phi[i]) * b_y + ring_y

    c_x = spring(10, R - s[i] - box_h / 2, 0.2)[0] - R
    c_y = spring(10, R - s[i], 0.2)[1]
    spring_c_x[i] = np.cos(phi[i]) * c_x + np.sin(phi[i]) * c_y + ring_x
    spring_c_y[i] = -np.sin(phi[i]) * c_x + np.cos(phi[i]) * c_y + ring_y

fig = plt.figure() # создаем холст, на котором будем рисовать фигуры
ax = fig.add_subplot(1, 1, 1)
ax.axis("equal")

surface = ax.plot([0, 0, 10], [10, 0, 0]) # пол и стена
ring, = ax.plot(ring_dots_x[0], ring_dots_y[0]) # кольцо
box, = ax.plot(box_dots_x[0], box_dots_y[0]) # груз
spring_a, = ax.plot(spring_a_x[0], spring_a_y[0]) # горизонтальная пружина
spring_b, = ax.plot(spring_b_x[0], spring_b_y[0]) # нижняя пружина
spring_c, = ax.plot(spring_c_x[0], spring_c_y[0]) # верхняя пружина

def animate(i):
    ring.set_data(ring_dots_x[i], ring_dots_y[i])
    box.set_data(box_dots_x[i], box_dots_y[i])
    spring_a.set_data(spring_a_x[i], spring_a_y[i])
    spring_b.set_data(spring_b_x[i], spring_b_y[i])
    spring_c.set_data(spring_c_x[i], spring_c_y[i])

    return ring, box, spring_a, spring_b, spring_c


animation = FuncAnimation(fig, animate, frames=len(t), interval=15)
plt.show()
```

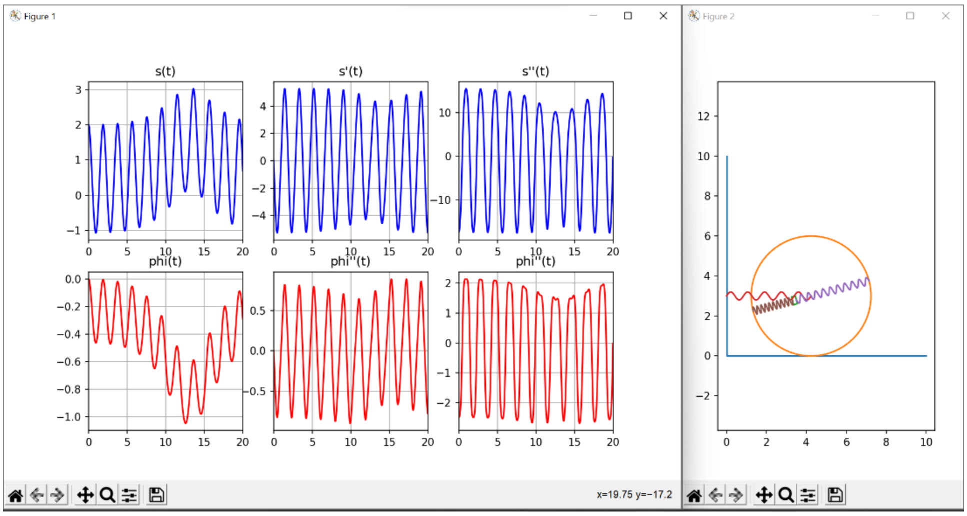
Функция `spring` является шаблоном для генерации точек пружины высотой `w`, шириной `h` и количеством витков `k`. Первым шагом я создал пустые массивы, которые в ходе выполнения цикла заполнял необходимыми значениями. После чего, используя заполненные массивы я создал графические объекты и анимировал их, реализовав функцию `animate`, выполняющую правила перехода к новому кадру анимации.

6. Результат работы программы

Выведем полученные графики работы программы

1. $m_1 = 2,\ m_2 = 1,\ r\ = 3,\ c\ = 5,\ c_1\ = 5,\ \varphi_0\ = 0,\ s_0 = 2,\ \dot{\varphi}_0\ = 0,\ \dot{s}_0\ = 0$

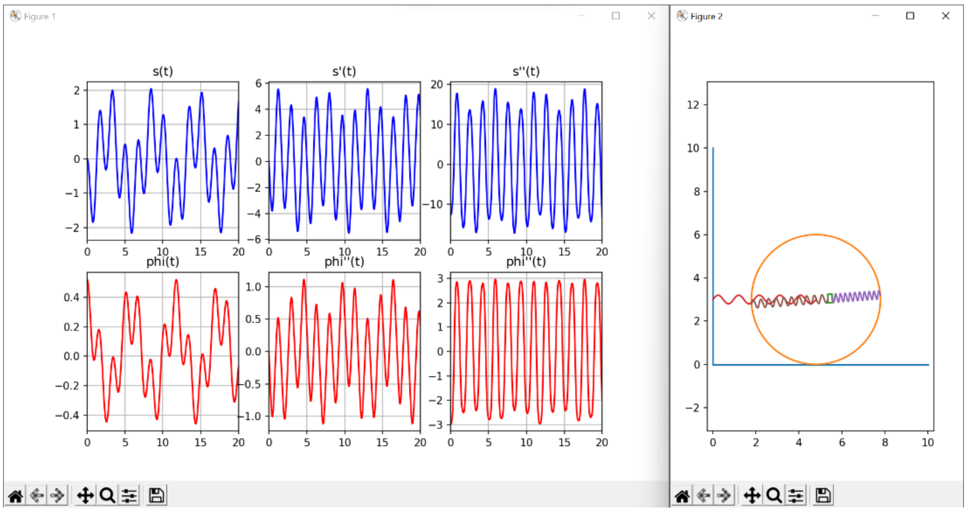
Кольцо имеет радиус 1, коэффициент упругости всех пружин одинаковый. Масса груза превышает массу кольца в 2 раза. В начальный момент времени кольцо не повернуто, горизонтальная пружина не деформирована, а груз отклонен.



Результат: Брусок совершает колебания, центр равновесия которых со временем изменяется (что видно по среднему значению графика $s(t)$. Также, колебания бруска затрагивают и само кольцо, которое начинает также совершает колебательное движение.

2. $m_1 = 2,\ m_2 = 2,\ r\ = 3,\ c\ = 5,\ c_1\ = 20,\ \varphi_0\ = \frac{\pi}{6},\ s_0 = 0,\ \dot{\varphi}_0\ = 0,\ \dot{s}_0\ = 0$

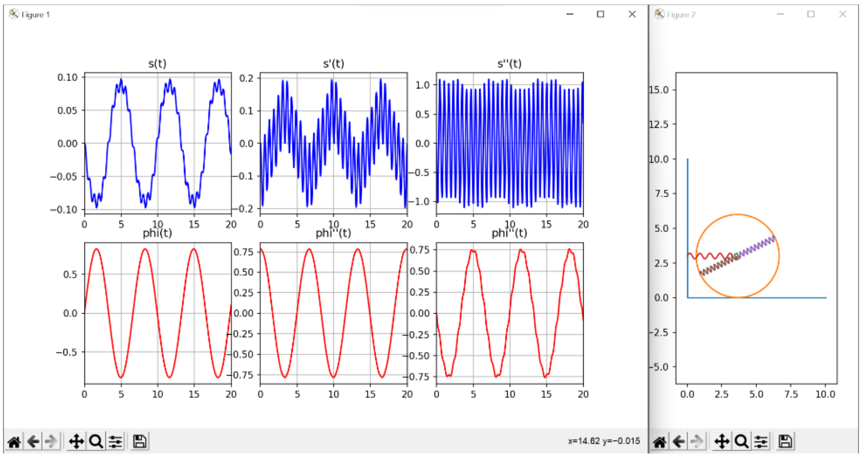
В начальный момент времени кольцо отклонено на малый угол. Обе пружины, находящие в кольце, в положении равновесия, Коэффициент упругости пружин в 4 раза больше по сравнению с прошлым опытом.



Результат: Кольцо совершает колебательные движения. Однако, они затрагивает и груз, из-за чего он также совершает колебательные движения. В ходе движения, кольцо останавливается в один момент времени.

3. $m_1 = 10,\ m_2 = 2,\ r = 3,\ c = 100,\ c_1 = 20,\ \varphi_0 = 0,\ s_0 = 0,\ \dot{\varphi}_0 = \frac{\pi}{4},\ \dot{s}_0 = 0$

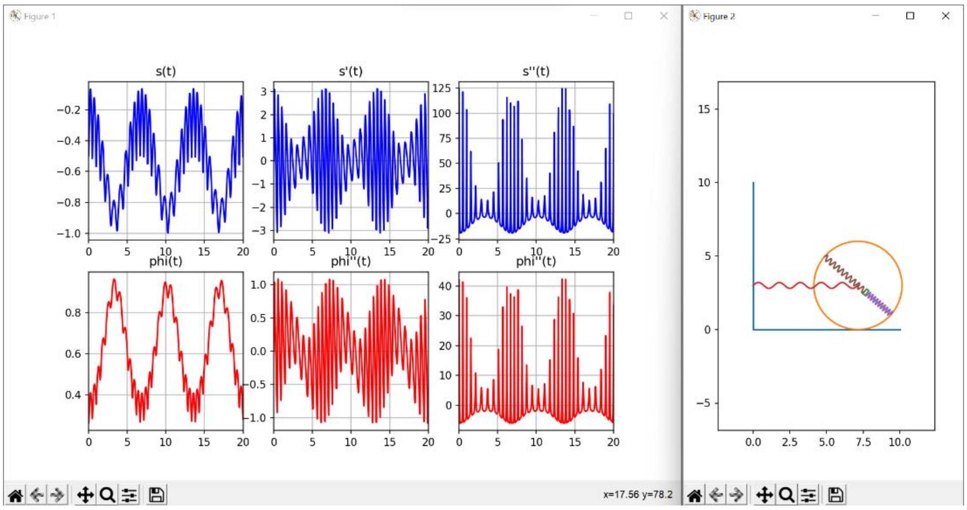
Массивное по сравнению с грузом кольцо, у которого ненулевая начальная угловая скорость. Коэффициенты упругости пружин внутри кольца имеют значительно большие значения.



Результат: Кольцо совершает колебательные движения. Груз же совершает колебания с крайне малой амплитудой, что практически не сказывается на движении кольца из-за большой массы последнего.

4. $m_1 = 1,\ m_2 = 20,\ r = 3,\ c = 100,\ c_1 = 20,\ \varphi_0 = \frac{\pi}{12},\ s_0 = -\frac{1}{2},\ \dot{\varphi}_0 = 0,\ \dot{s}_0 = 0$

Груз имеет значительно большую массу по сравнению с кольцом. Коэффициент упругости пружин внутри кольца имеет большое значение. В начальный момент времени кольцо и груз отклонены.



Результат: Вся система совершает прерывистые движения, циклически ускоряясь и резко останавливаясь.

7. Исследование на устойчивость

$$T = T_0 + T_1 + T_2$$
$$T_0 = 0$$
$$T_1 = 0$$
$$T_2 = T = m_1 r^2 \dot{\varphi}^2 + \frac{1}{2} m \left[\dot{s}^2 + \dot{\varphi}^2 r^2 + \dot{\varphi}^2 s^2 + 2\dot{\varphi} r (\dot{\varphi} s sin(\varphi) - \dot{s} cos(\varphi)) \right]$$

$$\Pi = \frac{1}{2} c_1 \varphi^2 r^2 + cs^2 + mg \cdot sin(\varphi)$$

$$\Pi_* = \Pi - T_0 = \Pi$$

$$\frac{\partial \Pi_*}{\partial s} = 2cs + mg \cdot sin(\varphi)$$
$$\frac{\partial \Pi_*}{\partial \varphi} = c_1 r^2 \varphi + mgs \cdot cos(\varphi)$$

$$\frac{\partial \Pi_*}{\partial s} = 2cs + mg \cdot sin(\varphi) = 0 \Rightarrow s_0 = -\frac{mg}{2c} sin(\varphi)$$
$$\left. \frac{\partial^2 \Pi_*}{\partial s^2} \right|_{s_0} = 2c|_{s_0} > 0$$

$$\frac{\partial \Pi_*}{\partial \varphi} = c_1 r^2 \varphi + mgs \cdot cos(\varphi) = 0 \Rightarrow$$
$$\text{при } s = 0 \ : \ c_1 r^2 \varphi = 0 \Rightarrow \varphi_0 = 0$$
$$\left. \frac{\partial^2 \Pi_*}{\partial \varphi^2} \right|_{\varphi_0} = r^2 c > 0$$

Начальные положения $\varphi_0 = 0$, $s_0 = 0$, $\dot{\varphi}_0 = 0$, $\dot{s}_0 = 0$ - устойчивый