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 №30

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

Ханнанов Руслан Маратович_____
подпись, дата

Проверил и принял

Доцент каф. 802, Чекина Е.А._____
подпись, дата

с оценкой _____

Москва, 2021

Лабораторная работа 3.

Симуляция движения системы с двумя степенями свободы

Задание: Проинтегрировать систему дифференциальных уравнений движения системы с двумя степенями свободы с помощью средств Python. Построить анимацию движения системы, а также графики законов движения.

Вариант 30.

Механическая система:

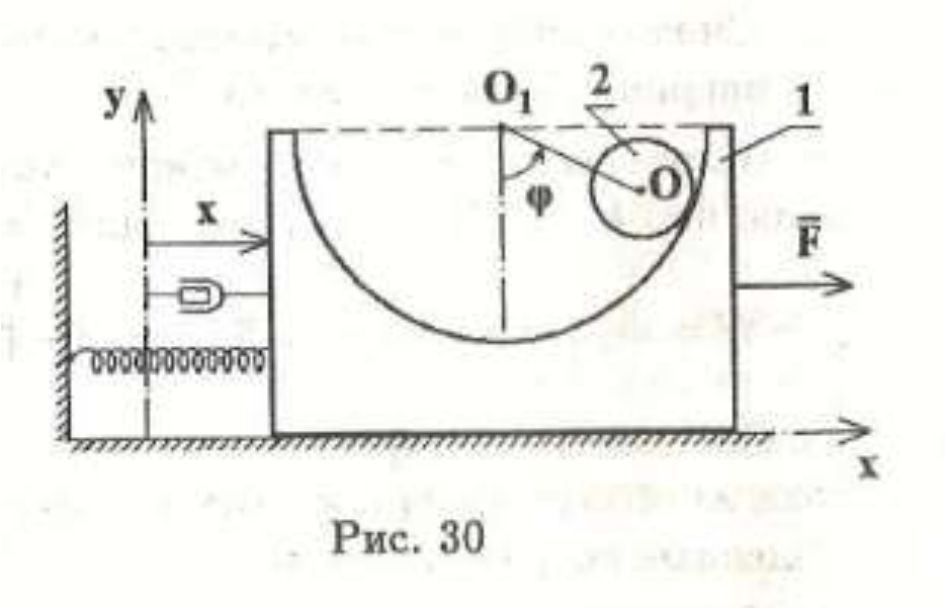


Рис. 30

Алгоритм

1) Определить количество степеней свободы системы n.

2) В соответствии с количеством степеней свободы системы, полученным в п. 1, ввести соответствующее число обобщенных координат $(q_1, q_2, \dots, q_n)$.

3) Посчитать кинетическую энергию в зависимости от обобщенных координат и обобщенных скоростей $T(q_1, \dots, q_n, \dot{q}_1, \dots, \dot{q}_n)$.

4) Найти обобщенные силы, которые соответствуют обобщенным координатам $(Q_1, \dots, Q_n)$.

5) Составить n уравнений Лагранжа второго рода. Общий вид i-го уравнения Лагранжа второго рода выглядит следующим образом:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) - \frac{\partial T}{\partial q_i} = Q_i.$$

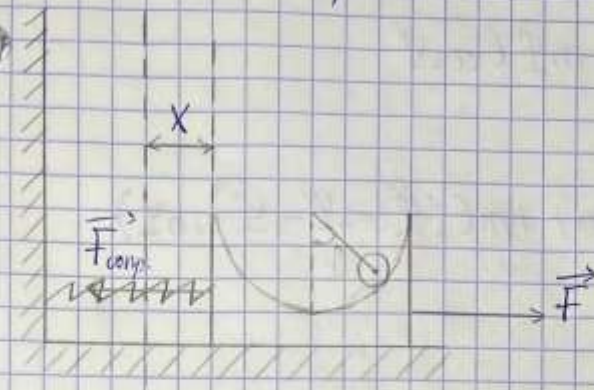
Если система консервативна, то можно упростить вид уравнения, используя функцию Лагранжа $L = T - П$.

Тогда i-е уравнение Лагранжа второго рода для консервативной системы будет выглядеть следующим образом:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0.$$

Вывод уравнений:

Уравнения Лагранжа



$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i^*$$

Рампа

$$T_р = \frac{M\dot{x}^2}{2}$$

$$l = (R - r)$$

Цилиндр

$$T_ц = \frac{m v_ц^2}{2} + \frac{\dot{\varphi}^2 l^2}{r^2}\cdot\frac{mr^2}{4}$$

($\dot{\varphi}^2$: ω^2; $\frac{mr^2}{4}$: $\frac{I}{2}$)

$$v_ц^2 = \dot{x}^2 + \dot{\varphi}^2 l^2 + 2\dot{x} l \dot{\varphi}\cos\varphi$$

$$T_ц = \frac{m}{2}\left(\dot{x}^2 + \frac{3}{2}\dot{\varphi}^2 l^2 + 2\dot{x}\dot{\varphi} l\cos\varphi\right)$$

$$\boxed{T} = T_р + T_ц = \frac{M\dot{x}^2}{2} + \frac{m}{2}\left(\dot{x} + \frac{3}{2}\dot{\varphi}^2 l^2 + 2\dot{x}\dot{\varphi} l\cos\varphi\right)$$

$$\boxed{П} = П_{mg} + П_{пруж} = (1-\cos\varphi) mg l + \frac{cx^2}{2}$$

$$\boxed{L} = T - П = \dot{x}^2\left(\frac{M}{2} + \frac{m}{2}\right) + \dot{\varphi}^2\left(\frac{3ml^2}{4}\right) + m\dot{x}\dot{\varphi} l\cos\varphi - (1-\cos\varphi) mg l - \frac{cx^2}{2}$$

[X]:

$$\frac{\partial L}{\partial \dot{x}} = \dot{x}(M+m) + m\dot{\varphi}\ell\cos\varphi$$

$$\frac{d}{dt}(\searrow) = \ddot{x}(M+m) + m\ell(\ddot{\varphi}\cos\varphi - \dot{\varphi}^2\sin\varphi)$$

$$\frac{\partial L}{\partial x} = -cx$$

$$\boxed{(M+m)\ddot{x} + m\ell(\ddot{\varphi}\cos\varphi - \dot{\varphi}^2\sin\varphi) + cx = 0}$$

[φ]:

$$\frac{\partial L}{\partial \dot{\varphi}} = \dot{\varphi}\cdot\frac{8m\ell^2}{2} + m\dot{x}\ell\cos\varphi$$

$$\frac{d}{dt}(\searrow) = \ddot{\varphi}\,\frac{8m\ell^2}{2} + m\ell(\ddot{x}\cos\varphi - \dot{x}\dot{\varphi}\sin\varphi)$$

$$\frac{\partial L}{\partial \varphi} = -m\dot{x}\dot{\varphi}\ell\sin\varphi - \sin\varphi\, mg\ell$$

$$\boxed{\frac{8}{2}\ell\ddot{\varphi} + \ddot{x}\cos\varphi + \sin\varphi\, mg = 0}$$

$$\begin{cases} Q_x^* = F_0\sin pt - k\dot{x} \\ Q_\varphi^* = 0 \end{cases} \Rightarrow$$ второе уравнение остается таким же,

а первое:

$$\boxed{(M+m)\ddot{x} + m\ell(\ddot{\varphi}\cos\varphi - \dot{\varphi}\sin\varphi) + cx + k\dot{x} = F_0\sin pt}$$

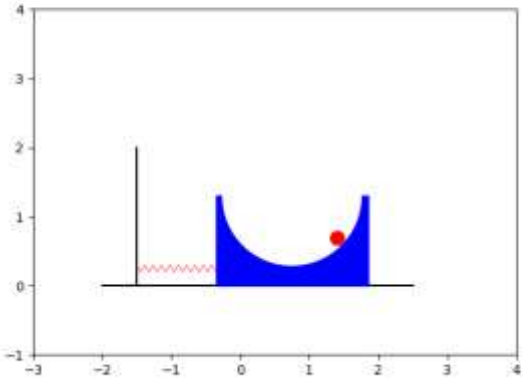
Код программы:

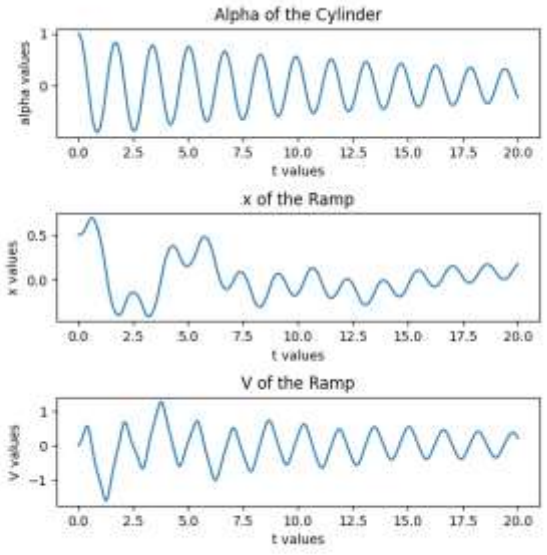
```
import matplotlib.pyplot as plt
import matplotlib.lines as mlines
import numpy as np
import math
from matplotlib.animation import FuncAnimation
import sympy as sp
from scipy.integrate import odeint

def odesys(y, t, M, m, c, k, l, F0, p):
    dy = np.zeros(4)
    dy[0] = y[2]
    dy[1] = y[3]

    a11 = M + m
    a12 = m * l * np.cos(y[1])
    a21 = np.cos(y[1])
    a22 = 3 * l / 2

    b1 = F0 * np.sin(p*t) - c * y[0] - k * y[2] + m * l * y[3] ** 2 *
np.sin(y[1])
    b2 = -np.sin(y[1]) * m * 9.81 * l

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

    return dy

if __name__ == '__main__':


    # Начальные параметры

    M = 5
    m = 2
    R = 1
    r = 0.1
    F0 = 1
    p = 0.4
    c = 10
    k = 3
    l = R-r
    x0 = 0.5
    alpha0 = 1
    dx0 = 0
    dalpha0 = 0

    d = R - r
    O1_x = 1.1
    O1_y = 1.3

    T = np.linspace(0, 20, 4001)

    y0 = [x0, alpha0, dx0, dalpha0]

    # Решение
    Y = odeint(odesys, y0, T, (M, m, c, k, l, F0, p))

    alpha = Y[:, 1]
    Rec_x = Y[:, 0]
    V_x = Y[:, 2]

    # Массивы
```

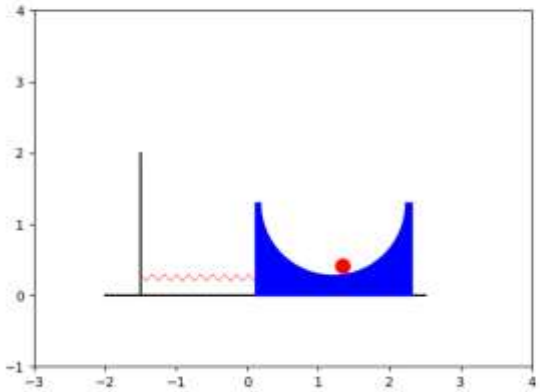
```
    OY = np.zeros_like(T)
    OX = np.zeros_like(T)
    VXREC = np.zeros_like(T)
    VXO = np.zeros_like(T)
    VYO = np.zeros_like(T)

    for i in np.arange(len(T)):
        # ALPHA[i] = sp.Subs(alpha, t, T[i])
        # REC_X[i] = sp.Subs(Rec_x, t, T[i])
        OY[i] = O1_y - d * sp.cos(alpha[i])
        OX[i] = O1_x + d * sp.sin(alpha[i]) + Rec_x[i]
        # VXREC[i] = sp.Subs(VxRec, t, T[i])
        # VXO[i] = sp.Subs(VxO, t, T[i])
        # VYO[i] = sp.Subs(VyO, t, T[i])

    # Границы рисунка
    #fig, ax = plt.subplots()
    fig = plt.figure(figsize=(4, 10))
    ax = fig.add_subplot(1, 2, 1)
    plt.xlim(-3, 4)
    plt.ylim(-1, 4)
    ax.set_aspect(1)

    # Прямоугольник
    body1 = plt.Rectangle((x0, 0.0), width=2.2, height=1.3, color='b')

    # Линия нижней поверхности
    bottom_line_x = [-2, 2.5]
    bottom_line_y = [0, 0]
    plt.plot(bottom_line_x, bottom_line_y, 'k')

    # Линия боковой поверхности
    side_line_x = [-1.5, -1.5]
    side_line_y = [0, 2]
    plt.plot(side_line_x, side_line_y, 'k')

    # Выколотая окружность
    white_circle = plt.Circle((1.1, 1.3), radius=1, color='w')

    # Функция создает набор координат x,y для построения зигзагообразной
линии, которая изображает пружину
    def get_spring_line(length, coils, diameter):
        x = np.linspace(0, length, coils * 2)
        y = [diameter * 0.5 * (-1) ** i for i in range(len(x))]
        return np.array([x, y])

    # Пружина
    spring_xy = get_spring_line(1.5, 10, 0.1)
    spring = mlines.Line2D(spring_xy[0] - 1.5, spring_xy[1] + 0.25, lw=0.5,
color='r')

    # Цилиндр
    cylinder = plt.Circle((OX[0], OY[0]), radius=0.1, color='r')

    # Графики
    ax2 = fig.add_subplot(4, 2, 2)
    ax2.plot(T, alpha)
    plt.title('Alpha of the Cylinder')
    plt.xlabel('t values')
    plt.ylabel('alpha values')

    ax3 = fig.add_subplot(4, 2, 4)
    ax3.plot(T, Rec_x)
    plt.title('x of the Ramp')
```

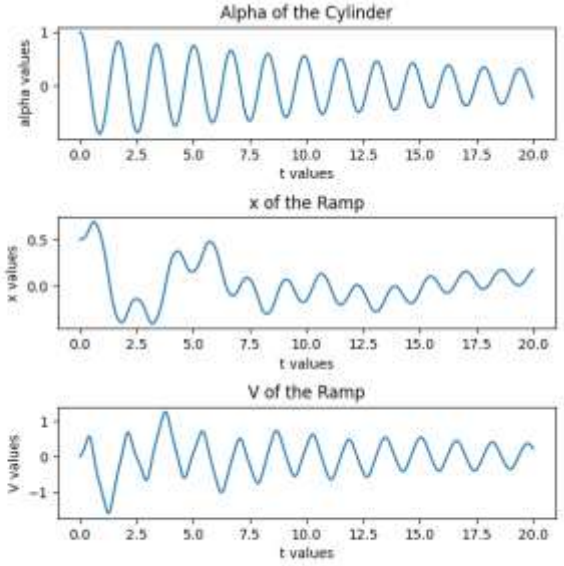
```
    plt.xlabel('t values')
    plt.ylabel('x values')

    ax4 = fig.add_subplot(4, 2, 6)
    ax4.plot(T, V_x)
    plt.title('V of the Ramp')
    plt.xlabel('t values')
    plt.ylabel('V values')

    plt.subplots_adjust(wspace=0.3, hspace=0.7)

    def init():
        # Прямоугольник
        ax.add_patch(body1)
        # Выколотая окружность
        ax.add_patch(white_circle)
        # Пружина
        ax.add_line(spring)
        # Цилиндр
        ax.add_patch(cylinder)
        return body1, white_circle, spring, cylinder

    def anima(j):  # Анимация движения
        cylinder.center = OX[j], OY[j]
        white_circle.center = Rec_x[j] + 1.1, 1.3
        body1.xy = Rec_x[j], 0
        sp_xy = get_spring_line(1.5 + Rec_x[j], 10, 0.1)
        spring.set_data(sp_xy[0] - 1.5, sp_xy[1] + 0.25)
        return body1, white_circle, spring, cylinder


    # Анимация
    anim = FuncAnimation(fig, anima, init_func=init, frames=len(T),
interval=5, blit=True)
    plt.show()
```

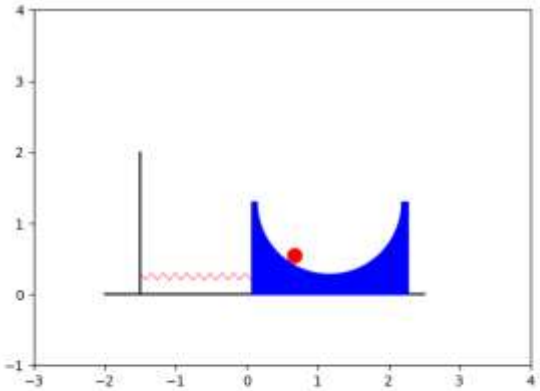
Работа программы:

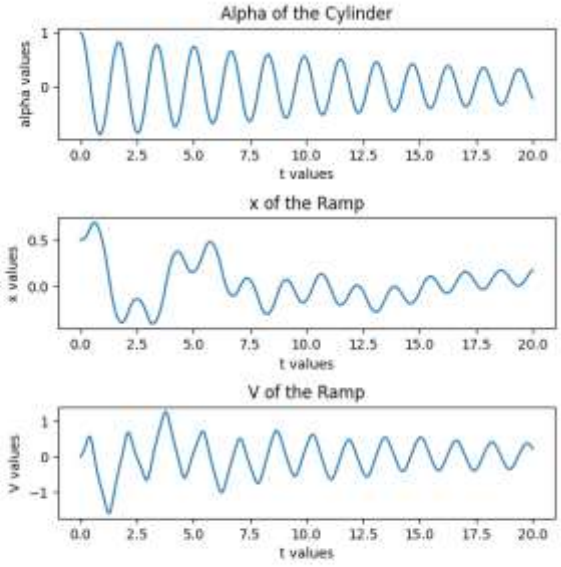







Alpha of the Cylinder
alpha values
t values
x of the Ramp
x values
t values
V of the Ramp
V values
t values



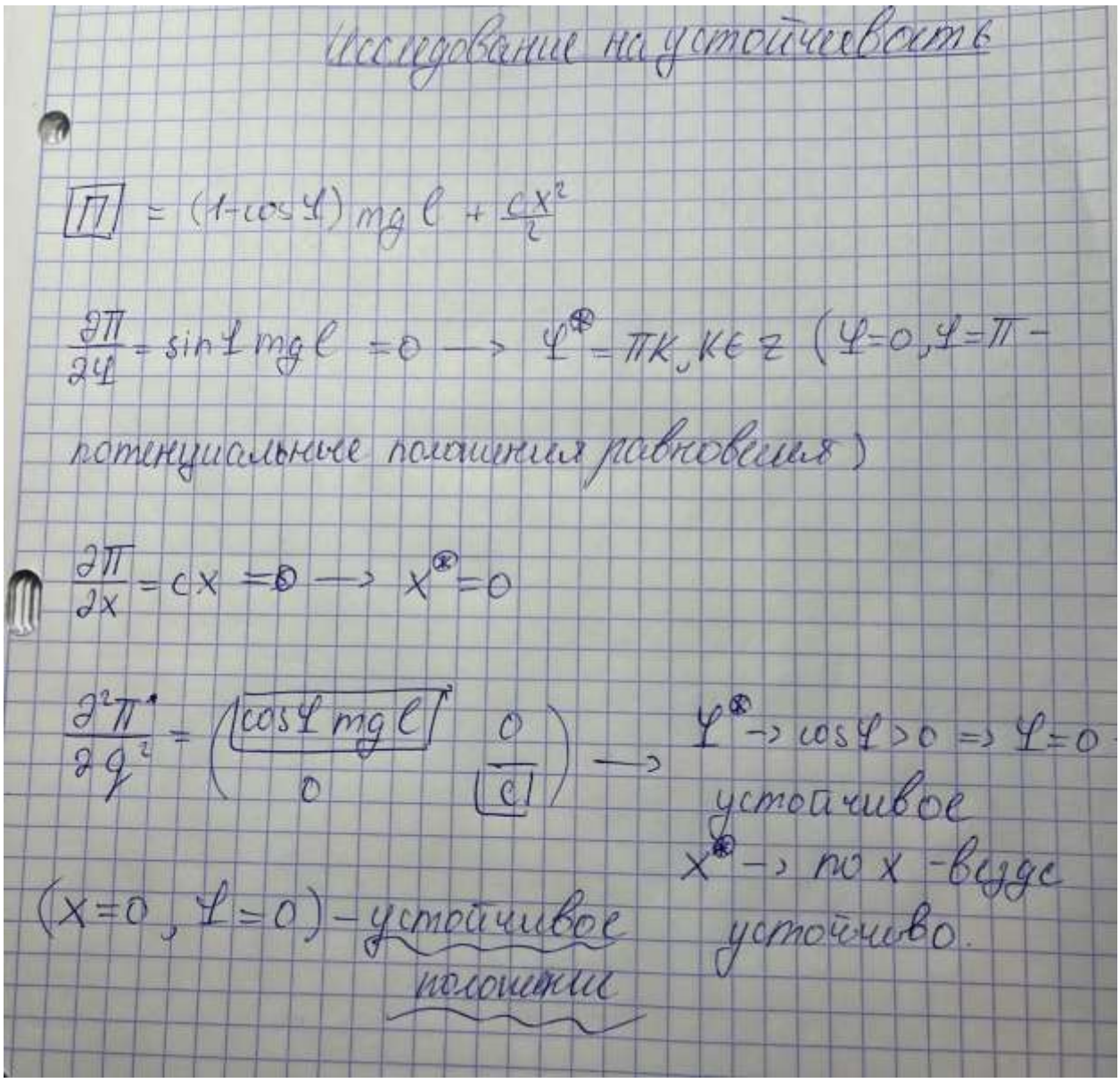

Alpha of the Cylinder
alpha values
t values
x of the Ramp
x values
t values
V of the Ramp
V values
t values

Лабораторная работа 4.

Симуляция движения системы с двумя степенями свободы

Задание: Построить анимацию движения системы, а также графики законов движения системы и указанных в задании реакций для разных случаев системы (поэкспериментировать с параметрами системы). Исследовать на устойчивость. Показать правильность работы своей механической системы.

Исследование на устойчивость.

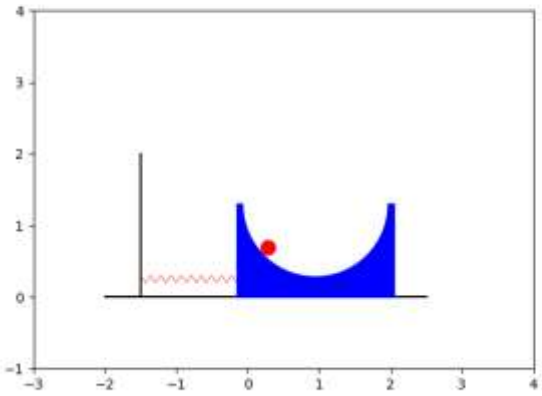


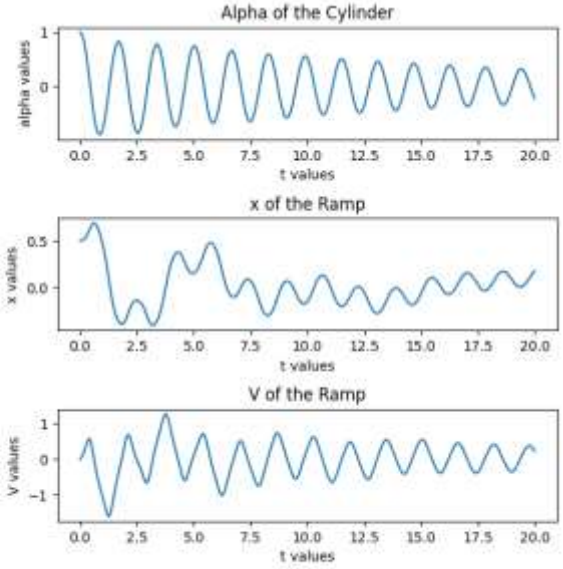
Результат работы программы.

Выведем полученные графики работы программы:

1) M = 5; m = 2; R = 1; r = 0.1; F0 = 1; p = 0.4; c = 10; k = 3;

y0 = [0.5, 1, 0, 0] – значения, которые даны в 12 задании лабораторной в качестве начальных. (3 Лабораторная работа):

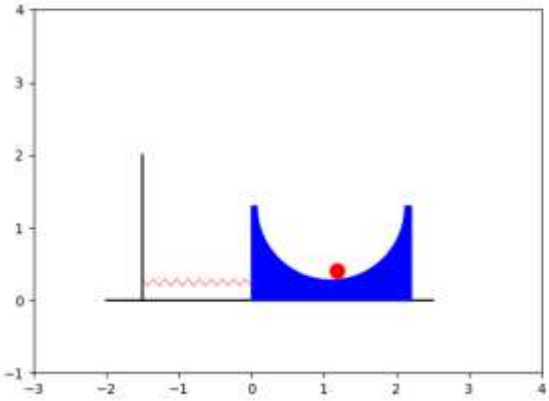


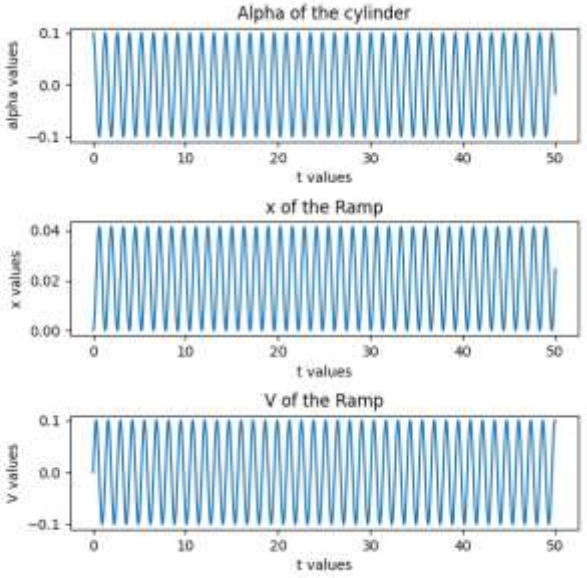


Результат: из-за присутствия внешних сил, в начале движения ‘рампа’ относительно сильно отходит от положения равновесия, но со временем, она приходит в него и колеблется в его пределах. Отклонение цилиндра от положения равновесия так же со временем уменьшается.

2) M = 10; m = 3; R = 1; r = 0.1; F0 = 0; p = 0; c = 0; k = 0;

y0 = [0, 0.1, 0, 0] – пружины фактически нет, потому что коэффициент упругости (c) равен нулю, сопротивления пружины так же нет, внешние силы отсутствуют. Изначальное положение ‘рампы’ в положении равновесия. Цилиндр немного отклонен.

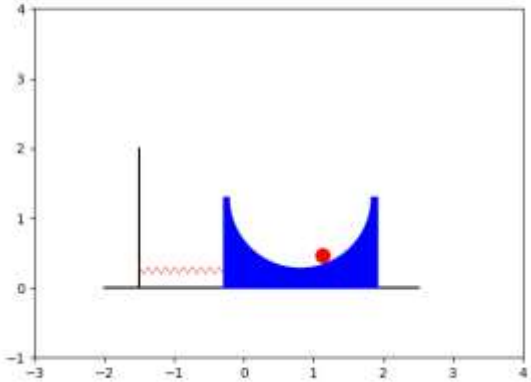


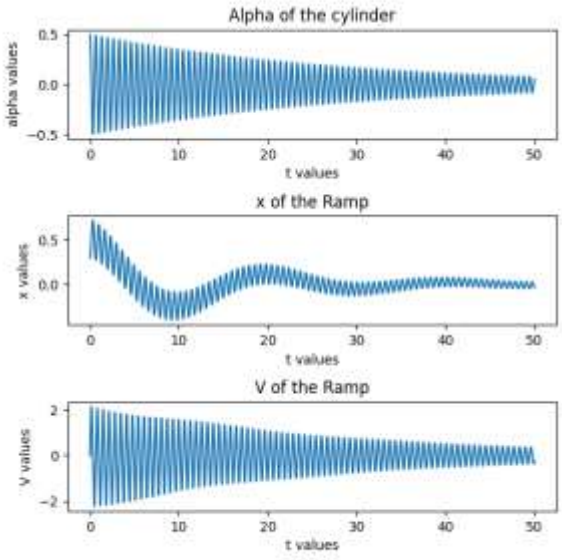


Результат: Отношение массы цилиндра к массе ‘рампы’ уменьшилось, соответственно уменьшилось и его влияние на отклонение ‘рампы’. Изначальное отклонение цилиндра не большое, что позволят смоделировать малые колебания системы относительно её положения равновесия. По графикам можно судить о том, насколько незначительно отклоняется ‘рампа’ от своего положения равновесия. Влияние со стороны пружины на систему нет, следовательно на рампу действует только цилиндр.

3) M = 10; m = 10; R = 1; r = 0.1; F0 = 0; p = 0; c = 2; k = 3;

y0 = [0.3, 0.5, 0, 0] – значительные изначальные отклонения от положения равновесия цилиндра и ‘рампы’. Слабая пружина. Масса цилиндра равна массе ‘рампы’.



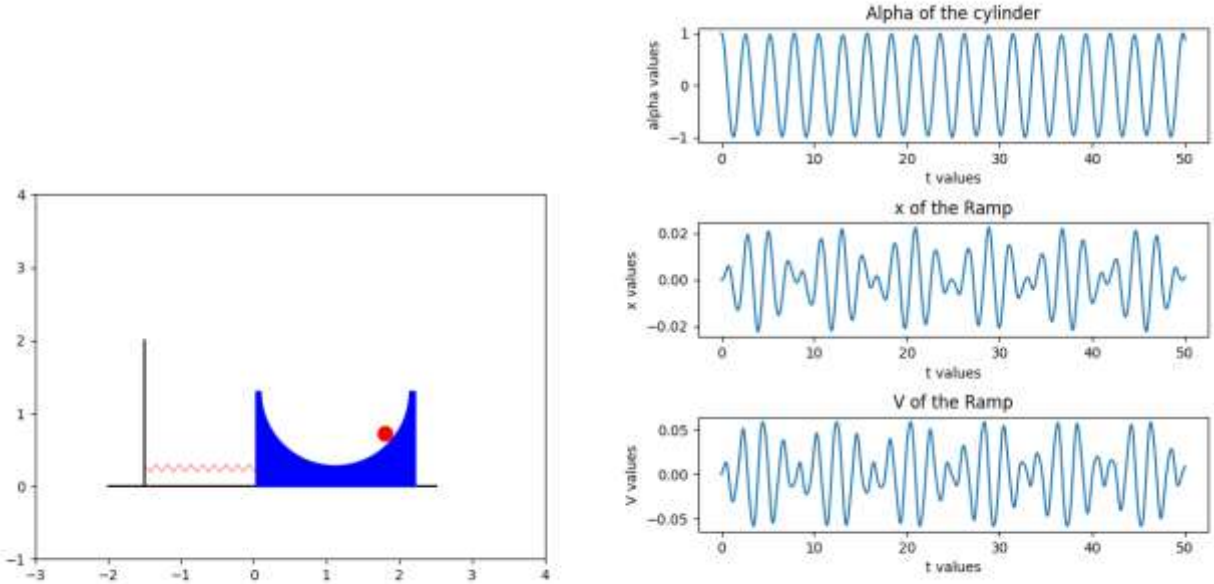


Результат: Коэффициент упругости пружины небольшой, следовательно её влияние на систему мало. Масса цилиндра равна массе ‘рампы’ и его

начальное отклонение от положения равновесия значительно, из-за чего при его движении ‘рампа’ сильно колеблется. Начальные отклонения от положения равновесия большие, по вышеописанным причинам, но со временем они так же уменьшаются и приближаются к положению равновесия. По графику положения ‘рампы’ можно увидеть, что в данном эксперименте из-за значительного начального отклонения, она колебалась сначала по одно сторону от положения равновесия, а затем по другую, после чего колебалась уже относительно положения равновесия.

4) M = 100; m = 1; R = 1; r = 0.1; F0 = 0; p = 0; c = 1000; k = 0;

y0 = [0, 1, 0, 0] – ‘Рампа’ изначально находится в положении равновесия. Жесткость пружины очень большая. Масса цилиндра незначительно относительно массы ‘рампы’. Начальное отклонение цилиндра сильное.



Результат: В связи с тем, что ‘рампа’ изначально находится в положении равновесия, жесткость пружины огромна, а масса цилиндра незначительна, ‘рампа’ практически не двигается с места, что можно увидеть на графике положения рампы. Сам цилиндр совершает сильные колебания относительно положения равновесия.

Вывод

По ходу работы над этой лабораторной я научился моделировать сложное механическое движение, задействовав в реализации симуляции уравнение Лагранжа второго рода. Получил навыки работы с таким библиотека Python, как matplotlib – для отрисовки картинки, numpy – для более удобной работы с массивами, scipy – для решения дифференциального уравнения, sympy – для работы с символьными вычислениями и дифференцирования. Также я улучшил свои навыки по теоретической механике и её понимание.