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 №9**

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

Зубко Дмитрий Валерьевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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 при помощи средств Python. Построить анимацию движения системы, а также графики законов движения системы. А также составить уравнение Лагранжа второго рода для заданной системы.

Расчет уравнения Лагранжа производится по следующему алгоритму:

1) Определить количество степеней свободы системы 𝑛.

2) В соответствии с количеством степеней свободы системы, полученным в п. 1, ввести соответствующее число обобщенных координат (𝑞1,𝑞2,…,𝑞𝑛).

3) Посчитать кинетическую энергию в зависимости от обобщенных координат и обобщенных скоростей 𝑇(𝑞1,…,𝑞𝑛,𝑞̇1,…,𝑞̇𝑛).

4) Найти обобщенные силы, которые соответствуют обобщенным координатам (𝑄1,…,𝑄𝑛).

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

()− =𝑄𝑖.

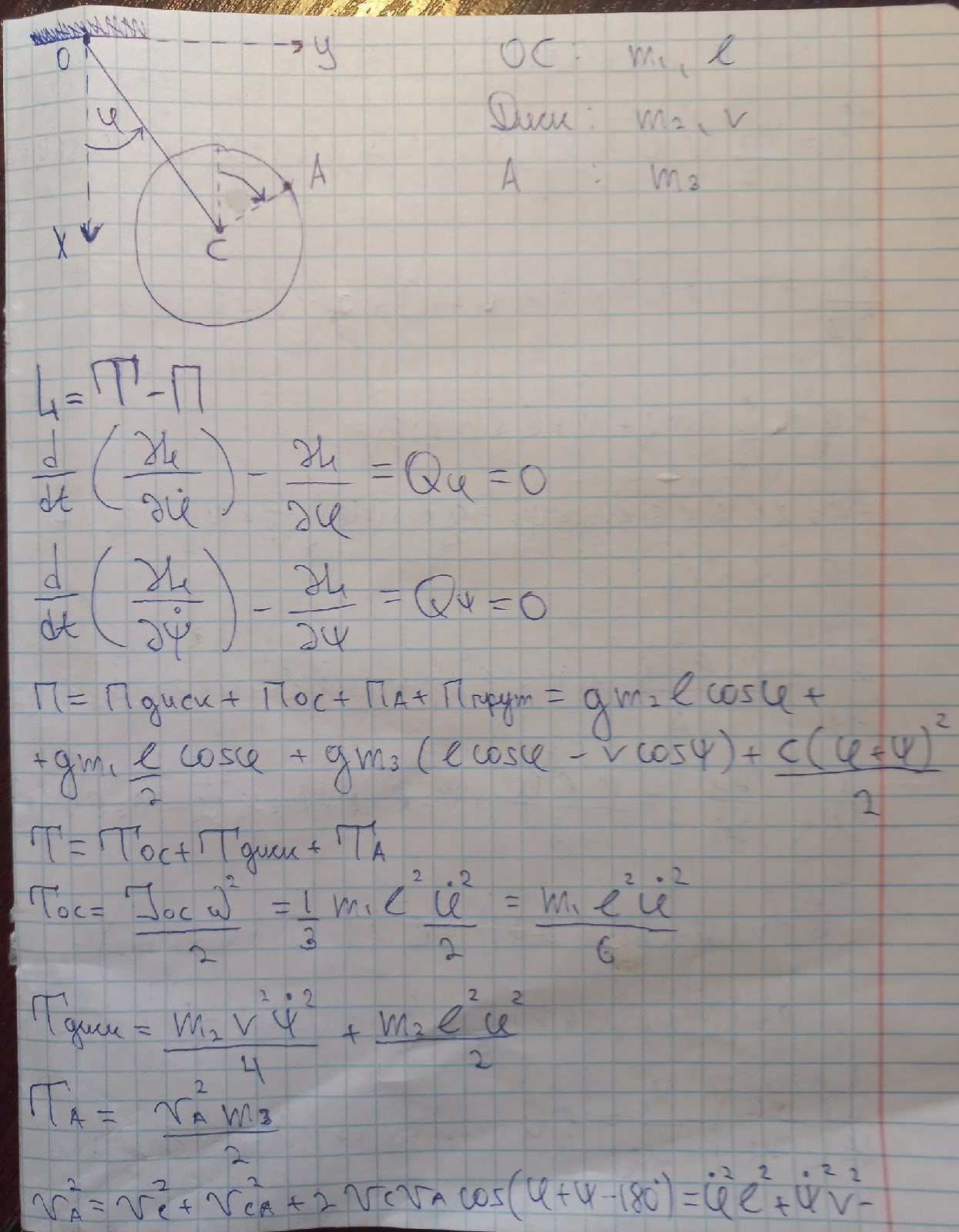
Если система консервативна, то можно упростить вид уравнения, используя функцию Лагранжа

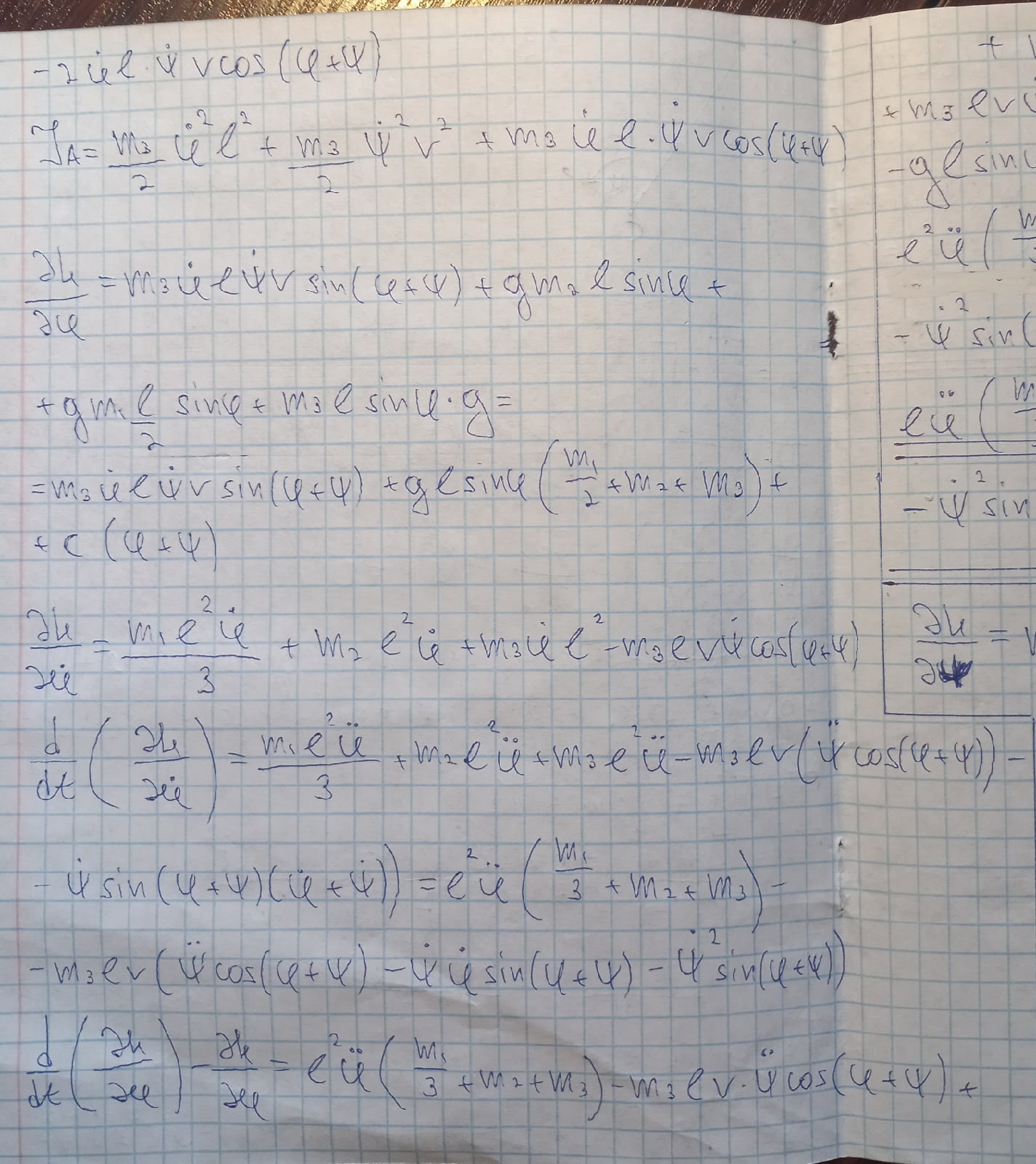
𝐿=𝑇−𝛱.

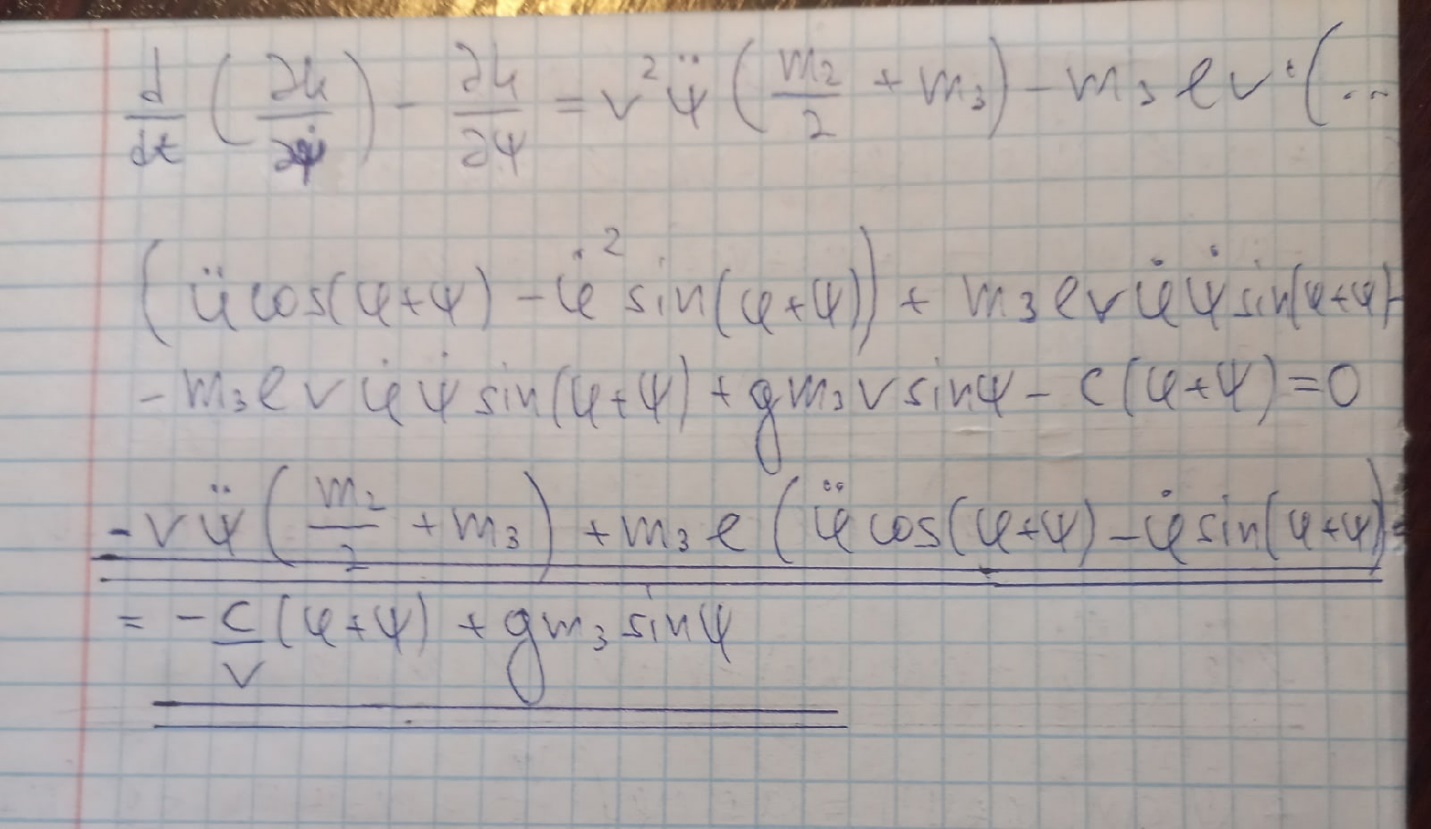
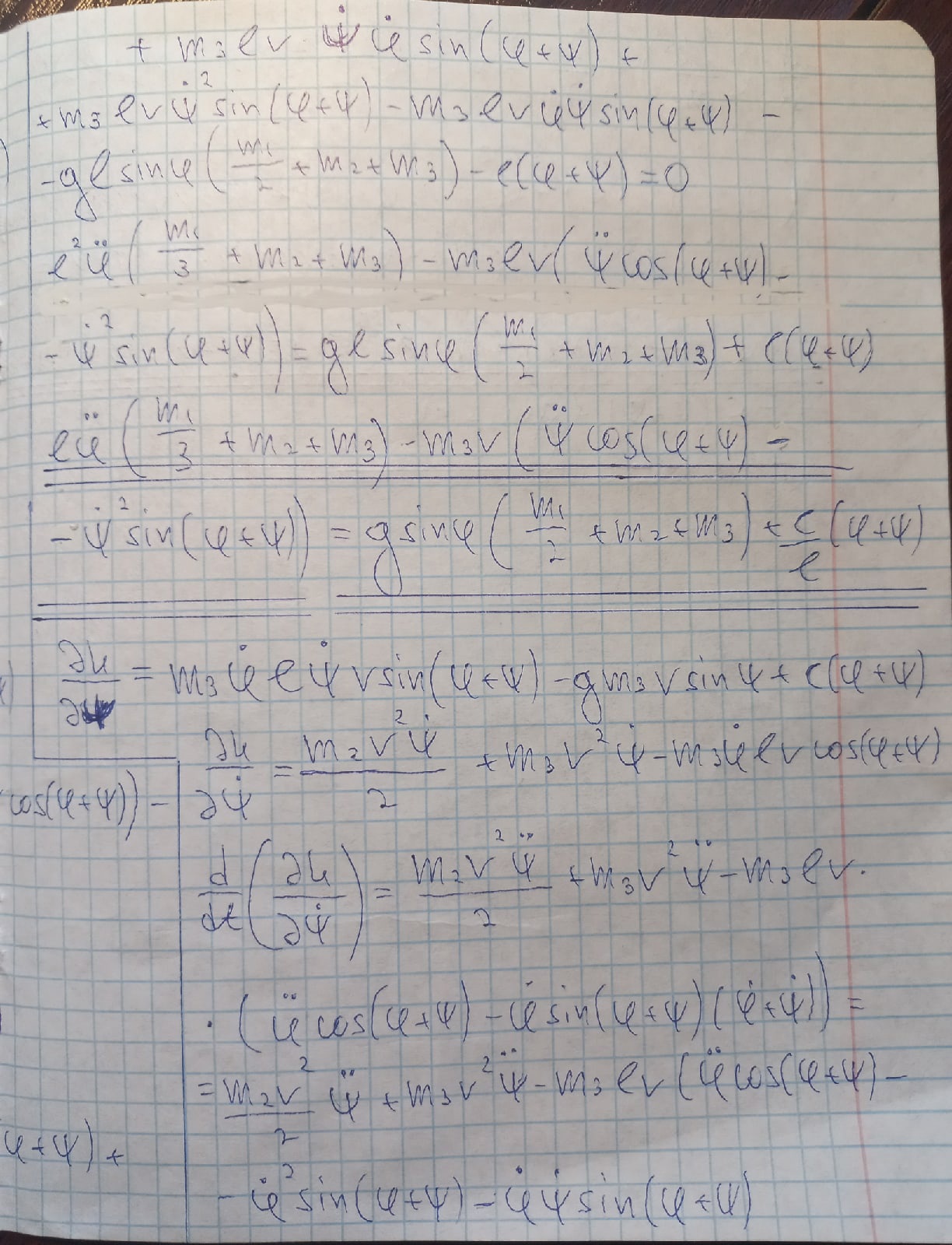
Тогда уравнение Лагранжа второго рода для консервативной системы будет выглядеть следующим образом:

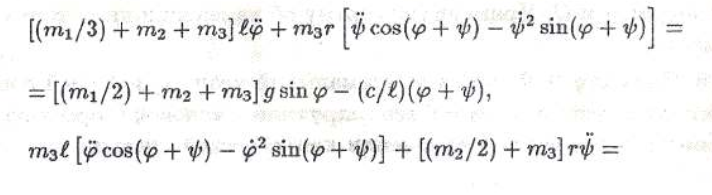
()− =0.

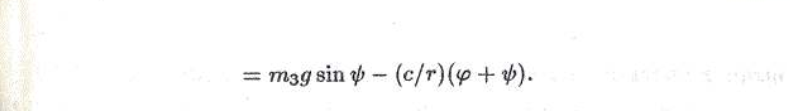
Ниже привожу вывод всех нужных уравнений, проделанный собственноручно:











Исходный код программы:

import numpy as np

import matplotlib.pyplot as plt

import math

from scipy.integrate import odeint

from matplotlib.animation import FuncAnimation

// задаём начальные значения и временной промежуток

Steps = 1000

t = np.linspace(0, 25, Steps)

m1 = 1

m2 = 0.001

m3 = 0.001

l = 4

r = 1

c = 10

g = 9.8

def odesys(y, t, m1, m2, m3, l, r, c):

dy = np.zeros(4)

dy[0] = y[2]

dy[1] = y[3]

a11 = l \* (m1 / 3 + m2 + m3)

a12 = m3 \* r \* np.cos(y[0] + y[1])

a21 = m3 \* r \* np.cos(y[0] + y[1])

a22 = (m2 / 2 + m3) \* r

b1 = (m1 / 2 + m2 + m3) \* g \* np.sin(y[1]) - c / l \* (y[1] + y[0]) + m3 \* r \* (y[2] \*\* 2) \* np.sin(y[0] + y[1])

b2 = m3 \* g \* np.sin(y[0]) - c / r \* (y[0] + y[1]) + m3 \* l \* (y[3] \*\* 2) \* np.sin(y[0] + y[1])

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

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

return dy

# 0.35 0.35 0.025 0.025

// начальные значения углов

psi0 = -3.14 \* 2/ 8

phi0 = 3.14 \* 1.9 / 4

dpsi0 = 0

dphi0 = 0

y0 = [psi0, phi0, dpsi0, dphi0]

Y = odeint(odesys, y0, t, (m1, m2, m3, l, r, c))

psi = Y[:, 0]

phi = Y[:, 1]

fig = plt.figure(figsize=[10, 5])

ax = fig.add\_subplot(1, 2, 1)

ax.axis('equal')

ax.set(xlim=[-7, 7], ylim=[-7, 3])

X\_Ground = [-1, 1]

Y\_Ground = [0, 0]

ax.plot(X\_Ground, Y\_Ground, color='black', linewidth=4)

L = 4

WheelR = 1

T = 2 \* 3.14 \* np.sqrt(L / g)

omega = 2 \* 3.14 / T

A = np.sin(phi[0]) \* l

X\_A = A \* np.sin(phi)

Y\_A = -np.sqrt(L \*\* 2 - X\_A \*\* 2)

X\_V\_A = np.diff(X\_A)

Y\_V\_A = np.diff(Y\_A)

X\_W\_A = np.diff(X\_V\_A)

Y\_W\_A = np.diff(Y\_V\_A)

//построение графиков

ax2 = fig.add\_subplot(4, 2, 2)

ax2.plot(X\_V\_A)

plt.title('Vx of dot')

plt.xlabel('t values')

plt.ylabel('Vx values')

ax2 = fig.add\_subplot(4, 2, 4)

ax2.plot(Y\_V\_A)

plt.title('Vy of dot')

plt.xlabel('t values')

plt.ylabel('Vy values')

ax2 = fig.add\_subplot(4, 2, 6)

ax2.plot(X\_W\_A)

plt.title('Wx of dot')

plt.xlabel('t values')

plt.ylabel('Wx values')

ax2 = fig.add\_subplot(4, 2, 8)

ax2.plot(Y\_W\_A)

plt.title('Wy of dot')

plt.xlabel('t values')

plt.ylabel('Wy values')

plt.subplots\_adjust(wspace=0.3, hspace=0.7)

// рисуем окружность

tetta = np.linspace(0, 6.28, 25)

X\_Wheel = WheelR \* np.sin(tetta)

Y\_Wheel = WheelR \* np.cos(tetta)

Drawed\_Wheel = ax.plot(X\_A[0] + X\_Wheel, Y\_A[0] + Y\_Wheel)[0]

// рисуем точки

Point\_O = ax.plot(0, 0, marker='o')[0]

Point\_A = ax.plot(X\_A[0], Y\_A[0], marker='o')[0]

Line\_AO = ax.plot([X\_A[0], 0], [Y\_A[0], 0])[0]

A = np.sin(psi[0]) \* WheelR

X\_B = A \* np.sin(phi)

Y\_B = np.sqrt(WheelR \*\* 2 - X\_B \*\* 2)

Point\_B = ax.plot(X\_B[0] + X\_A[0], Y\_B[0] + Y\_A[0], marker='o')[0]

Line\_AB = ax.plot([X\_A[0], X\_B[0] + X\_A[0]], [Y\_A[0], Y\_B[0] + Y\_A[0]])[0]

// рисуем пружину

Nv = 3

R1 = 0.1

R2 = 0.5

gretta = np.linspace(0, Nv \* 6.28 - np.arctan((X\_A[0] / Y\_A[0])), 100)

X\_SpiralSpr = -(R1 + gretta \* (R2 - R1) / gretta[-1]) \* np.sin(gretta)

Y\_SpiralSpr = (R1 + gretta \* (R2 - R1) / gretta[-1]) \* np.cos(gretta)

Drawed\_SpiralSpring = ax.plot(X\_SpiralSpr + X\_A[0], Y\_SpiralSpr + Y\_A[0])[0]

// функция анимации

def anima(i):

Line\_AO.set\_data([X\_A[i], 0], [Y\_A[i], 0])

Point\_A.set\_data(X\_A[i], Y\_A[i])

Drawed\_Wheel.set\_data(X\_A[i] + X\_Wheel, Y\_A[i] + Y\_Wheel)

Point\_B.set\_data(X\_B[i] + X\_A[i], Y\_B[i] + Y\_A[i])

Line\_AB.set\_data([X\_A[i], X\_B[i] + X\_A[i]], [Y\_A[i], Y\_B[i] + Y\_A[i]])

gretta = np.linspace(0, Nv \* 6.28 - np.arctan((X\_A[i] / Y\_A[i])), 100)

X\_SpiralSpr = -(R1 + gretta \* (R2 - R1) / gretta[-1]) \* np.sin(gretta)

Y\_SpiralSpr = (R1 + gretta \* (R2 - R1) / gretta[-1]) \* np.cos(gretta)

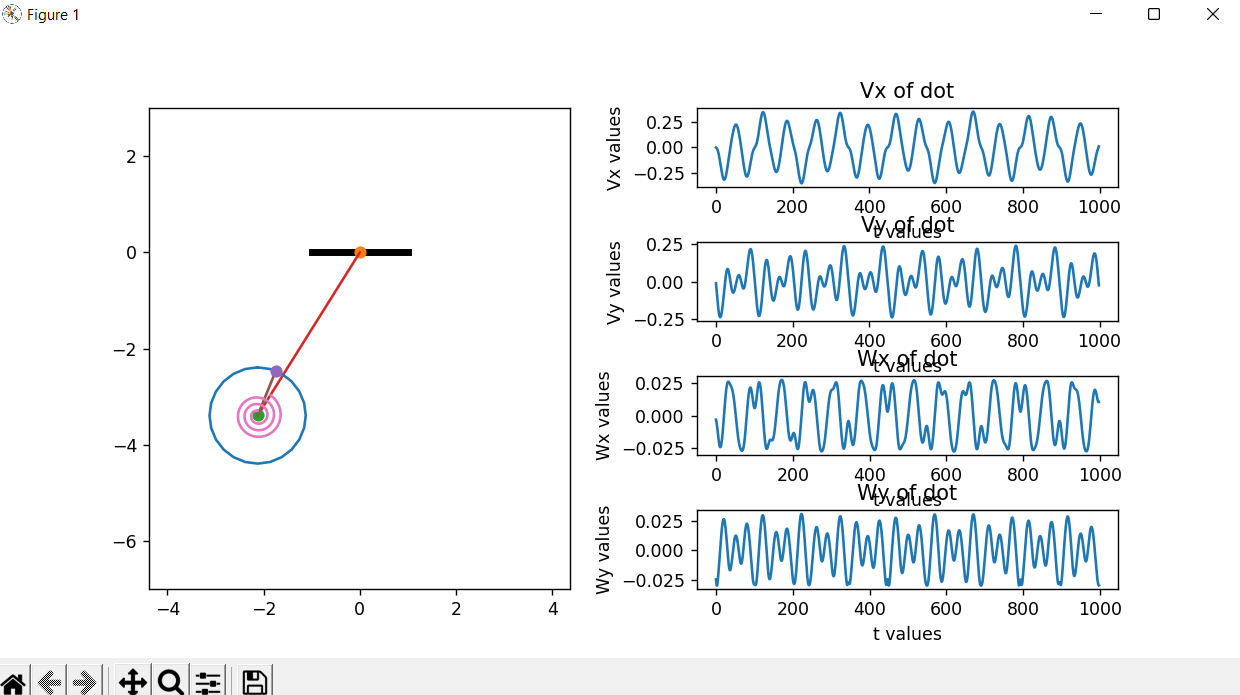
Drawed\_SpiralSpring.set\_data(X\_SpiralSpr + X\_A[i], Y\_SpiralSpr + Y\_A[i])

return [Line\_AO, Point\_A, Drawed\_Wheel, Line\_AB, Point\_B, Drawed\_SpiralSpring]

anim = FuncAnimation(fig, anima, frames=Steps, interval=100, blit=True)

plt.show()

Результат выполнения программы:



**Лабораторная работа 4**

**Симуляция движения системы с двумя степенями свободы**

Задание: Построить анимацию движения системы, а также графики законов движения системы. Исследовать на устойчивость. Показать правильность работы своей механической системы.

1). Выведем полученные графики работы программы:

y0 = [psi0, phi0, dpsi0, dphi0]

psi0 = -3.14 / 8

phi0 = 3.14 / 4

dpsi0 = 0

dphi0 = 0

m1 = 0.1

m2 = 1

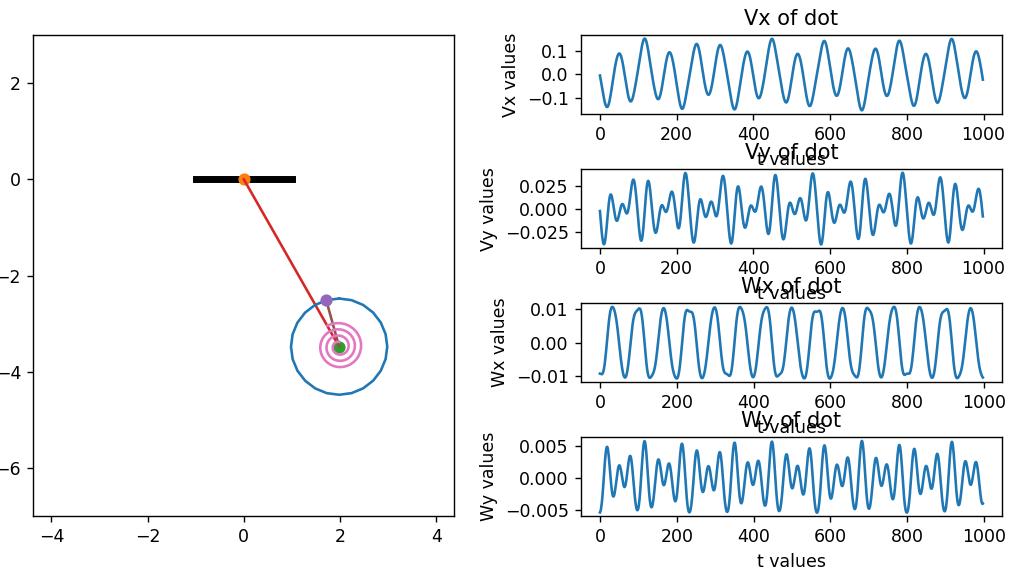
m3 = 0.1

l = 4

r = 1

c = 10

g = 9.8



При данных значениях система совершает нормальные колебания, маятник качается адекватно, нет скачков скорости.

2). Выведем полученные графики работы программы:

y0 = [psi0, phi0, dpsi0, dphi0]

psi0 = -3.14 / 8

phi0 = 3.14 / 4

dpsi0 = 0

dphi0 = 0

Увеличим в данном случае массы в некоторое количество раз

m1 = 10

m2 = 100

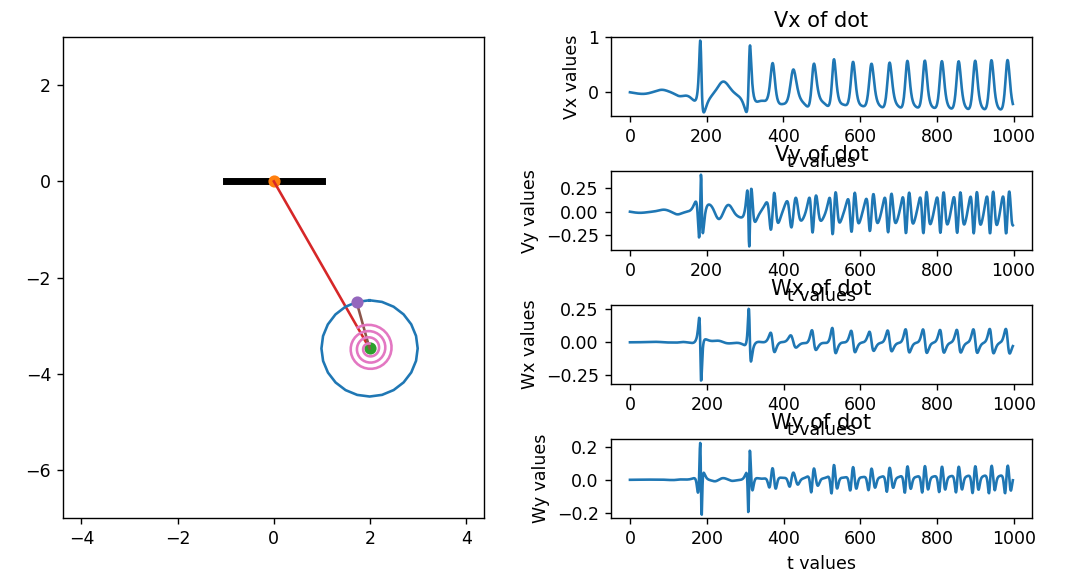
m3 = 10

l = 4

r = 1

c = 10

g = 9.8



При больших значениях массы система совершает уже не гармонические колебания, происходят скачки в скорости.

3). Выведем полученные графики работы программы:

y0 = [psi0, phi0, dpsi0, dphi0]

увеличим угол фи0 в 1.9 раз

psi0 = -3.14 / 8

phi0 = 3.14 \* 1.9 / 4

dpsi0 = 0

dphi0 = 0

ввернём значения масс как в 1 случае

m1 = 0.1

m2 = 1

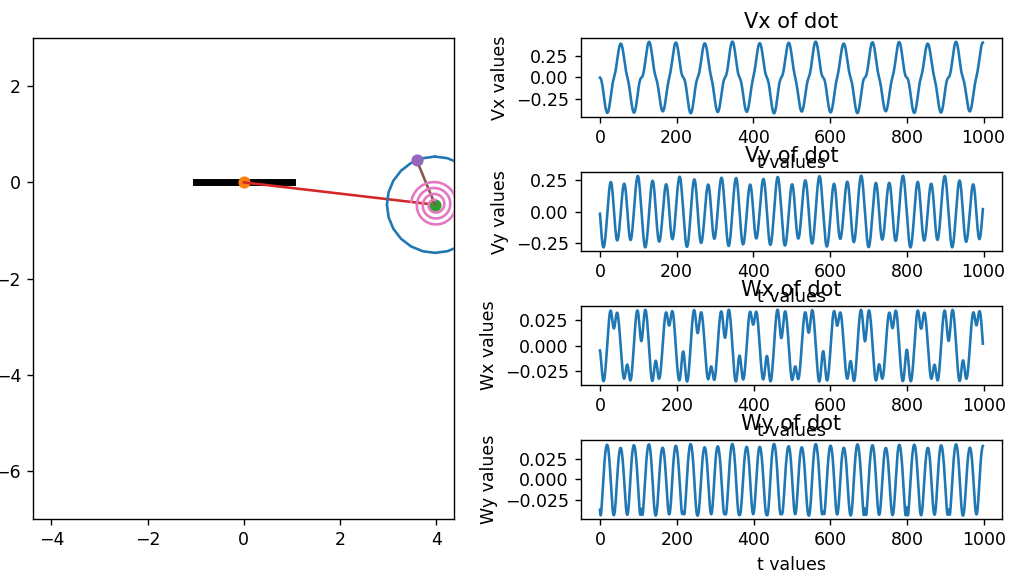
m3 = 0.1

l = 4

r = 1

c = 10

g = 9.8



При данных значениях система будет совершать колебания с большей скоростью, чем в первом случае. Это происходит потому что больше первоначальный угол отклонения фи0.

4). Выведем полученные графики работы программы:

y0 = [psi0, phi0, dpsi0, dphi0]

psi0 = 0

phi0 = 0

dpsi0 = 0

dphi0 = 0

m1 = 0.1

m2 = 1

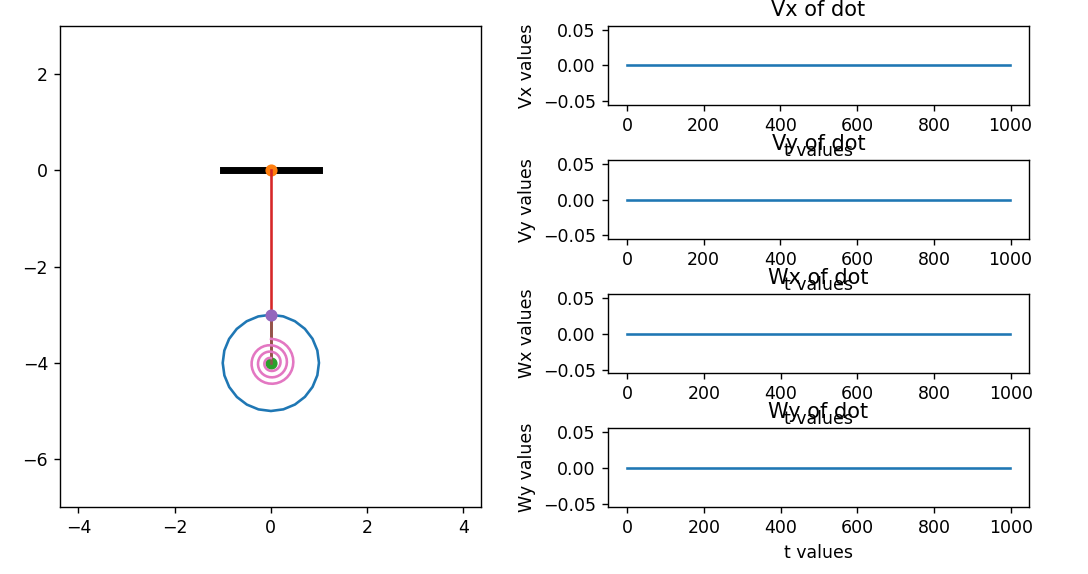
m3 = 0.1

l = 4

r = 1

c = 10

g = 9.8



При данных значениях система будет оставаться на месте, так как нет энергии, чтобы совершать колебания.

**Вывод:**

В данной лабораторной работе я составил анимацию движения системы и вывел уравнение Лагранжа второго рода для заданной системы. Рассмотрел, как ведёт себя система при разных начальных значениях.

Лабораторные работы помогли мне изучить данные темы и освоить специальные библиотеки языка Python, с помощью которых можно создавать разные анимации, а также расширили мои знания по теоретической механике.