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 № 16**

Выполнил студент группы M8O-206Б-22

Волков А.Д.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Зав. каф. 802, Бардин Б.С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

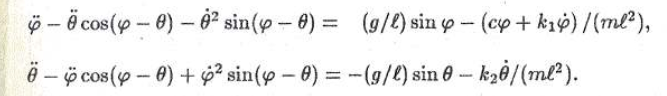
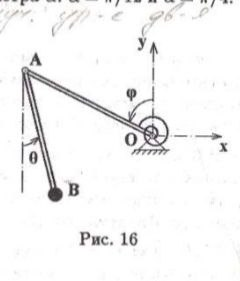
с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2023

*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.

*Вариант 16*:



*Программа:*

import numpy as np

import sympy as s

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from scipy.integrate import odeint

import math

def odesys(y, t, g, l, c, k1, k2, m):

dy = np.zeros\_like(y)

dy[0] = y[2]

dy[1] = y[3]

a11 = 1

a12 = -np.cos(y[0] - y[1])

a21 = -np.cos(y[0] - y[1])

a22 = 1

b1 = (g / l) \* np.sin(y[0]) - (c \* y[0] + k1 \* y[2]) / (m \* l \*\* 2) + y[3] \*\* 2 \* np.sin(y[0] - y[1])

b2 = -(g / l) \* np.sin(y[1]) - k2 \* y[3] / (m \* l \*\* 2) - y[2] \*\* 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

m = 0.3

l = 1

c = 2

k1 = 1

k2 = 1

g = 9.81

l\_OA = l

l\_AB = l

phi0 = np.pi / 3

thetta0 = -np.pi / 3

dphi0 = 0

dtet0 = 0

y0 = [phi0, thetta0, dphi0, dtet0]

Steps = 1500

t = np.linspace(0, 20, Steps)

Y = odeint(odesys, y0, t, (g, l, c, k1, k2, m))

phi = Y[:, 0]

tet = Y[:, 1]

dphi = Y[:, 2]

dtet = Y[:, 3]

figstat = plt.figure()

plphi = figstat.add\_subplot(4, 1, 1)

plphi.plot(t, phi)

pltet = figstat.add\_subplot(4, 1, 2)

pltet.plot(t, tet)

ddphi = np.zeros\_like(t)

ddtet = np.zeros\_like(t)

for i in range(len(t)):

ddphi[i] = odesys(Y[i], t[i], g, l, c, k1, k2, m)[2]

ddtet[i] = odesys(Y[i], t[i], g, l, c, k1, k2, m)[3]

Rx = m \* l \* (ddtet \* np.cos(tet) - dtet \*\* 2 \* np.sin(tet) - ddphi \* np.cos(phi) + dphi \*\* 2 \* np.sin(phi))

Ry = m \* l \* (ddtet \* np.sin(tet) + dtet \*\* 2 \* np.cos(tet) - ddphi \* np.sin(phi) - dphi \*\* 2 \* np.cos(phi)) + m \* g

pRx = figstat.add\_subplot(4, 1, 3)

pRx.plot(t, Rx)

pRy = figstat.add\_subplot(4, 1, 4)

pRy.plot(t, Ry)

X\_O = 0

Y\_O = 0

X\_A = l\_OA \* np.cos(phi)

Y\_A = l\_OA \* np.sin(phi)

X\_B = l\_AB \* (-np.sin(tet))

Y\_B = l\_AB \* (-np.cos(tet))

X\_Ground = [0, 0, 4]

Y\_Ground = [4, 0, 0]

Nv = 3

R1 = 0.1

R2 = 0.3

numpoints = np.linspace(0, 1, 20 \* Nv + 1)

Betas = numpoints \* (2 \* np.pi \* Nv + phi[0])

X\_Spiral = np.cos(Betas) \* (R1 + (R2 - R1) \* numpoints)

Y\_Spiral = np.sin(Betas) \* (R1 + (R2 - R1) \* numpoints)

# Создание области

fig = plt.figure()

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

ax.axis('equal')

ax.set(xlim = [-5, 5], ylim = [-5, 5])

# Отрисовка системы координат

ax.plot(X\_Ground, Y\_Ground, color = 'black', linewidth = 1)

# Отрисовка стержня OA

OA = ax.plot([X\_O, X\_A[0]], [Y\_O, Y\_A[0]])[0]

# Отрисовка стержня AB

AB = ax.plot([X\_A[0], X\_B[0]], [Y\_A[0], Y\_B[0]])[0]

# Отрисовка точек О, A, B

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

Point\_A = ax.plot(X\_A[0], Y\_A[0], marker = 'o', markersize = 6)[0]

Point\_B = ax.plot(X\_B[0], Y\_B[0], marker = 'o', markersize = 15)[0]

# Отрисовка пружины

Spiral = ax.plot(X\_Spiral, Y\_Spiral)[0]

def run(i):

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

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

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

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

Betas = numpoints \* (2 \* np.pi \* Nv + phi[i])

X\_Spiral = np.cos(Betas) \* (R1 + (R2 - R1) \* numpoints)

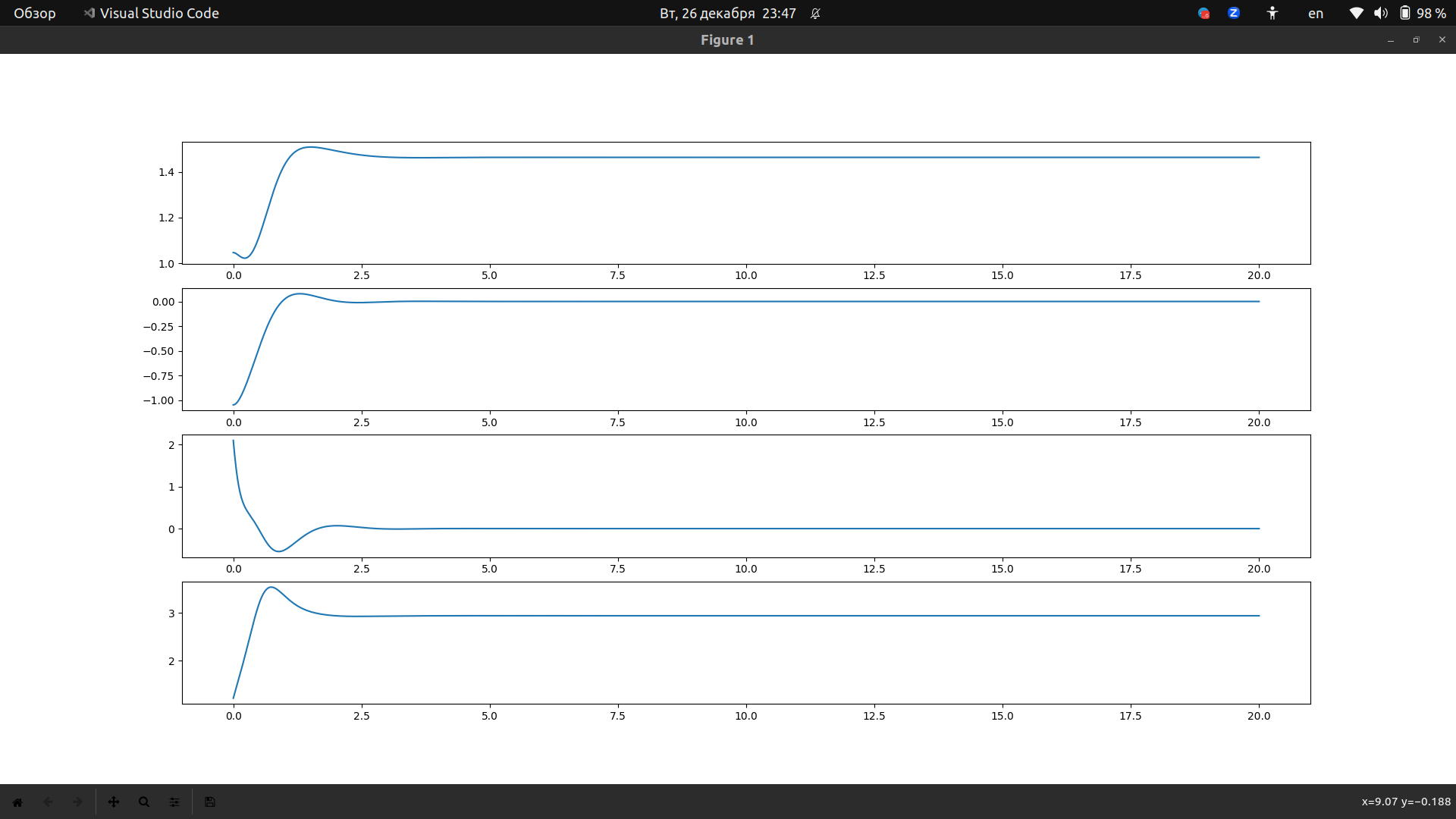
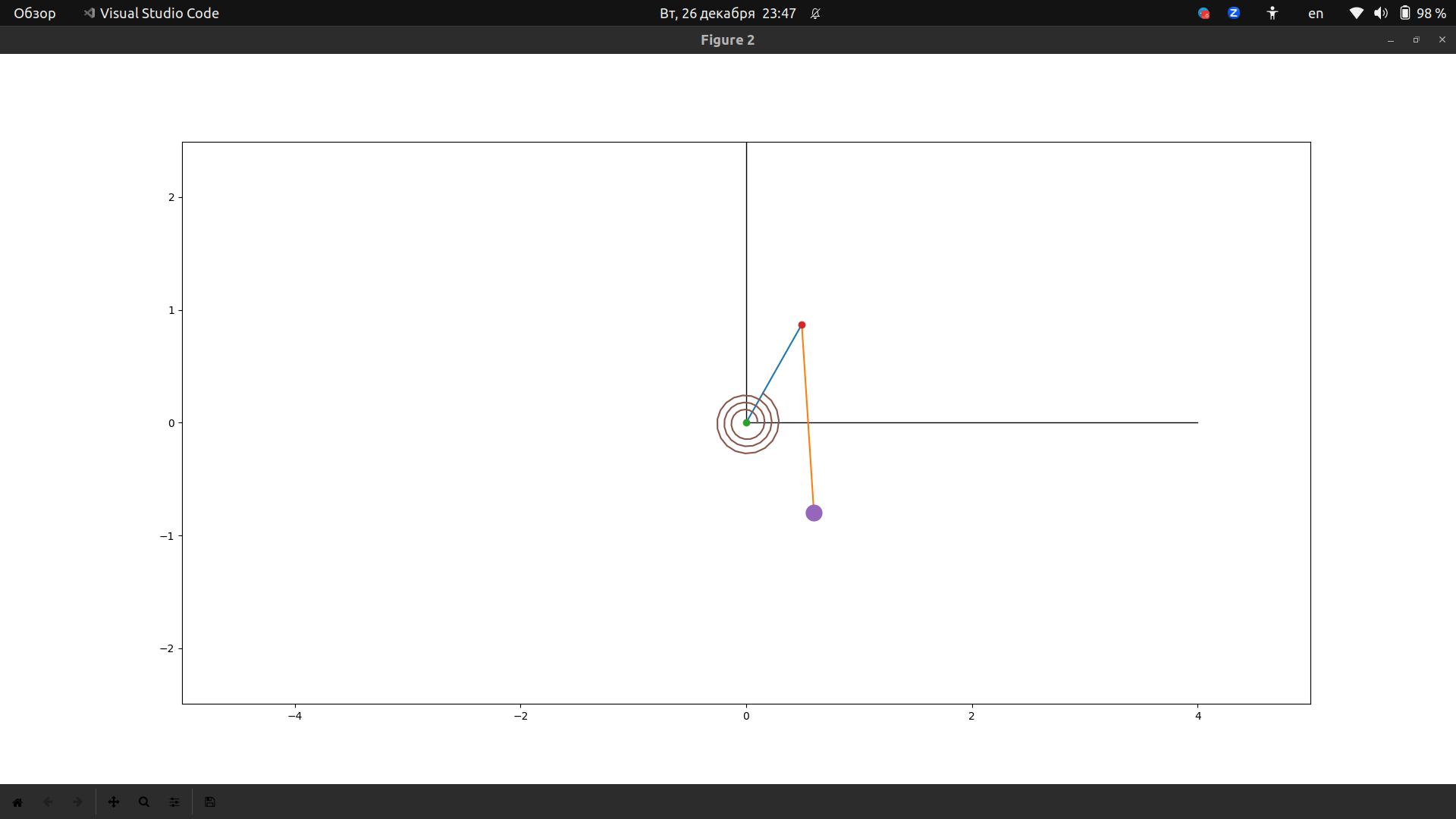
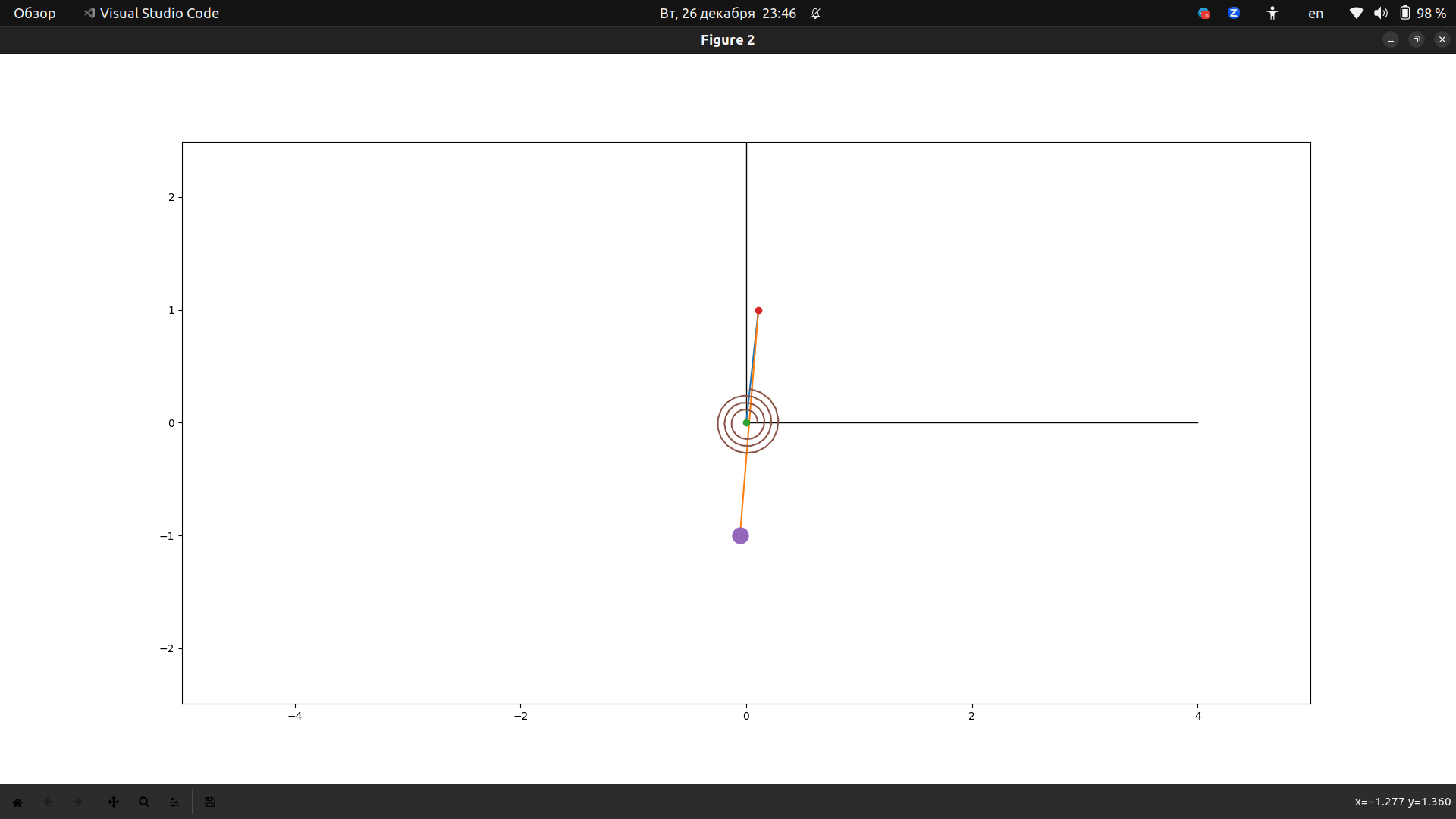
Y\_Spiral = np.sin(Betas) \* (R1 + (R2 - R1) \* numpoints)

Spiral.set\_data(X\_Spiral, Y\_Spiral)

return OA, AB, Point\_A, Point\_B, Betas, X\_Spiral, Y\_Spiral, Spiral

anim = FuncAnimation(fig, run, frames = len(t), interval = 50, repeat = False)

plt.show()



*Выводы:*

Я успешно выполнил данную лабораторную работу по теоретической механике по теме «Динамика системы». В моей программе используются реальные законы движения, благодаря чему можно посмотреть, как эта система будет вести себя в реальной жизни.