Московский авиационный институт

(национальный исследовательский университет)

Институт № 8 «Информационные технологии и прикладная математика»

**Лабораторная работа №4**

**по курсу «Теоретическая механика»**

**Малые колебания**

Выполнил студент группы М8О-207Б-20

Чекменев Вячеслав Алексеевич

Преподаватель: Чекина Евгения Алексеевна

Оценка:

Дата:

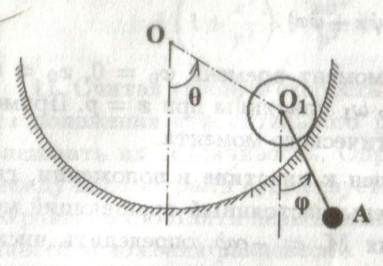
Москва, 2021

**Вариант №«25»**

**Задание:**

Реализовать анимацию движения механической системы в среде Python на основе уравнений Лагранжа 2-го рода для малых колебаний, масса грузика (точка А) = 0

**Механическая система:**

****

**Текст программы:**

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from scipy.integrate import odeint

import sympy as sp

import math

def formY2(y, t, fOm):

y1,y2 = y

dydt = [y2,fOm(y1,y2)]

return dydt

Steps = 1001

R\_Ground = 6

R\_Circle = R\_Ground/6

m1 = 0.0001

m2 = 0.00005

g = 9.81

l = R\_Ground/2 # length of the palka between O1 and A

# defining t as a symbol (it will be the independent variable)

t = sp.Symbol('t')

# defining s, phi, V=ds/dt and om=dphi/dt as functions of 't'

thetta = sp.Function('thetta')(t) # s

phi = 0;

omega\_thetta = sp.Function('omega\_thetta')(t) # V

omega\_phi = 0;

# Check the derivating process

print(sp.diff(5\*omega\_thetta\*\*2, omega\_thetta))

#1 defining the kinetic energy

V\_O1 = (R\_Ground - R\_Circle) \* omega\_thetta

om1 = V\_O1 / R\_Circle

J\_O1 = (m1 \* R\_Circle\*\*2) / 2

T1 = (m1 \* V\_O1\*\*2) / 2 + (J\_O1 \* om1\*\*2) / 2

Ve = V\_O1

Vr = omega\_phi \* l # changed sp.diff(phi, t) to omega\_phi

T2 = (m2 \* (Ve\*\*2 + Vr\*\*2 + 2\*Ve\*Vr\*sp.cos(thetta - phi))) / 2

T = T1 + T2

#2 defining the potential energy

P1 = -m1\*g\*(R\_Ground - R\_Circle)\*sp.cos(thetta)

P2 = -m2\*g\*((R\_Ground - R\_Circle)\*sp.cos(thetta) + l\*sp.cos(phi))

P = P1 + P2

#Lagrange function

L = T - P

#equations

ur1 = sp.diff(sp.diff(L,omega\_thetta),t)-sp.diff(L,thetta)

#ur2 = sp.diff(sp.diff(L,omega\_phi),t)-sp.diff(L,phi)

print(ur1)

a11 = ur1.coeff(sp.diff(omega\_thetta, t), 1)

b1 = -(ur1.coeff(sp.diff(omega\_thetta, t), 0)).subs([(sp.diff(thetta, t), omega\_thetta)])

domega\_thettadt = b1/a11

# Constructing the system of differential equations

T = np.linspace(0, 100, Steps)

fomega\_thetta = sp.lambdify([thetta, omega\_thetta], domega\_thettadt, "numpy")

y0 = [0.05, 0.1] ################################################

sol = odeint(formY2, y0, T, args=(fomega\_thetta,))

Thetta = sol[:, 0]

Omega\_thetta = sol[:, 1]

# static

# Point O

X\_O = R\_Ground

Y\_O = R\_Ground

# Ground

alpha = np.linspace(-math.pi, 0, 500)

X\_Ground = R\_Ground + R\_Ground \* np.cos(alpha)

Y\_Ground = R\_Ground + R\_Ground \* np.sin(alpha)

# circle

beta = np.linspace(0, 2\*math.pi, 500)

X\_Circle = R\_Circle \* np.cos(beta)

Y\_Circle = R\_Circle \* np.sin(beta)

# constructing functions

# Point O1

x\_o1 = X\_O + (R\_Ground - R\_Circle) \* sp.sin(thetta)

y\_o1 = Y\_O - (R\_Ground - R\_Circle) \* sp.cos(thetta)

X\_O1 = sp.lambdify(thetta, x\_o1)

Y\_O1 = sp.lambdify(thetta, y\_o1)

XO1 = X\_O1(sol[:, 0])

YO1 = Y\_O1(sol[:, 0])

# Points C1 and C2 -- points on surface of the circle relative to point O1

X\_C1 = sp.lambdify([thetta], x\_o1 + R\_Circle\*sp.sin(thetta))

X\_C2 = sp.lambdify([thetta], x\_o1 - R\_Circle\*sp.sin(thetta))

Y\_C1 = sp.lambdify([thetta], y\_o1 + R\_Circle\*sp.cos(thetta))

Y\_C2 = sp.lambdify([thetta], y\_o1 - R\_Circle\*sp.cos(thetta))

XC1 = X\_C1(sol[:, 0])

XC2 = X\_C2(sol[:, 0])

YC1 = Y\_C1(sol[:, 0])

YC2 = Y\_C2(sol[:, 0])

# some settings

fig = plt.figure()

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

ax.axis("equal")

ax.set(xlim=(0, 12), ylim=(0, 12))

# plot zero state

Ground = ax.plot(X\_Ground, Y\_Ground, color='black', linewidth=2)

Point\_O = ax.plot(X\_O, Y\_O, color='red', linewidth=4)

Draw\_palka = ax.plot([X\_O, XO1[0]], [Y\_O, YO1[0]], 'r--')[0]

Draw\_palka1 = ax.plot([XC1[0], XC2[0]], [YC1[0], YC2[0]], 'b')[0]

Draw\_Circle = ax.plot(

X\_Circle + XO1[0], Y\_Circle + YO1[0], color='blue', linewidth=1)[0]

Draw\_point\_O1 = ax.plot(XO1[0], YO1[0], color='blue',

linewidth=3, marker='o')[0]

# graphs

fig\_for\_graphs = plt.figure(figsize=[13, 7])

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

ax\_for\_graphs.plot(T, Thetta, color='red')

ax\_for\_graphs.set\_title('Thetta(t)')

ax\_for\_graphs.set(xlim=[0, 12])

ax\_for\_graphs.grid(True)

ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2, 2, 3)

ax\_for\_graphs.plot(T, Omega\_thetta, color='black')

ax\_for\_graphs.set\_title("thetta'(t) = omega\_thetta(t)")

ax\_for\_graphs.set(xlim=[0, 12])

ax\_for\_graphs.grid(True)

# function for updating state of the system

def kinoteatr\_five\_zvezd\_na\_novokuzneckoy(i):

Draw\_point\_O1.set\_data(XO1[i], YO1[i])

Draw\_Circle.set\_data(X\_Circle + XO1[i], Y\_Circle + YO1[i])

Draw\_palka.set\_data([X\_O, XO1[i]], [Y\_O, YO1[i]])

Draw\_palka1.set\_data([XC1[i], XC2[i]], [YC1[i], YC2[i]])

return [Draw\_point\_O1, Draw\_Circle, Draw\_palka, Draw\_palka1]

anime = FuncAnimation(fig, kinoteatr\_five\_zvezd\_na\_novokuzneckoy,

frames=Steps, interval=30)

plt.show()

**Результат работы:**

|  |
| --- |
|  |
|  |

**Замечание насчет вычислений.**

В данной ЛР я посчитал период малых колебаний. Когда решал сам, на бумаге, получил значение 5,49384. Это с хорошей точностью совпало с тем, что вывела программа на питоне, так как в программе было ~ 5,5