Московский авиационный институт

(национальный исследовательский университет)

Институт № 8 «Информационные технологии и прикладная математика»

**Лабораторная работа №4**

**по курсу «Теоретическая механика»**

**Затухающие колебания**

Выполнил студент группы М8О-207Б-20

Михеева Кристина Олеговна

Преподаватель: Чекина Евгения Алексеевна

Оценка:

Дата: 26.12.2021

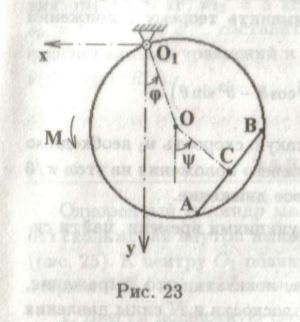
Москва, 2021

**Вариант № «23»**

**Задание:**

Получить уравнения Лагранжа 2го рода и по ним реализовать анимацию для малых колебаний.

**Механическая система:**

****

**Вычисления:**

**Текст программы**

Основная:

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

from sympy import simplify

def odesys(y, t, rCircle, mCircle, mRod, k, l, M0, gamma, OC, g):

dy = np.zeros(4)

dy[0] = y[2]

dy[1] = y[3]

a11 = (2 \* mCircle + mRod) \* (rCircle \*\* 2)

a12 = mRod \* rCircle \* OC \* np.cos(y[1] - y[0])

a21 = rCircle \* OC \* np.cos(y[1] - y[0])

a22 = rCircle \*\* 2 - (2 / 3) \* (l \*\* 2)

b1 = mRod \* rCircle \* OC \* (y[3] \*\* 2) \* np.sin(y[1] - y[0]) - (mCircle + mRod) \* g \* \

rCircle \* np.sin(y[0]) + M0 \* np.sin(gamma \* t) - k \* y[2]

b2 = -rCircle \* OC \* (y[2] \*\* 2) \* np.sin(y[1] - y[0]) - OC \* 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

PI = math.pi

# t = sp.Symbol('t')

t\_fin = 20

T = np.linspace(0, t\_fin, 1001)

# defining parameters

# parameters of rod

mCircle = 2

rCircle = 0.5

# parameters of rod

l = 0.25

rodLength = OA = 2 \* l

mRod = 0

# initial parameters of motion and const value

M0 = 0 # amplitude of strength moment

gamma = 3 \* PI / 2 # const value in strength moment

k = 1 # coefficient in strength moment of resistance

t0 = phi0 = dpsi0 = 0 # initial data

dphi0 = 0.2

psi0 = PI / 6

OC = math.sqrt(rCircle \*\* 2 - l \*\* 2) # distance between point O and point C

g = 9.81

y0 = [phi0, psi0, dphi0, dpsi0]

t = sp.Symbol('t')

phi = sp.Function('phi')(t)

psi = sp.Function('psi')(t)

dphi = sp.Function('dphi')(t)

dpsi = sp.Function('dpsi')(t)

JO = mCircle\* (rCircle\*\*2)

JO1 = mCircle \* rCircle + JO

T1 = (JO1\*(dphi\*\*2))/2

Ve = rCircle\*dphi

Vr = sp.sqrt(rCircle \*\* 2 - l \*\* 2)\*dpsi

Vc = Ve\*\*2 + Vr\*\*2 + 2\*Ve\*Vr\*sp.cos(psi-phi)

JC = (mRod\*(l\*\*2))/3

T2 = (mRod\*Vc)/2+(JC\*(dpsi\*\*2))/2

TT = T1+T2

P1 = -mCircle\*g\*sp.cos(phi)

P2 = -mRod\*g \* (rCircle\*sp.cos(phi)+sp.sqrt(rCircle\*\*2-l\*\*2))\*sp.cos(psi)

P = P1 + P2

M = M0\*sp.sin(gamma \* t)

Mc = k\*dphi

delAphi = M \* sp.diff(phi, t) - Mc \* sp.diff(phi, t)

Qphi = simplify(delAphi/sp.diff(phi, t))

L = TT -P

ur1 = sp.diff(sp.diff(L,dphi),t)-sp.diff(L,phi)-Qphi

ur2 = sp.diff(sp.diff(L,dpsi),t)-sp.diff(L,psi)

Y = odeint(odesys, y0, T, (rCircle, mCircle, mRod, k, l, M0, gamma, OC, g))

phi = Y[:, 0]

psi = Y[:, 1]

dphi = Y[:, 2]

dpsi = Y[:, 3]

XpO = (-1) \* rCircle \* np.sin(phi) # X component of point O

YpO = rCircle \* np.cos(phi) # Y component of point O

XpC = XpO + OC \* np.sin(psi)

YpC = YpO + OC \* np.cos(psi)

XpA = XpC + l \* np.cos(psi)

YpA = YpC - l \* np.sin(psi)

XpB = XpC - l \* np.cos(psi)

YpB = YpC + l \* np.sin(psi)

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

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

ax\_for\_graphs.plot(T, phi, color='blue')

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(2,2,2)

ax\_for\_graphs.plot(T, dphi, color='green')

ax\_for\_graphs.set\_title("phi'(t)")

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

ax\_for\_graphs.grid(True)

# here we start to plot

fig = plt.figure(figsize=(17, 10))

ax1 = fig.add\_subplot(1, 2, 1)

ax1.axis('equal')

ax1.set(xlim=[-2, 2], ylim=[-2, 3])

ax1.invert\_xaxis()

ax1.invert\_yaxis()

ax1.plot(0, 0, marker='o', color='blue') # point O1

PCIRCLE, = ax1.plot(XpO[0], YpO[0], 'b', marker='o', markersize=3)

PpA, = ax1.plot(XpA[0], YpA[0], 'g', marker='o', markersize=2)

PpC, = ax1.plot(XpC[0], YpC[0], 'r', marker='o', markersize=3)

PpB, = ax1.plot(XpB[0], YpB[0], 'black', marker='o', markersize=2)

Rod, = ax1.plot([XpA[0], XpB[0]], [YpA[0], YpB[0]], 'r')

def Kino(i):

CIRCLE = plt.Circle((XpO[i], YpO[i]), rCircle, color='b', fill=False)

ax1.add\_artist(CIRCLE)

PCIRCLE.set\_data(XpO[i], YpO[i])

PpC.set\_data(XpC[i], YpC[i])

PpA.set\_data(XpA[i], YpA[i])

Rod.set\_data([XpA[i], XpB[i]], [YpA[i], YpB[i]])

PpB.set\_data(XpB[i], YpB[i])

return [PCIRCLE, CIRCLE, PpC, Rod, PpA, PpB]

anima = FuncAnimation(fig, Kino, frames=len(T), interval=10, blit=True)

plt.show()

**Результат работы программы:**

|  |  |
| --- | --- |
|  |  |