Московский авиационный институт
(национальный исследовательский университет)
Институт № 8 «Информационные технологии и прикладная математика»

Лабораторная работа №3
по курсу «Теоретическая механика»
Уравнение Лагранжа

Выполнил студент группы М8О-207Б-20
Мерц Савелий Павлович
Преподаватель: Чекина Евгения Алексеевна

Оценка:

Дата: 21/12/2021

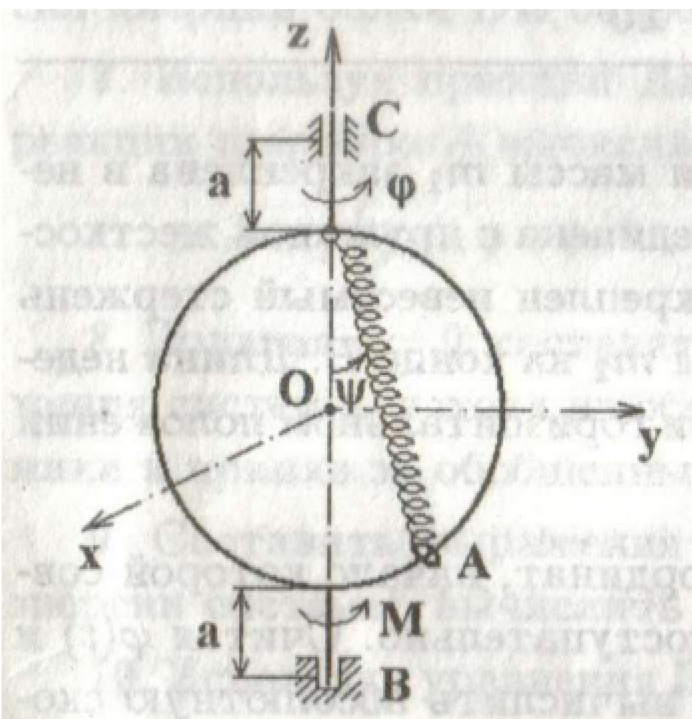
Москва, 2021

Вариант №«19»

Задание:

Написать на языку python программу визуализирующую кинематику плоского движения механической системы или сложного движения, согласно варианту, используя свободные координаты полученные из уравнения Лагранжа. Кроме анимации системы вывести справа в том же окне графики скоростей обозначенных точек системы.

Механическая система:



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

def formY(y, t, fV, fOm):
    y1,y2,y3,y4 = y
    dydt = [y3,y4,fV(y1,y2,y3,y4),fOm(y1,y2,y3,y4)]
    return dydt

# defining parameters
# the angle of the plane (and the prism)
alpha = math.pi / 6
M = 1
m = 0.1
R = 0.3
c = 20
l0 = 0.2
g = 9.81

# 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'
phi=sp.Function('phi')(t)
psi=sp.Function('psi')(t)
Vphi=sp.Function('Vphi')(t)
Vpsi=sp.Function('Vpsi')(t)
```

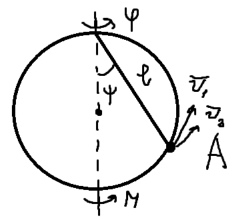
```
l = 2 * R * sp.cos(psi)  # длина пружины
#constructing the Lagrange equations
#1 defining the kinetic energy
TT1 = M * R**2 * Vphi**2 / 2
V1 = 2*Vpsi * R
V2 = Vphi * R * sp.sin(2 * psi)
Vr2 = V1**2 + V2**2
TT2 = m * Vr2 / 2
TT = TT1+TT2
# 2 defining the potential energy
Pi1 = 2 * R * m * g * sp.sin(psi)**2
Pi2 = (c * (l - l0)**2) / 2
Pi = Pi1+Pi2
# 3 Not potential force
M = alpha * phi**2;

# Lagrange function
L = TT-Pi

# equations
ur1 = sp.diff(sp.diff(L,Vphi),t)-sp.diff(L,phi) - M
ur2 = sp.diff(sp.diff(L,Vpsi),t)-sp.diff(L,psi)

# isolating second derivatives(dV/dt and dom/dt) using Kramer's method
a11 = ur1.coeff(sp.diff(Vphi,t),1)
a12 = ur1.coeff(sp.diff(Vpsi,t),1)
a21 = ur2.coeff(sp.diff(Vphi,t),1)
a22 = ur2.coeff(sp.diff(Vpsi,t),1)
```

```
b1 =
-(ur1.coeff(sp.diff(Vphi,t),0)).coeff(sp.diff(Vpsi,t),0).subs([(sp.diff(phi,t)
,Vphi), (sp.diff(psi,t), Vpsi)])

b2 =
-(ur2.coeff(sp.diff(Vphi,t),0)).coeff(sp.diff(Vpsi,t),0).subs([(sp.diff(phi,t)
,Vphi), (sp.diff(psi,t), Vpsi)])


detA = a11*a22-a12*a21

detA1 = b1*a22-b2*a21

detA2 = a11*b2-b1*a21


dVdt = detA1/detA

domdt = detA2/detA


countOfFrames = 2000


# Constructing the system of differential equations

T = np.linspace(0, 25, countOfFrames)

# Pay attention here, the function lambdify translate function from the sympy
to numpy and then form arrays much more

# faster then we did using subs in previous lessons!

fVphi = sp.lambdify([phi,psi,Vphi,Vpsi], dVdt, "numpy")

fVpsi = sp.lambdify([phi,psi,Vphi,Vpsi], domdt, "numpy")

y0 = [0, np.pi/6, -0.5, 0]

sol = odeint(formY, y0, T, args = (fVphi, fVpsi))


#sol - our solution

#sol[:,0] - phi

#sol[:,1] - psi

#sol[:,2] - dphi/dt

#sol[:,3] - dpsi/dt
```

```
# Ввод переменной t и радиусов необходимых окружностей + ввод угла поворота шариков

t = sp.Symbol('t')


# Построение графика и подграфика с выравниванием осей

fig = plt.figure(figsize=(17, 8))

ax1 = fig.add_subplot(1, 2, 1)

ax1.axis('equal')


phi = sol[:,0]

psi = sol[:,1]

Vphi = sol[:,2]

Vpsi = sol[:,3]


w = np.linspace(0, 2 * math.pi, countOfFrames)

conline, = ax1.plot([sp.sin(2*psi[0]) * R * sp.cos(phi[0]), 0], [-1, R], 'black')

P, = ax1.plot(sp.sin(2*psi[0]) * R * sp.cos(phi[0]), sp.cos(2*psi[0]) * R, marker='o', color='black')

Circ, = ax1.plot(R * sp.cos(phi[0]) * np.cos(w), R * np.sin(w), 'black')


#Доп графики

ax2 = fig.add_subplot(4, 2, 2)

ax2.plot(T, Vphi)

ax2.set_xlabel('T')

ax2.set_ylabel('Vphi')

ax3 = fig.add_subplot(4, 2, 4)

ax3.plot(T, Vpsi)

ax3.set_xlabel('T')

ax3.set_ylabel('Vpsi')


def anima(i):
```

```
    P.set_data(sp.sin(2*psi[i]) * R * sp.cos(phi[i]), sp.cos(2*psi[i]) * R)
    conline.set_data([sp.sin(2*psi[i]) * R * sp.cos(phi[i]), 0], [sp.cos(2*psi[i]) * R, R])
    Circ.set_data(R * sp.cos(phi[i]) * np.cos(w), R * np.sin(w))
    return Circ, P, conline


anim = FuncAnimation(fig, anima, frames=countOfFrames, interval=1, blit=True)
plt.show()
```

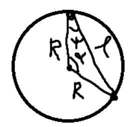
Вывод уравнения лагранжа:



$v = \omega R$

$$\Pi_A = mg(2R - l\cos\psi) = mg(2R - 2R\cos^2\psi) = 2Rmg\sin^2\psi$$

$$\Pi_{пруж} = \frac{c(l - l_0)^2}{2} = \frac{c(2R\cos\psi - l_0)^2}{2}$$



$$\alpha = 180 - 2\psi$$

$$l = \sqrt{2R^2 - 2R^2\cos\alpha} = \sqrt{2}R\sqrt{1-\cos\alpha} = 2R\cos\psi.$$

$$v_A^2 = v_1^2 + v_2^2 = 4\dot\psi^2R^2 + (l\sin\psi\cdot\dot\varphi)^2 = 4\dot\psi^2R^2 + 4R^2\cos^2\psi\sin^2\psi\,\dot\varphi^2$$

$$T_A = \frac{m v_A^2}{2} = 2m\dot\psi^2R^2 + \frac{mR^2\dot\varphi^2\sin^2(2\psi)}{2}$$

$$T_{кольц} = \frac{J_{кол}\omega^2}{2} = \frac{MR^2\dot\varphi^2}{2}$$

$$L = T_A + T_{кольц} - \Pi_A - \Pi_{пруж} = 2m\dot\psi^2R^2 + \frac{mR^2\dot\varphi^2\sin^2(2\psi)}{2} + \frac{MR^2\dot\varphi^2}{2} - 2Rmg\sin^2\psi - \frac{c(2R\cos\psi - l_0)^2}{2}$$

$$M_z = \alpha\varphi^2;\ \delta A_\varphi = M_z\,\delta\varphi;\ Q_\varphi = M_z = \alpha\varphi^2$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot\psi}\right) - \frac{\partial L}{\partial\psi} = 0 \Rightarrow УР_1 = 4mR^2\ddot\psi - mR^2\dot\varphi\sin(4\psi) + 2Rmg\sin(2\psi) - 2cR(2R\cos\psi - l_0)\sin\psi =$$
$$= mR(4\ddot\psi - \dot\varphi\sin(4\psi)) + 2mg\sin(2\psi) - 2c(2R\cos\psi - l_0)\sin\psi = 0$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot\varphi}\right) - \frac{\partial L}{\partial\varphi} = Q_\varphi \Rightarrow УР_2 = mR^2\sin^2(2\psi)\ddot\varphi + \frac{MR^2}{2}\ddot\varphi + 4mR^2\sin(2\psi)\cos(2\psi)\dot\psi\dot\varphi =$$
$$= R^2\ddot\varphi\left(\frac{M}{2} + m\sin^2(2\psi)\right) + 2mR^2\sin(4\psi)\dot\psi\dot\varphi$$

Результат работы: