Московский авиационный институт

(национальный исследовательский университет)

Институт № 8 «Компьютерные науки и прикладная математика»

Лабораторная работа № 4
по курсу "Теоретическая механика и компьютерное моделирование"

Положение равновесия системы

Выполнил студент группы М8О-206Б-21

Синюков А.С.

Преподаватель: Чекина Евгения Алексеевна

Оценка:

Дата: 07.01.2023

Москва, 2022

Вариант № 24

Задание:
Невесомая трубка, выгнутая в форме кругового кольца радиуса r, закреплена на платформе, которая имеет массу m_1 и может скользить без трения по горизонтальной плоскости. В трубке движется без трения точечный груз M массы m_2.
Считая угол θ малым, составить уравнение малых колебаний точки M в окрестности ее нижнего положения. Найти период малых колебаний.
Рисунок:

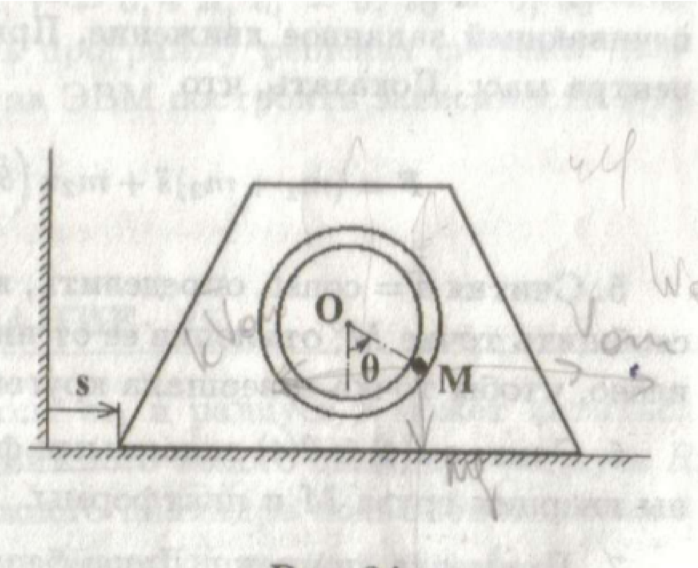

Рис. 24

Текст программы:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import sympy as sp
import math
import time
from scipy.integrate import odeint

def Platform(x0, y0): #линия в форме трапеции из 5 точек
    PX = [x0 - 10, x0 - 5, x0 + 5, x0 + 10, x0 - 10]
    PY = [y0 - 7.5, y0 + 10, y0 + 10, y0 - 7.5, y0 - 7.5]
    return PX, PY



def formY(y, t, f_om):
    y1, y2 = y
    dy_dt = [y2, f_om(y1, y2)]
    return dy_dt
```

```
#Начальные данные
radius = 0.4
mass_m1 = 20
mass_m2 = 5
g = 9.8

# Составление законов движений, уравнений Лагранжа, диффуров
t = sp.Symbol('t')

s = 0 # из условий задачи
phi = sp.Function('phi')(t)
v = 0
om = sp.Function('om')(t)


xa = 0.8 + 5 * sp.sin(phi)
ya = 7.5 - 5 * sp.cos(phi)


#v_2_cm = v**2+(om**2)*(radius**2)/4 - v*om*radius*sp.cos(phi)
#moment_inertia = (mass_m2 * radius*radius) # момент инерции диска относительно центра

kin_energy = (mass_m2 * om * om * radius * radius)/2
pot_energy = -(mass_m2 * g * radius * sp.cos(phi))

L = kin_energy - pot_energy

# ur1 = sp.diff(sp.diff(L, v), t) - sp.diff(L, s)
ur2 = sp.diff(sp.diff(L, om), t) - sp.diff(L, phi)

# a11 = ur1.coeff(sp.diff(v, t), 1)
# a12 = ur1.coeff(sp.diff(om, t), 1)
# a21 = ur2.coeff(sp.diff(v, t), 1)
a22 = ur2.coeff(sp.diff(om, t), 1)
# b1 = -(ur1.coeff(sp.diff(v, t), 0)).coeff(sp.diff(om, t), 0).subs([(sp.diff(s, t), v), (sp.diff(phi, t), o
b2 =  -ur2.coeff(sp.diff(om, t), 0).subs(sp.diff(phi, t), om)


# det = a11 * a22 - a12 * a21
# det1 = b1 * a22 - b2 * a12
# det2 = a11 * b2 - b1 * a21

# dv_dt = det1 / det
# dom_dt = det2 / det

dom_dt = b2 / a22

# построения
T = np.linspace(0, 20, 1000)
```

```python
y0 = [0, 2]

# f_v = sp.lambdify([s, phi, v, om], dv_dt, "numpy")
f_om = sp.lambdify([phi, om], dom_dt, "numpy")
sol = odeint(formY, y0, T, args=(f_om,))

XA_def = sp.lambdify(phi, xa)
YA_def = sp.lambdify(phi, ya)
Cord_def = sp.lambdify(t, t)

XA = XA_def(sol[:, 0])
YA = YA_def(sol[:, 0])
Cord = Cord_def(T)


fig = plt.figure(figsize = (20, 10))

ax1 = fig.add_subplot(1, 2, 1)
ax1.axis('equal')
ax1.set(xlim=[-20, 20], ylim=[-20, 30])
# ax1.plot([X.min() - 10, X.max() + 10], [0, 0], 'black')
# ax1.plot([X.min() - 10, X.min() - 10], [0, Y.max() + 15], 'black')

PrX, PrY = Platform(0.8, 7.5)
Prism = ax1.plot(PrX, PrY, 'blue')[0]

radius, = ax1.plot([0.8, XA[0]], [7.5, YA[0]], 'black')

Phi = np.linspace(0, 6.28, 20)
r = 0.2
Point = ax1.plot(XA[0] + r * np.cos(Phi), YA[0] + r * np.sin(Phi), 'blue')[0]

# ax2 = fig.add_subplot(4, 2, 2)
# ax2.set(xlim=[0, 15], ylim=[-1.5, 1.5])
# Vgx = [Cord[0]]
# Vgy = [sol[:, 2][0]]
# V_graph, = ax2.plot(Vgx, Vgy, 'blue')
# ax2.set_ylabel('V')

ax2 = fig.add_subplot(4, 2, 2)
ax2.set(xlim=[0, 15], ylim=[-4, 4])
Phigx = [Cord[0]]
Phigy = [sol[:, 0][0]]
Phi_graph, = ax2.plot(Phigx, Phigy, 'blue')
ax2.set_ylabel('PHI')

ax3 = fig.add_subplot(4, 2, 4)
ax3.set(xlim=[0, 15], ylim=[-4.0, 4.0])
Omgx = [Cord[0]]
Omgy = [sol[:, 1][0]]
```

```python
Om_graph, = ax3.plot(Omgx, Omgy, 'orange')
ax3.set_ylabel('OMEGA')

plt.subplots_adjust(wspace = 0.2, hspace = 0.2)

# Анимируем
def anima(i):
    PrX, PrY = Platform(0.8, 7.5)
    Prism.set_data(PrX, PrY)
    radius.set_data([0.8, XA[i]], [7.5, YA[i]])
    Point.set_data(XA[i] + r * np.cos(Phi), YA[i] + r * np.sin(Phi))
    # Vgx.append(Cord[i])
    # Vgy.append(sol[:, 2][i])
    Phigx.append(Cord[i])
    Phigy.append(sol[:, 0][i])
    Omgx.append(Cord[i])
    Omgy.append(sol[:, 1][i])
    # V_graph.set_data(Vgx, Vgy)
    Phi_graph.set_data(Phigx, Phigy)
    Om_graph.set_data(Omgx, Omgy)
    if(-0.005 < sol[:, 0][i] < 0.005):
        print(time.perf_counter())
    return Prism, radius, Point, Om_graph, Phi_graph # , V_graph

anim = FuncAnimation(fig, anima, frames = 1000, interval = 1, blit = True)

plt.show()
```

Выводы формул:

$$T = \frac{m v^2}{2} = \frac{m \omega^2 r^2}{2} \qquad П = -mgr\cos\theta$$

$$L = T - П = \frac{m\omega^2 r^2}{2} + mgr\cos\theta$$

$$\frac{\partial L}{\partial \omega} = m\omega r^2 \qquad \frac{d}{dt}\left(\frac{\partial L}{\partial \omega}\right) = m\varepsilon r^2$$

$$\frac{\partial L}{\partial \theta} = -mgr\sin\theta$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \omega}\right) - \frac{\partial L}{\partial \theta} = m\varepsilon r^2 + mgr\sin\theta = 0 \Rightarrow$$

$$\Rightarrow \varepsilon r + g\sin\theta = r\ddot{\theta} + g\sin\theta = 0$$

При малых колебаниях $\sin\theta \approx \theta$

$$r\ddot{\theta} + g\theta = 0$$

$$\theta = C_1 \cos\sqrt{\frac{g}{r}}\,t + C_2 \sin\sqrt{\frac{g}{r}}\,t \Rightarrow$$

$$\Rightarrow \theta = A\sin\left(\sqrt{\frac{g}{r}}\,t + B\right)$$

$$T = \frac{2\pi}{\omega} = 2\pi\sqrt{\frac{r}{g}}$$

При $r = 0{,}4$ м:

$$T = 2\pi\sqrt{\frac{r}{g}} \approx 1{,}2694 \text{ с.}$$

$$\Pi = -mgr\cos\theta$$

$$\frac{\partial \Pi}{\partial \theta} = mgr\sin\theta = 0 \Rightarrow \theta = \pi k,\ k = 0, 1.$$

$$\frac{\partial^2 \Pi}{\partial \theta^2} = mgr\cos\theta > 0 \Rightarrow \theta = 0. \Rightarrow$$

$\Rightarrow \theta = 0$ – устойчивое положение равновесия.

Программа показала период колебаний примерно разный 1.4 с, что близко к расчетам.

Результат работы программы: