Московский авиационный институт

(национальный исследовательский университет)

Институт № 8 «Информационные технологии и прикладная математика»

**Лабораторная работа №3**

**по курсу «Теоретическая механика»**

**Динамика системы**

Выполнил студент группы М8О-207Б-21

Дубровин Дмитрий Константинович

Преподаватель: Беличенко Михаил Валериевич

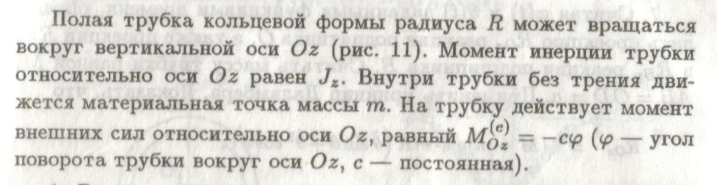
Оценка: -

Дата: 29.12.2022

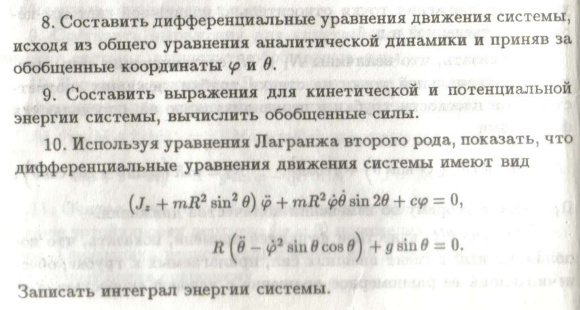
Москва, 2022

**Вариант №11**

**Задание:**

Численно решить дифференциальные уравнения движения механической системы в среде Octave (или Matlab), сделать задание №12 курсовой и построить анимацию движения системы.

**Механическая система:**

****

**Текст программы**

import numpy as np

import sympy as sp

import math

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from matplotlib.patches import Circle

from scipy.integrate import odeint

import mpl\_toolkits.mplot3d.art3d as art3d

def odesys(y, t, g, m, J, R, c):

dy = np.zeros(4)

dy[0] = y[2]

dy[1] = y[3]

a11 = J + m \* R \*\* 2 \* np.sin(y[1]) \*\* 2

a12 = 0

a21 = 0

a22 = R

b1 = -c \* y[0] - m \* R \*\* 2 \* y[2] \* y[3] \* np.sin(2 \* y[1])

b2 = -g \* np.sin(2 \* y[1]) + R \* y[2] \*\* 2 \* np.sin(y[1]) \* np.cos(y[1])

dy[2] = (b1 \* a22 - b2 \* a12) / (a11 \* a22 - a12 \* a21)

dy[3] = (b2 \* a11 - b1 \* a21) / (a11 \* a22 - a12 \* a21)

return dy

tn = np.linspace(0, 10, 2000)

xn = np.zeros\_like(tn)

yn = np.zeros\_like(tn)

zn = np.zeros\_like(tn)

g = 9.81

m = 1

J = 0.5

R = 0.5

c = 2

phi0 = 0.3

theta0 = 0.6

dphi0 = -0.2

dtheta0 = 0.3

y0 = [phi0, theta0, dphi0, dtheta0]

Y = odeint(odesys, y0, tn, (g, m, J, R, c))

phi = Y[:, 0]

theta = Y[:, 1]

for i in range(len(tn)):

xn[i] = R \* np.sin(theta[i]) \* np.cos(phi[i])

yn[i] = R \* np.sin(theta[i]) \* np.sin(phi[i])

zn[i] = - R \* np.cos(theta[i])

fig = plt.figure()

ax = fig.add\_subplot(projection="3d")

ax.set(xlim=[-1, 1], ylim=[-1, 1], zlim=[-1, 1])

fig2 = plt.figure()

ax1 = fig2.add\_subplot(2, 2, 1)

ax1.plot(Y[:, 0])

ax1.set\_title("$\\varphi$")

ax2 = fig2.add\_subplot(2, 2, 2)

ax2.plot(Y[:, 1])

ax2.set\_title("$\\theta$")

ax3 = fig2.add\_subplot(2, 2, 3)

ax3.plot(Y[:, 2])

ax3.set\_title("$\\varphi'$")

ax4 = fig2.add\_subplot(2, 2, 4)

ax4.plot(Y[:, 3])

ax4.set\_title("$\\theta'$")

def plot\_vector(fig, orig, v, color='blue'):

ax = fig.gca(projection='3d')

orig = np.array(orig)

v = np.array(v)

ax.quiver(orig[0], orig[1], orig[2], v[0], v[1], v[2], color=color)

ax.set\_xlim(0, 10)

ax.set\_ylim(0, 10)

ax.set\_zlim(0, 10)

ax = fig.gca(projection='3d')

return fig

def rotation\_matrix(d):

sin\_angle = np.linalg.norm(d)

if sin\_angle == 0: return np.identity(3)

d /= sin\_angle

eye = np.eye(3)

ddt = np.outer(d, d)

skew = np.array([[0, d[2], -d[1]],

[-d[2], 0, d[0]],

[d[1], -d[0], 0]], dtype=np.float64)

M = ddt + np.sqrt(1 - sin\_angle \*\* 2) \* (eye - ddt) + sin\_angle \* skew

return M

def pathpatch\_2d\_to\_3d(pathpatch, z, normal):

if type(normal) is str: # Translate strings to normal vectors

index = "xyz".index(normal)

normal = np.roll((1.0, 0, 0), index)

normal /= np.linalg.norm(normal) # Make sure the vector is normalised

path = pathpatch.get\_path() # Get the path and the associated transform

trans = pathpatch.get\_patch\_transform()

path = trans.transform\_path(path) # Apply the transform

pathpatch.\_\_class\_\_ = art3d.PathPatch3D # Change the class

pathpatch.\_code3d = path.codes # Copy the codes

pathpatch.\_facecolor3d = pathpatch.get\_facecolor # Get the face color

verts = path.vertices # Get the vertices in 2D

d = np.cross(normal, (0, 0, 1)) # Obtain the rotation vector

M = rotation\_matrix(d) # Get the rotation matrix

pathpatch.\_segment3d = np.array([np.dot(M, (x, y, 0)) + (0, 0, z) for x, y in verts])

def pathpatch\_translate(pathpatch, delta):

pathpatch.\_segment3d += delta

def plot\_plane(ax, point, normal, size=10, color='y'):

p = Circle((0, 0), size, facecolor=color, alpha=.2)

ax.add\_patch(p)

pathpatch\_2d\_to\_3d(p, z=0, normal=normal)

pathpatch\_translate(p, (point[0], point[1], point[2]))

def update(i):

point.set\_data\_3d(xn[i], yn[i], zn[i])

global cr

cr.remove()

cr = Circle((0, 0), R)

cr.set\_alpha(0.4)

ax.add\_patch(cr)

pathpatch\_2d\_to\_3d(cr, z=0, normal=[yn[i], -xn[i], 0])

return point

point = ax.plot(xn[0], yn[0], zn[0], marker=".", color="black")[0]

cr = Circle((0, 0), R)

cr.set\_alpha(0.4)

ax.add\_patch(cr)

pathpatch\_2d\_to\_3d(cr, z=0, normal=[0, yn[0], 0])

a = FuncAnimation(fig, update, frames=len(tn), interval=10)

plt.show()

**Результат работы:**

