Московский авиационный институт

(национальный исследовательский университет)

Институт № 8 «Информационные технологии и прикладная математика»

**Лабораторная работа №2**

**по курсу «Теоретическая механика»**

**Анимация системы**

Выполнил студент группы М8О-1302С-08

Дубровин Дмитрий Константинович

Преподаватель: Беличенко Михаил Валериевич

Оценка: -

Дата: 29.12.2022

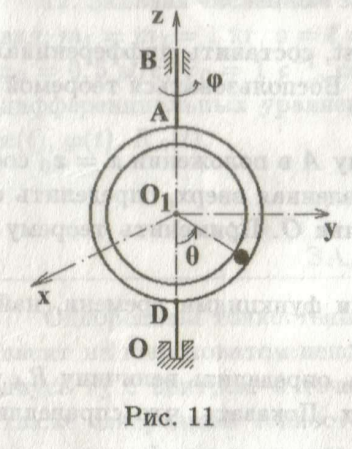
Москва, 2021

**Вариант №11**

**Задание:**

Реализовать анимацию движения механической системы в среде Octave (или Matlab)

**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

import mpl\_toolkits.mplot3d.art3d as art3d

R = 5

k\_phi = 0

k\_theta = 0.4

t = sp.Symbol('t')

phi = k\_phi \* np.pi \* t

theta = k\_theta \* np.pi \* t

x = R \* sp.sin(theta) \* sp.cos(phi)

y = R \* sp.sin(theta) \* sp.sin(phi)

z = - R \* sp.cos(theta)

vx = sp.diff(x, t)

vy = sp.diff(y, t)

vz = sp.diff(z, t)

wx = sp.diff(vx, t)

wy = sp.diff(vy, t)

wz = sp.diff(vz, t)

print("x(t) = ", x)

print("y(t) = ", y)

print("z(t) = ", z)

print("vx(t) = ", vx)

print("vy(t) = ", vy)

print("vz(t) = ", vz)

print("wx(t) = ", wx)

print("wy(t) = ", wy)

print("wz(t) = ", wz)

tn = np.linspace(0, 20, 500)

xn = np.zeros\_like(tn)

yn = np.zeros\_like(tn)

zn = np.zeros\_like(tn)

vxn = np.zeros\_like(tn)

vyn = np.zeros\_like(tn)

vzn = np.zeros\_like(tn)

wxn = np.zeros\_like(tn)

wyn = np.zeros\_like(tn)

wzn = np.zeros\_like(tn)

for i in range(len(tn)):

xn[i] = sp.Subs(x, t, tn[i])

yn[i] = sp.Subs(y, t, tn[i])

zn[i] = sp.Subs(z, t, tn[i])

vxn[i] = sp.Subs(vx, t, tn[i])

vyn[i] = sp.Subs(vy, t, tn[i])

vzn[i] = sp.Subs(vz, t, tn[i])

wxn[i] = sp.Subs(wx, t, tn[i])

wyn[i] = sp.Subs(wy, t, tn[i])

wzn[i] = sp.Subs(wz, t, tn[i])

fig = plt.figure()

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

ax.set(xlim=[-8, 8], ylim=[-8, 8], zlim=[-8, 8])

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])

velocity.set\_data\_3d(xn[i] + vxn[i], yn[i] + vyn[i], zn[i] + vzn[i])

vv.set\_data\_3d([xn[i], xn[i] + vxn[i]], [yn[i], yn[i] + vyn[i]], [zn[i], zn[i] + vzn[i]])

acceleration.set\_data\_3d(xn[i] + wxn[i], yn[i] + wyn[i], zn[i] + wzn[i])

av.set\_data\_3d([xn[i], xn[i] + wxn[i]], [yn[i], yn[i] + wyn[i]], [zn[i], zn[i] + wzn[i]])

global c

c.remove()

c = Circle((0, 0), R)

c.set\_alpha(0.4)

ax.add\_patch(c)

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

return point

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

c = Circle((0, 0), R)

velocity = ax.plot(xn[0] + vxn[0], yn[0] + vyn[0], zn[0] + vzn[0], marker=".", color="blue")[0]

vv = ax.plot([xn[0], xn[0] + vxn[0]], [yn[0], yn[0] + vyn[0]], [zn[0], zn[0] + vzn[0]], color="blue")[0]

acceleration = ax.plot(xn[0] + wxn[0], yn[0] + wyn[0], zn[0] + wzn[0], marker=".", color="black")[0]

av = ax.plot([xn[0], xn[0] + wxn[0]], [yn[0], yn[0] + wyn[0]], [zn[0], zn[0] + wzn[0]], color="red")[0]

c.set\_alpha(0.4)

ax.add\_patch(c)

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

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

plt.show()

**Результат работы:**

