Московский авиационный институт

(национальный исследовательский университет)

Институт № 8 «Информационные технологии и прикладная математика»

**Лабораторная работа №1**

**по курсу «Теоретическая механика»**

**Анимация точки**

Выполнил студент группы М8О-207Б-20

Михеева Кристина Олеговна

Преподаватель: Чекина Евгения Алекмеевна

Оценка:

Дата: 3.10.2021

Москва, 2021

**Вариант № «17»**

**Задание:**

Построить заданную траекторию и анимацию движения точки, а также отобразить стрелки скорости и ускорения.

**Закон движения точки:**

r(t) **=** 2+0.5sin(12t)  φ(t) = 1.2t+0.2cos(12t)

**Текст программы**

Основная:

import matplotlib.pyplot as plt

import numpy as np

import math

from matplotlib.animation import FuncAnimation

import sympy as sp

t = sp.Symbol('t')

T = np.linspace(1, 15, 1000)

Radius = 4

Omega = 1

r = 2 + (0.5\*sp.sin(12 \* t))

phi = 1.2 \* t + (0.2 \* (sp.cos(12 \* t)))

x = r \* sp.cos(phi)

y = r \* sp.sin(phi)

Vx = sp.diff(x, t)

Vy = sp.diff(y, t)

Wx = sp.diff(Vx, t)

Wy = sp.diff(Vy, t)

V = sp.sqrt(Vx\*\*2 + Vy\*\*2) // вектор скорости

Wfull = sp.sqrt(Wx\*\*2 + Wy\*\*2) // вектор ускорения

Wtan = sp.diff(V) // тангенциальное ускорение

СurveVector = V\*\*2 / sp.sqrt(Wfull\*\*2 - Wtan\*\*2) // радиус кривизны

R = np.zeros\_like(T) // радиус вектор

PHI = np.zeros\_like(T)

X = np.zeros\_like(T)

Y = np.zeros\_like(T)

VX = np.zeros\_like(T)

VY = np.zeros\_like(T)

WX = np.zeros\_like(T)

WY = np.zeros\_like(T)

W = np.zeros\_like(T)

W\_T = np.zeros\_like(T)

RO = np.zeros\_like(T)

CVector = np.zeros\_like(T)

for i in np.arange(len(T)):

R[i] = sp.Subs(r, t, T[i])

PHI[i] = sp.Subs(phi, t, T[i])

X[i] = sp.Subs(x, t, T[i])

Y[i] = sp.Subs(y, t, T[i])

VX[i] = sp.Subs(Vx, t, T[i])

VY[i] = sp.Subs(Vy, t, T[i])

WX[i] = sp.Subs(Wx, t, T[i])

WY[i] = sp.Subs(Wy, t, T[i])

CVector[i] = sp.Subs(СurveVector, t, T[i])

XX = [0 for i in range(1000)]

YY = [0 for i in range(1000)]

fig = plt.figure()

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

ax1.axis('equal')

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

ax1.plot(X, Y)

P, = ax1.plot(X[0], Y[0], 'r', marker='o')

Vline, = ax1.plot([X[0], X[0] + VX[0]], [Y[0], Y[0] + VY[0]], 'r')

Vline2, = ax1.plot([X[0], X[0] + WX[0]], [Y[0], Y[0] + WY[0]], 'g')

Vline3, = ax1.plot([XX[0], X[0]], [YY[0], Y[0]], 'b')

Cvector, = ax1.plot([X[0], X[0] + (Y[0] + VY[0]) \* CVector[0] / sp.sqrt((Y[0] + VY[0])\*\*2 +(X[0] + VX[0])\*\*2)], [Y[0], Y[0] - (X[0] + VX[0]) \* CVector[0] / sp.sqrt((Y[0] + VY[0])\*\*2 + (X[0] + VX[0])\*\*2)], 'orange')

def Rot2D(X, Y, Alpha):

RX = X \* np.cos(Alpha) - Y \* np.sin(Alpha)

RY = X \* np.sin(Alpha) + Y \* np.cos(Alpha)

return RX, RY

ArrowX = np.array([-0.2\*Radius, 0, -0.2\*Radius])

ArrowY = np.array([0.1\*Radius, 0, -0.1\*Radius])

ArrowWX = np.array([-Radius, 0, -Radius])

ArrowWY = np.array([Radius, 0, -Radius])

ArrowRX = np.array([-0.1\*Radius, 0, -0.1\*Radius])

ArrowRY = np.array([0.05\*Radius, 0, -0.05\*Radius])

RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(VY[0], VX[0]))

RArrowWX, RArrowWY = Rot2D(ArrowWX, ArrowWY, math.atan2(WY[0], WX[0]))

RArrowRX, RArrowRY = Rot2D(ArrowRX, ArrowRY, math.atan2(Y[0], X[0]))

VArrow, = ax1.plot(RArrowX + X[0] + VX[0], RArrowY + Y[0] + VY[0], 'r')

WArrow, = ax1.plot(RArrowWX + X[0] + WX[0], RArrowY + Y[0] + WY[0], 'g')

RArrow, = ax1.plot(ArrowRX + X[0], ArrowRY + Y[0], 'b')

def anima(j):

P.set\_data(X[j], Y[j])

Vline.set\_data([X[j], X[j] + VX[j]], [Y[j], Y[j] + VY[j]])

Vline2.set\_data([X[j], X[j] + WX[j]], [Y[j], Y[j] + WY[j]])

Vline3.set\_data([XX[j], X[j]], [YY[j], Y[j]])

RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(VY[j], VX[j]))

VArrow.set\_data(RArrowX + X[j] + VX[j], RArrowY + Y[j] + VY[j])

RArrowWX, RArrowWY = Rot2D(ArrowWX, ArrowWY, math.atan2(WY[j], WX[j]))

WArrow.set\_data(RArrowWX + X[j] + WX[j], RArrowWY + Y[j] + WY[j])

RArrowRX, RArrowRY = Rot2D(ArrowRX, ArrowRY, math.atan2(Y[j], X[j]))

RArrow.set\_data(RArrowRX + X[j], RArrowRY + Y[j])

Cvector.set\_data([X[j], X[j] + (Y[j] + VY[j]) \* CVector[j] / sp.sqrt((Y[j] + VY[j]) \*\* 2 +(X[j] + VX[j]) \*\* 2)],[Y[j], Y[j] - (X[j] + VX[j]) \* CVector[j] /sp.sqrt((Y[j] + VY[j]) \*\* 2 + (X[j] + VX[j]) \*\* 2)])

return P, Vline, VArrow, Vline2, WArrow, Vline3, RArrow,Cvector

anim = FuncAnimation(fig, anima, frames=1500, interval=60, blit=True)

plt.grid()

plt.show()

**Функция Rot2D:**

def Rot2D(X, Y, Alpha):

RX = X \* np.cos(Alpha) - Y \* np.sin(Alpha)

RY = X \* np.sin(Alpha) + Y \* np.cos(Alpha)

return RX, RY

**Результат работы программы:**

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