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-208Б-20

Морозов Артем Борисович

Преподаватель: Волков Евгений Валерьевич

Оценка:

Дата: 13.12.2021

Подпись:

Москва, 2021

**Вариант № «16»**

**Задание:**

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

**Закон движения точки:**



**Текст программы**

import matplotlib.pyplot as plt

import numpy as np

import math

from matplotlib.animation import FuncAnimation

import sympy as sp

t = sp.Symbol('t') #вводим формулу, t считается за переменную

T = np.linspace(1, 15, 1000) #задаем массив Т с 1000 элементами от 1 до 15

Radius = 4

Omega = 1

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

phi = 1.8 \* t + (0.2 \* (sp.cos(12 \* t)) \*\* 2)

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)

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

Wtan = sp.diff(V)

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

print("Print function of coordinates:")

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

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

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

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

print("Vx(t) =", Vx)

print("Vy(t) =", Vy)

print("Wx(t) =", Wx)

print("Wy(t) =", Wy)

print("W(t) =", Wfull)

print("Wtan =", Wtan)

print("CurveVector=", CurveVector)

R = np.zeros\_like(T) #заполняем 1000 элементов масисва R нулями

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) #тангенциальное ускорение

CVector = np.zeros\_like(T) #вектор кривизны

for i in np.arange(len(T)): #заполняем массивы значениями в разные периоды времени (1000 значений от 1 до 15)

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)] #начало координат (откуда синий вектор идет)

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

fig = plt.figure()

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

ax1.axis('equal')

ax1.set(xlim=[-2.5, 2.5], ylim=[-3, 3])

ax1.plot(X, Y)

P, = ax1.plot(X[0], Y[0], 'r', marker='o') #точка, где тело сейчас находится (будет подсвечено красным)

#рисуем вектор в нулевой момент времени (в anima будем рисовать с 1 до последнего момента времени)

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): #матрица поворота

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

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

return RX, RY

#заполняем массивы векторов стрелочками (на конце вектора стрелочки, по аналогии с программой, которую делали в аудитории)

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(X[0], Y[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): #рисуем в каждый момент времени i нужные нам вектора

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

#вывод на экран

anim = FuncAnimation(fig, anima, frames=3000, interval=200, blit=True)

plt.grid()

plt.show()

**Результат работы программы:**

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

**Вывод программы:**

**Функция зависимости радиус вектора точки от времени:**

r(t) = (sin(12\*t) + 2)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2) **Функция зависимости угла вектора точки от времени:**

phi(t) = (sin(12\*t) + 2)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2) **Функция зависимости координаты Х точки от времени:**

x(t) = (sin(12\*t) + 2)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2)  
 **Функция зависимости координаты У точки от времени:**

y(t) = (sin(12\*t) + 2)\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2) **Функция зависимости скорости точки по координате Х от времени:**

Vx(t) = -(-4.8\*sin(12\*t)\*cos(12\*t) + 1.8)\*(sin(12\*t) + 2)\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2) + 12\*cos(12\*t)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2)  
 **Функция зависимости скорости точки по координате У от времени:**

Vy(t) = (-4.8\*sin(12\*t)\*cos(12\*t) + 1.8)\*(sin(12\*t) + 2)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2) + 12\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2)\*cos(12\*t) **Функция зависимости ускорения точки по координате Х от времени:**

Wx(t) = -24\*(-4.8\*sin(12\*t)\*cos(12\*t) + 1.8)\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2)\*cos(12\*t) + (23.04\*(-sin(12\*t)\*cos(12\*t) + 0.375)\*\*2)\*(-sin(12\*t) - 2)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2) + (-sin(12\*t) - 2)\*(57.6\*sin(12\*t)\*\*2 - 57.6\*cos(12\*t)\*\*2)\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2) - 144\*sin(12\*t)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2) **Функция зависимости ускорения точки по координате У от времени:**

Wy(t) = 24\*(-4.8\*sin(12\*t)\*cos(12\*t) + 1.8)\*cos(12\*t)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2) - 23.04\*(-sin(12\*t)\*cos(12\*t) + 0.375)\*\*2\*(sin(12\*t) + 2)\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2) + (sin(12\*t) + 2)\*(57.6\*sin(12\*t)\*\*2 - 57.6\*cos(12\*t)\*\*2)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2) - 144\*sin(12\*t)\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2) **Функция зависимости ускорения точки от времени:**

W(t) = sqrt((-24\*(-4.8\*sin(12\*t)\*cos(12\*t) + 1.8)\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2)\*cos(12\*t) + (23.04\*(-sin(12\*t)\*cos(12\*t) + 0.375)\*\*2)\*(-sin(12\*t) - 2)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2) + (-sin(12\*t) - 2)\*(57.6\*sin(12\*t)\*\*2 - 57.6\*cos(12\*t)\*\*2)\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2) - 144\*sin(12\*t)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2))\*\*2 + (24\*(-4.8\*sin(12\*t)\*cos(12\*t) + 1.8)\*cos(12\*t)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2) - 23.04\*(-sin(12\*t)\*cos(12\*t) + 0.375)\*\*2\*(sin(12\*t) + 2)\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2) + (sin(12\*t) + 2)\*(57.6\*sin(12\*t)\*\*2 - 57.6\*cos(12\*t)\*\*2)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2) - 144\*sin(12\*t)\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2))\*\*2)

**Функция зависимости тангенциального ускорения точки от времени:**

Wtan = ((-(-4.8\*sin(12\*t)\*cos(12\*t) + 1.8)\*(sin(12\*t) + 2)\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2) + 12\*cos(12\*t)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2))\*(-48\*(-4.8\*sin(12\*t)\*cos(12\*t) + 1.8)\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2)\*cos(12\*t) + 2\*(23.04\*(-sin(12\*t)\*cos(12\*t) + 0.375)\*\*2)\*(-sin(12\*t) - 2)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2) + 2\*(-sin(12\*t) - 2)\*(57.6\*sin(12\*t)\*\*2 - 57.6\*cos(12\*t)\*\*2)\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2) - 288\*sin(12\*t)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2))/2 + ((-4.8\*sin(12\*t)\*cos(12\*t) + 1.8)\*(sin(12\*t) + 2)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2) + 12\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2)\*cos(12\*t))\*(48\*(-4.8\*sin(12\*t)\*cos(12\*t) + 1.8)\*cos(12\*t)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2) - 2\*23.04\*(-sin(12\*t)\*cos(12\*t) + 0.375)\*\*2\*(sin(12\*t) + 2)\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2) + 2\*(sin(12\*t) + 2)\*(57.6\*sin(12\*t)\*\*2 - 57.6\*cos(12\*t)\*\*2)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2) - 288\*sin(12\*t)\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2))/2)/sqrt((-(-4.8\*sin(12\*t)\*cos(12\*t) + 1.8)\*(sin(12\*t) + 2)\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2) + 12\*cos(12\*t)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2))\*\*2 + ((-4.8\*sin(12\*t)\*cos(12\*t) + 1.8)\*(sin(12\*t) + 2)\*cos(1.8\*t + 0.2\*cos(12\*t)\*\*2) + 12\*sin(1.8\*t + 0.2\*cos(12\*t)\*\*2)\*cos(12\*t))\*\*2)  
 **Функция зависимости радиуса кривизны точки от времени:**

CurveVector(t) = 0.00462962962962963\*(324.0\*(-0.0555555555555556\*(1.8 - 2.4\*sin(12\*t))\*(1.8\*sin(12\*t) + 1)\*sin(1.8\*t + 0.2\*cos(12\*t)) + cos(12\*t)\*cos(1.8\*t + 0.2\*cos(12\*t)))\*\*2 + 324.0\*(0.0555555555555556\*(1.8 - 2.4\*sin(12\*t))\*(1.8\*sin(12\*t) + 1)\*cos(1.8\*t + 0.2\*cos(12\*t)) + sin(1.8\*t + 0.2\*cos(12\*t))\*cos(12\*t))\*\*2)/sqrt(-0.00694444444444444\*((-0.0555555555555556\*(1.8 - 2.4\*sin(12\*t))\*(1.8\*sin(12\*t) + 1)\*sin(1.8\*t + 0.2\*cos(12\*t)) + cos(12\*t)\*cos(1.8\*t + 0.2\*cos(12\*t)))\*(2\*(5.76\*(0.520833333333333 - sin(12\*t))\*\*2)\*(-0.0833333333333333\*sin(12\*t) - 0.0555555555555556)\*cos(1.8\*t + 0.2\*cos(12\*t)) - 4.0\*(1.8- 2.4\*sin(12\*t))\*sin(1.8\*t + 0.2\*cos(12\*t))\*cos(12\*t) - 57.6\*(-0.0833333333333333\*sin(12\*t) - 0.0555555555555556)\*sin(1.8\*t + 0.2\*cos(12\*t))\*cos(12\*t) - 24\*sin(12\*t)\*cos(1.25\*t + 0.2\*cos(12\*t)))/2 + (0.0555555555555556\*(1.8 - 2.4\*sin(12\*t))\*(1.8\*sin(12\*t) + 1)\*cos(1.8\*t + 0.2\*cos(12\*t)) + sin(1.8\*t + 0.2\*cos(12\*t))\*cos(12\*t))\*(-2\*5.76\*(0.520833333333333 - sin(12\*t))\*\*2\*(0.0833333333333333\*sin(12\*t) + 0.0555555555555556)\*sin(1.8\*t + 0.2\*cos(12\*t)) + 4.0\*(1.8 - 2.4\*sin(12\*t))\*cos(12\*t)\*cos(1.8\*t + 0.2\*cos(12\*t)) - 57.6\*(0.0833333333333333\*sin(12\*t) + 0.0555555555555556)\*cos(12\*t)\*cos(1.25\*t + 0.2\*cos(12\*t)) - 24\*sin(12\*t)\*sin(1.8\*t + 0.2\*cos(12\*t)))/2)\*\*2/((-0.0555555555555556\*(1.8 - 2.4\*sin(12\*t))\*(1.8\*sin(12\*t) + 1)\*sin(1.8\*t + 0.2\*cos(12\*t)) + cos(12\*t)\*cos(1.8\*t + 0.2\*cos(12\*t)))\*\*2 + (0.0555555555555556\*(1.25 - 2.4\*sin(12\*t))\*(1.8\*sin(12\*t) + 1)\*cos(1.8\*t + 0.2\*cos(12\*t)) + sin(1.8\*t + 0.2\*cos(12\*t))\*cos(12\*t))\*\*2) + (0.0266666666666667\*(0.520833333333333 - sin(12\*t))\*\*2\*(-1.8\*sin(12\*t) - 1)\*cos(1.8\*t + 0.2\*cos(12\*t)) - 0.166666666666667\*(1.8 - 2.4\*sin(12\*t))\*sin(1.8\*t + 0.2\*cos(12\*t))\*cos(12\*t) - 0.133333333333333\*(-1.8\*sin(12\*t) - 1)\*sin(1.8\*t + 0.2\*cos(12\*t))\*cos(12\*t) - sin(12\*t)\*cos(1.8\*t + 0.2\*cos(12\*t)))\*\*2 + (-0.0266666666666667\*(0.520833333333333 - sin(12\*t))\*\*2\*(1.8\*sin(12\*t) + 1)\*sin(1.8\*t + 0.2\*cos(12\*t)) + 0.166666666666667\*(1.8 - 2.4\*sin(12\*t))\*cos(12\*t)\*cos(1.8\*t + 0.2\*cos(12\*t)) - 0.133333333333333\*(1.8\*sin(12\*t) + 1)\*cos(12\*t)\*cos(1.8\*t + 0.2\*cos(12\*t)) - sin(12\*t)\*sin(1.8\*t + 0.2\*cos(12\*t)))\*\*2)