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

Ханнанов Руслан Маратович

Преподаватель: Волков Евгений Валерьевич

Оценка:

Дата: 13.12.2021

Подпись:

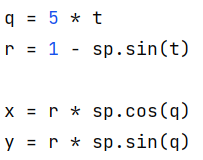
Москва, 2021

**Вариант № «30»**

**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')  
R = 4  
Omega = 1  
  
q = 5 \* t  
r = 1 - sp.sin(t)  
  
*#x = R \* (Omega \* t - sp.sin(Omega \* t))  
#y = R \* (1 - sp.cos(Omega \* t))*x = r \* sp.cos(q)  
y = r \* sp.sin(q)  
  
Vx = sp.diff(x, t)  
Vy = sp.diff(y, t)  
  
Wx = sp.diff(Vx, t)  
Wy = sp.diff(Vy, t)  
  
T = np.linspace(1, 10, 1000)  
  
X = np.zeros\_like(T)  
Y = np.zeros\_like(T)  
  
VX = np.zeros\_like(T)  
VY = np.zeros\_like(T)  
  
WN = np.zeros\_like(T)  
RO = np.zeros\_like(T)  
  
WX = np.zeros\_like(T)  
WY = np.zeros\_like(T)  
  
CIRCLE\_POINT\_X = np.zeros\_like(T)  
CIRCLE\_POINT\_Y = np.zeros\_like(T)  
  
sub\_1 = np.zeros\_like(T)  
sub\_2 = np.zeros\_like(T)  
  
for i in np.arange(len(T)):  
 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])  
  
 sub\_1[i] = (X[i] + VX[i]) \* (Y[i] + VY[i] + WY[i]) - (X[i] + VX[i] + WX[i]) \* (Y[i] + VY[i])  
 sub\_2[i] = (X[i] + VX[i]) \* (X[i] + VX[i] + WX[i]) + (Y[i] + VY[i]) \* (Y[i] + VY[i] + WY[i])  
  
 *# aplha = atan2(x1 \* y2 - x2 \* y1, x1 \* x2 + y1 \* y2) Формула из скалярного и векторного произведений векторов* WN[i] = math.sqrt(WX[i] \*\* 2 + WY[i] \*\* 2) \* math.sin(math.atan2(sub\_1[i], sub\_2[i]))  
 RO[i] = (VX[i] \*\* 2 + VY[i] \*\* 2) / WN[i]  
  
 CIRCLE\_POINT\_X[i] = (2 \* RO[i] \* -VY[i]) / math.sqrt(VY[i]\*\*2 + VX[i]\*\*2)  
 CIRCLE\_POINT\_Y[i] = (2 \* RO[i] \* (VX[i])) / math.sqrt(VY[i]\*\*2 + VX[i]\*\*2)  
  
  
fig = plt.figure()  
ax1 = fig.add\_subplot(1, 1, 1)  
ax1.axis('equal')  
ax1.set(xlim=[-R \* 2, 2 \* R], ylim=[-R \* 2, 2 \* R])  
ax1.plot(X, Y)  
P, = ax1.plot(X[0], Y[0], marker='o')  
  
  
P\_circle, = ax1.plot(CIRCLE\_POINT\_X[0], CIRCLE\_POINT\_Y[0], 'b', marker='o', markersize = 3)  
  
  
Vline, = ax1.plot([X[0], X[0] + VX[0]], [Y[0], Y[0] + VY[0]], 'r')  
  
Wline, = ax1.plot([X[0], X[0] + VX[0] + WX[0]], [Y[0], Y[0] + VY[0] + WY[0]], 'g')  
  
Rline, = ax1.plot([X[0], (2\*X[0] + CIRCLE\_POINT\_X[0]) / 2], [Y[0], (2\*Y[0] + CIRCLE\_POINT\_Y[0]) / 2], 'b')  
  
  
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([-0.02 \* R, 0, -0.02 \* R])  
ArrowY = np.array([0.01 \* R, 0, -0.01 \* R])  
RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(VY[0], VX[0]))  
VArrow, = ax1.plot(RArrowX + X[0] + VX[0], RArrowY + Y[0] + VY[0], 'r')  
  
ArrowX\_2 = np.array([-0.02 \* R, 0, -0.02 \* R])  
ArrowY\_2 = np.array([0.01 \* R, 0, -0.01 \* R])  
RArrowX\_2, RArrowY\_2 = Rot2D(ArrowX\_2, ArrowY\_2, math.atan2(VY[0] + WY[0], VX[0] + WX[0]))  
VArrow\_2, = ax1.plot(RArrowX\_2 + X[0] + VX[0] + WX[0], RArrowY\_2 + Y[0] + VY[0] + WY[0], 'g')  
  
def anima(j): *# анимация движения стрелочки* P.set\_data(X[j], Y[j])  
  
 P\_circle.set\_data(X[j] + CIRCLE\_POINT\_X[j], Y[j] + CIRCLE\_POINT\_Y[j])  
  
 Vline.set\_data([X[j], X[j] + VX[j]], [Y[j], Y[j] + VY[j]])  
  
 Wline.set\_data([X[j], X[j] + VX[j] + WX[j]], [Y[j], Y[j] + VY[j] + WY[j]])  
  
 Rline.set\_data([X[j], (2\*X[j] + CIRCLE\_POINT\_X[j]) / 2], [Y[j], (2\*Y[j] + CIRCLE\_POINT\_Y[j]) / 2])  
  
 RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(VY[j], VX[j]))  
  
 RArrowX\_2, RArrowY\_2 = Rot2D(ArrowX\_2, ArrowY\_2, math.atan2(VY[j] + WY[j], VX[j] + WX[j]))  
  
 VArrow.set\_data(RArrowX + X[j] + VX[j], RArrowY + Y[j] + VY[j])  
  
 VArrow\_2.set\_data(RArrowX\_2 + X[j] + VX[j] + WX[j], RArrowY\_2 + Y[j] + VY[j] + WY[j])  
  
 circle1 = plt.Circle(((2\*X[j] + CIRCLE\_POINT\_X[j]) / 2, (2\*Y[j] + CIRCLE\_POINT\_Y[j]) / 2), RO[j], color='r', fill=False)  
 ax1.add\_artist(circle1)  
  
 return P, Vline, VArrow, VArrow\_2, Wline, P\_circle, circle1, Rline  
  
  
anim = FuncAnimation(fig, anima, frames=1000, interval=50, blit=True)  
plt.show()

**Результат работы программы:**

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

**Вывод программы:  
  
Функция зависимости координаты Х точки от времени:**

x(t) = (1 - sin(t))\*cos(5\*t)  
 **Функция зависимости координаты У точки от времени:**

y(t) = (1 - sin(t))\*sin(5\*t) **Функция зависимости скорости точки по координате Х от времени:**

Vx(t) = -5\*(1 - sin(t))\*sin(5\*t) - cos(t)\*cos(5\*t)  
 **Функция зависимости скорости точки по координате У от времени:**

Vy(t) = 5\*(1 - sin(t))\*cos(5\*t) - sin(5\*t)\*cos(t) **Функция зависимости ускорения точки по координате Х от времени:**

Wx(t) = 5\*(5\*sin(t) - 5)\*cos(5\*t) + sin(t)\*cos(5\*t) + 10\*sin(5\*t)\*cos(t) **Функция зависимости ускорения точки по координате У от времени:**

Wy(t) = -5\*(5 - 5\*sin(t))\*sin(5\*t) + sin(t)\*sin(5\*t) - 10\*cos(t)\*cos(5\*t) **Вывод:**

Благодаря данной лабораторной работе я познакомился с моделированием движения точки с помощью языка программирования Python с помощью следующих модулей: matplotlib – для отрисовки изображения и анимации, numpy – для более удобной работы с массивам, sympy – для символьных вычислений(нахождения производных функций). С помощью вышеперечисленных инструментов, я отрисовал движение точки, заданное в полярных координатах, векторы её скорости, ускорения и радиус кривизны.