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-21

Павлов Иван Дмитриевич

Преподаватель: Беличенко Михаил Валериевич

Оценка:

Дата:

Москва, 2022

**Вариант № 18**

**Задание:**

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

**Закон движения точки:**

r(t) = 1 + 1.5sin(12t)

φ(t) = 1.2t + 0.2cos(12t)

**Текст программы**

import numpy as np

import sympy as sp

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

class ModelOfPoint():

def \_\_init\_\_(self):

self.T = np.linspace(0, 10, 1001)

self.t = sp.Symbol('t')

self.pol\_ro = 1 + 1.5 \* sp.sin(12 \* self.t)

self.pol\_phi = 1.2 \* self.t + 0.2 \* sp.cos(12 \* self.t)

x = self.pol\_ro \* sp.cos(self.pol\_phi)

y = self.pol\_ro \* sp.sin(self.pol\_phi)

Vx = sp.diff(x, self.t)

Vy = sp.diff(y, self.t)

Wx = sp.diff(Vx, self.t)

Wy = sp.diff(Vy, self.t)

rox = (-Vy \* (Vx\*\*2 + Vy\*\*2)) / sp.sqrt((Vx \* Wy - Vy \* Wx) \*\* 2)

roy = (Vx \* (Vx\*\*2 + Vy\*\*2)) / sp.sqrt((Vx \* Wy - Vy \* Wx) \*\* 2)

self.x = self.F(x)(self.T)

self.y = self.F(y)(self.T)

self.Vx = self.F(Vx)(self.T) / 10

self.Vy = self.F(Vy)(self.T) / 10

self.Wx = self.F(Wx)(self.T) / 100

self.Wy = self.F(Wy)(self.T) / 100

self.Rox = self.F(rox)(self.T) / 100

self.Roy = self.F(roy)(self.T) / 100

self.phi = np.arctan2(self.Vy, self.Vx)

self.teta = np.arctan2(self.Wy, self.Wx)

def F(self, x):

return sp.lambdify(self.t, x, 'numpy')

def Rot2D(self, x, y, phi):

self.X = x \* np.cos(phi) - y \* np.sin(phi)

self.Y = x \* np.sin(phi) + y \* np.cos(phi)

return (self.X, self.Y)

def ArrowPattern(self):

a = 0.1

b = 0.03

X\_pattern = np.array([-1.5 \* a, -a, 0, -a, -1.5 \* a])

Y\_pattern = np.array([2 \* b, b, 0, -b, -2 \* b])

return (X\_pattern, Y\_pattern)

def Kadr(self, i) -> None:

self.P.set\_data(self.x[i], self.y[i])

self.Vline.set\_data([self.x[i], self.x[i] + self.Vx[i]],

[self.y[i], self.y[i] + self.Vy[i]])

self.Wline.set\_data([self.x[i], self.x[i] + self.Wx[i]],

[self.y[i], self.y[i] + self.Wy[i]])

self.RVX = self.Rot2D(self.ArrowPattern()[0],

self.ArrowPattern()[1],

self.phi[i])[0]

self.RVY = self.Rot2D(self.ArrowPattern()[0],

self.ArrowPattern()[1],

self.phi[i])[1]

self.RWX = self.Rot2D(self.ArrowPattern()[0],

self.ArrowPattern()[1],

self.teta[i])[0]

self.RWY = self.Rot2D(self.ArrowPattern()[0],

self.ArrowPattern()[1],

self.teta[i])[1]

self.Varrow.set\_data(self.x[i] + self.Vx[i] + self.RVX,

self.y[i] + self.Vy[i] + self.RVY)

self.Warrow.set\_data(self.x[i] + self.Wx[i] + self.RWX,

self.y[i] + self.Wy[i] + self.RWY)

self.roVec.set\_data([self.x[i], self.Rox[i]],

[self.y[i], self.Roy[i]])

def forward(self) -> None:

fig = plt.figure(figsize=[16, 9])

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

ax.axis('equal')

ax.plot(self.x, self.y)

ax.set(xlim=[-5, 5], ylim=[-5, 5])

self.RVX = self.Rot2D(self.ArrowPattern()[0],

self.ArrowPattern()[1],

self.phi[0])[0]

self.RVY = self.Rot2D(self.ArrowPattern()[0],

self.ArrowPattern()[1],

self.phi[0])[1]

self.RWX = self.Rot2D(self.ArrowPattern()[0],

self.ArrowPattern()[1],

self.teta[0])[0]

self.RWY = self.Rot2D(self.ArrowPattern()[0],

self.ArrowPattern()[1],

self.teta[0])[1]

self.Varrow = ax.plot(self.x[0] + self.Vx[0] + self.RVX,

self.y[0] + self.Vy[0] + self.RVY,

color=[0, 1, 0])[0]

self.Vline = ax.plot([self.x[0], self.x[0] + self.Vx[0]],

[self.y[0], self.y[0] + self.Vy[0]],

color=[0, 1, 0])[0]

self.Warrow = ax.plot(self.x[0] + self.Vx[0] + self.RWX,

self.y[0] + self.Vy[0] + self.RWY,

color='m')[0]

self.Wline = ax.plot([self.x[0], self.x[0] + self.Wx[0]],

[self.y[0], self.y[0] + self.Wy[0]],

color='m')[0]

self.roVec = ax.plot([self.x[0], self.Rox[0]],

[self.y[0], self.Roy[0]],

color=[1, 1, 0])[0]

self.P = ax.plot(self.x[0], self.y[0], 'o', color=[1, 0, 0])[0]

multik = FuncAnimation(fig,

self.Kadr,

interval=(self.T[1] - self.T[0]) \* 1000,

frames=len(self.T))

plt.show()

model = ModelOfPoint()

if \_\_name\_\_ == "\_\_main\_\_":

model.forward()

**Результат работы программы:**

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