ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ ТОЧКИ»**

**ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА И ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ»**

**ВАРИАНТ ЗАДАНИЯ №4**

Выполнил(а) студент группы М8О-208Б-21

Белоусов Егор Леонидович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Зав. каф. 802, Бардин Б.С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

*Задание:* построить заданную траекторию, запустить анимацию движения точки, построить стрелки векторов скорости и ускорения, а также радиуса кривизны.

**Закон движения точки:**

**Текст программы**

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.lines import Line2D as l2

from matplotlib.animation import FuncAnimation

import sympy as sp

import math

def rotate\_2d(X, Y, Alpha):

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

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

return RX, RY

v\_scl = 0.1

a\_scl = v\_scl\*\*2

ar\_scl = 0.025

frame\_num = 1000

frame\_frequency = 10

t\_end = 12 \* math.pi / 6

t = sp.Symbol('t')

r = 1 + sp.sin(8 \* t)

phi = t + 0.5 \* sp.sin(8 \* t)

x = r \* sp.sin(phi)

y = r \* sp.cos(phi)

v\_x = sp.diff(x, t)

v\_y = sp.diff(y, t)

a\_x = sp.diff(v\_x, t)

a\_y = sp.diff(v\_y, t)

T = np.linspace(0, t\_end, frame\_num)

X = np.zeros\_like(T)

Y = np.zeros\_like(T)

V\_X = np.zeros\_like(T)

V\_Y = np.zeros\_like(T)

A\_X = np.zeros\_like(T)

A\_Y = np.zeros\_like(T)

RC\_X = np.zeros\_like(T)

RC\_Y = np.zeros\_like(T)

RC\_R = 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])

V\_X[i] = sp.Subs(v\_x, t, T[i])

V\_Y[i] = sp.Subs(v\_y, t, T[i])

A\_X[i] = sp.Subs(a\_x, t, T[i])

A\_Y[i] = sp.Subs(a\_y, t, T[i])

RC\_X[i] = (V\_X[i]\*\*2 + V\_Y[i]\*\*2) / (A\_X[i] \* V\_Y[i] - A\_Y[i] \* V\_X[i])

RC\_Y[i] = RC\_X[i] \* V\_Y[i]

RC\_X[i] \*= V\_X[i]

RC\_R[i] = math.sqrt(RC\_X[i]\*\*2 + RC\_Y[i]\*\*2)

V\_X \*= v\_scl

V\_Y \*= v\_scl

A\_X \*= a\_scl

A\_Y \*= a\_scl

fig = plt.figure()

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

ax1.axis('equal')

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

P, = ax1.plot(X[0], Y[0], color='red', marker='\*')

V, = ax1.plot([X[0], X[0] + V\_X[0]],

[Y[0], Y[0] + V\_Y[0]], color='red')

R, = ax1.plot([0, X[0]], [0, Y[0]], color='black')

A, = ax1.plot([X[0], X[0] + A\_X[0]],

[Y[0], Y[0] + A\_Y[0]], color='green')

Alpha = np.linspace(0, math.pi \* 2, 100)

RC, = ax1.plot([X[0], X[0] + RC\_Y[0]],

[Y[0], Y[0] - RC\_X[0]], color='blue')

ArrowX = np.array([-2 \* ar\_scl, 0, -2 \* ar\_scl])

ArrowY = np.array([ar\_scl, 0, -ar\_scl])

VArrowX, VArrowY = rotate\_2d(ArrowX, ArrowY, math.atan2(V\_Y[0], V\_X[0]))

VArrow, = ax1.plot(VArrowX + X[0] + V\_X[0],

VArrowY + Y[0] + V\_Y[0], color='red')

RArrowX, RArrowY = rotate\_2d(ArrowX, ArrowY, math.atan2(Y[0], X[0]))

RArrow, = ax1.plot(RArrowX + X[0], RArrowY + Y[0], color='black')

AArrowX, AArrowY = rotate\_2d(ArrowX, ArrowY, math.atan2(A\_Y[0], A\_X[0]))

AArrow, = ax1.plot(AArrowX + X[0] + A\_X[0],

AArrowY + Y[0] + A\_Y[0], color='green')

RCArrowX, RCArrowY = rotate\_2d(ArrowX, ArrowY, math.atan2(-RC\_X[0], RC\_Y[0]))

RCArrow, = ax1.plot(RCArrowX + X[0] + RC\_Y[0],

RCArrowY + Y[0] - RC\_X[0], color='blue')

ax1.plot(X, Y, 'grey')

def anim(i):

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

V.set\_data([X[i], X[i] + V\_X[i]],

[Y[i], Y[i] + V\_Y[i]])

R.set\_data([0, X[i]], [0, Y[i]])

A.set\_data([X[i], X[i] + A\_X[i]],

[Y[i], Y[i] + A\_Y[i]])

VArrowX, VArrowY = rotate\_2d(ArrowX, ArrowY, math.atan2(V\_Y[i], V\_X[i]))

VArrow.set\_data(VArrowX + X[i] + V\_X[i],

VArrowY + Y[i] + V\_Y[i])

RArrowX, RArrowY = rotate\_2d(ArrowX, ArrowY, math.atan2(Y[i], X[i]))

RArrow.set\_data(RArrowX + X[i], RArrowY + Y[i])

AArrowX, AArrowY = rotate\_2d(ArrowX, ArrowY, math.atan2(A\_Y[i], A\_X[i]))

AArrow.set\_data(AArrowX + X[i] + A\_X[i],

AArrowY + Y[i] + A\_Y[i])

RC.set\_data([X[i], X[i] + RC\_Y[i]],

[Y[i], Y[i] - RC\_X[i]])

RCArrowX, RCArrowY = rotate\_2d(ArrowX, ArrowY, math.atan2(-RC\_X[i], RC\_Y[i]))

RCArrow.set\_data(RCArrowX + X[i] + RC\_Y[i],

RCArrowY + Y[i] - RC\_X[i])

return P, V, VArrow, R, RArrow, A, AArrow, RC, RCArrow

anim1 = FuncAnimation(fig, anim, frames=frame\_num, interval=frame\_num / frame\_frequency, blit=True)

custom\_lines = [l2([0], [0], color='grey'),

l2([0], [0], color='black'),

l2([0], [0], color='red'),

l2([0], [0], color='green'),

l2([0], [0], color='blue')]

ax1.legend(custom\_lines, ['Траектория', 'Радиус-вектор', 'Скорость', 'Ускорение', 'Радиус кривизны'], loc='lower left')

plt.show()

**Результат работы программы:**

|  |  |
| --- | --- |
| **Рис.1** | **Рис.2** |
| **Рис.3** | **Рис.4** |