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 № 7**

Выполнил(а) студент группы М8О-203Б-22

Ибрагимов Роман Рифхатович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

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

подпись, дата

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

Москва, 2023

*Задание:* построить заданную траекторию, запустить анимацию движения точки, построить стрелки радиус-вектора, вектора скорости, вектора ускорения и радиуса кривизны.

 Код программы

import math

import sympy as s

import matplotlib.pyplot as plot

import numpy as np

from matplotlib.animation import FuncAnimation

def rotation2D(x, y, angle):

Rot\_x = x \* np.cos(angle) - y \* np.sin(angle)

Rot\_y = x \* np.sin(angle) + y \* np.cos(angle)

return Rot\_x, Rot\_y

def Vect\_arrow(VecX, VecY, X, Y):

a = 0.3

b = 0.2

arrow\_x = np.array([-a, 0, -a])

arrow\_y = np.array([b, 0, -b])

phi = math.atan2(VecY, VecX)

RotX, RotY = rotation2D(arrow\_x, arrow\_y, phi)

arrow\_x = RotX + X + VecX

arrow\_y = RotY + Y + VecY

return arrow\_x, arrow\_y

def anim(i):

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

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

RArrow.set\_data(Vect\_arrow(X[i], Y[i], 0, 0))

VVector.set\_data([X[i], X[i] + X\_velocity[i]], [Y[i], Y[i] + Y\_velocity[i]])

VArrow.set\_data(Vect\_arrow(X\_velocity[i], Y\_velocity[i], X[i], Y[i]))

AVector.set\_data([X[i], X[i] + X\_acceleration[i]], [Y[i], Y[i] + Y\_acceleration[i]])

AArrow.set\_data(Vect\_arrow(X\_acceleration[i], Y\_acceleration[i], X[i], Y[i]))

RCVector.set\_data([X[i], X[i] + X\_rcurvature[i]], [Y[i], Y[i] + Y\_rcurvature[i]])

RCArrow.set\_data(Vect\_arrow(X\_rcurvature[i], Y\_rcurvature[i], X[i], Y[i]))

return

t = s.Symbol('t')

r = 2 + s.cos(6 \* t)

phi = t + 1.2 \* s.cos(6 \* t)

x = r \* s.cos(phi)

y = r \* s.sin(phi)

x\_velocity = s.diff(x)

y\_velocity = s.diff(y)

x\_acceleration = s.diff(x\_velocity)

y\_acceleration = s.diff(y\_velocity)

Velocity = s.sqrt(x\_velocity \*\* 2 + y\_velocity \*\* 2)

Acceleration = s.sqrt(x\_acceleration \*\* 2 + y\_acceleration \*\* 2)

Acceleration\_t = s.diff(Velocity)

Acceleration\_n = s.sqrt(Acceleration \*\* 2 - Acceleration\_t \*\* 2)

RСurvature = (Velocity \*\* 2) / Acceleration\_n

step = 2000

T = np.linspace(0, 10, step)

X = np.zeros\_like(T)

Y = np.zeros\_like(T)

X\_velocity = np.zeros\_like(T)

Y\_velocity = np.zeros\_like(T)

X\_acceleration = np.zeros\_like(T)

Y\_acceleration = np.zeros\_like(T)

X\_rcurvature = np.zeros\_like(T)

Y\_rcurvature = np.zeros\_like(T)

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

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

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

X\_velocity[i] = s.Subs(x\_velocity, t, T[i])

Y\_velocity[i] = s.Subs(y\_velocity, t, T[i])

X\_acceleration[i] = s.Subs(x\_acceleration, t, T[i])

Y\_acceleration[i] = s.Subs(y\_acceleration, t, T[i])

Veloctity\_angle = math.atan2(Y\_velocity[i], X\_velocity[i])

Acceleration\_angle = math.atan2(Y\_acceleration[i], X\_acceleration[i])

RСurvature\_angle = Veloctity\_angle - math.pi / 2 if Veloctity\_angle - Acceleration\_angle > 0 else Veloctity\_angle + math.pi / 2

X\_rcurvature[i] = RСurvature.subs(t, T[i]) \* math.cos(RСurvature\_angle)

Y\_rcurvature[i] = RСurvature.subs(t, T[i]) \* math.sin(RСurvature\_angle)

fgr = plot.figure()

grf = fgr.add\_subplot(1, 1, 1)

grf.axis('equal')

grf.set(xlim=[-10, 10], ylim=[-10, 10])

grf.plot(X, Y)

Pnt = grf.plot(X[0], Y[0], marker='o')[0]

X\_RArrow, Y\_RArrow = Vect\_arrow(X[0], Y[0], 0, 0)

RArrow = grf.plot(X\_RArrow, Y\_RArrow, 'black')[0]

RVector = grf.plot([0, X[0]], [0, Y[0]], 'black')[0]

X\_VArrow, Y\_VArrow = Vect\_arrow(X\_velocity[0], Y\_velocity[0], X[0], Y[0])

VArrow = grf.plot(X\_VArrow, Y\_VArrow, 'r')[0]

VVector = grf.plot([X[0], X[0] + X\_velocity[0]], [Y[0], Y[0] + Y\_velocity[0]], 'r')[0]

X\_AArrow, Y\_AArrow = Vect\_arrow(X\_acceleration[0], Y\_acceleration[0], X[0], Y[0])

AArrow = grf.plot(X\_AArrow, Y\_AArrow, 'g')[0]

AVector = grf.plot([X[0], X[0] + X\_acceleration[0]], [Y[0], Y[0] + Y\_acceleration[0]], 'g')[0]

X\_RCArrow, Y\_RCArrow = Vect\_arrow(X\_rcurvature[0], Y\_rcurvature[0], X[0], Y[0])

RCArrow = grf.plot(X\_RCArrow, Y\_RCArrow, 'y')[0]

RCVector = grf.plot([X[0], X[0] + X\_rcurvature[0]], [Y[0], Y[0] + Y\_rcurvature[0]], 'y')[0]

an = FuncAnimation(fgr, anim, frames=step, interval=20)

plot.show()

Результат выполнения программы





