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 №17**

Выполнил(а) студент группы М8О-212Б-22

Мамонтов Егор Олегович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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.

*Задание варианта:*

r(t) = 2 + 0.5 \* sin(12\*t);

φ(t) = 1.2\*t + 0.2 \* cos(12\*t);

*Программа:*

import numpy as np

import math

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

import sympy as sp

class ArrowedVector:

    def get\_x(S):

        return np.concatenate([[S.x0, S.x1], S.x\_arrow])

    def get\_y(S):

        return np.concatenate([[S.y0, S.y1], S.y\_arrow])

    def rotate\_2d(S, X, Y, alph):

        R\_X = X \* np.cos(alph) - Y \* np.sin(alph)

        R\_Y = X \* np.sin(alph) + Y \* np.cos(alph)

        return R\_X, R\_Y

    def set(S, A, B):

        S.x0, S.y0 = A[0], A[1]

        S.x1, S.y1 = B[0], B[1]

        S.x\_arrow = np.array([-0.05, 0, -0.05])

        S.y\_arrow = np.array([0.025, 0, -0.025])

        S.x\_arrow, S.y\_arrow = S.rotate\_2d(S.x\_arrow, S.y\_arrow, math.atan2(S.y1 - S.y0, S.x1 - S.x0))

        if A == B:

            S.x\_arrow, S.y\_arrow = np.zeros(3), np.zeros(3)

        S.x\_arrow += B[0]

        S.y\_arrow += B[1]

    def \_\_init\_\_(S, A, B) -> None:

        S.x0, S.y0 = A[0], A[1]

        S.x1, S.y1 = B[0], B[1]

        S.x\_arrow = np.array([-0.05, 0, 0.05])

        S.y\_arrow = np.array([0.025, 0, -0.25])

        S.x\_arrow, S.y\_arrow = S.rotate\_2d(S.x\_arrow, S.y\_arrow, math.atan2(S.y1 - S.y0, S.x1 - S.x0))

        if A == B:

            S.x\_arrow, S.y\_arrow = np.zeros(3), np.zeros(3)

        S.x\_arrow += B[0]

        S.y\_arrow += B[1]

t = sp.Symbol("t")

r = 2 + 0.5 \* sp.sin(12 \* t)

phi = 1.2 \* t + 0.2 \* sp.cos(12 \* t)

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)

T = np.linspace(0, 10, 1000)

X, Y = np.zeros\_like(T), np.zeros\_like(T)

Vx, Vy, Wx, Wy = np.zeros\_like(T), np.zeros\_like(T), np.zeros\_like(T), np.zeros\_like(T)

V = ArrowedVector([0, 0], [0, 0])

W = ArrowedVector([0, 0], [0, 0])

R0 = ArrowedVector([0, 0], [0, 0])

for i, time in enumerate(T):

    X[i] = sp.Subs(x, t, time)

    Y[i] = sp.Subs(y, t, time)

    Wx[i] = sp.Subs(wx, t, time)

    Wy[i] = sp.Subs(wy, t, time)

    Vx[i] = sp.Subs(vx, t, time)

    Vy[i] = sp.Subs(vy, t, time)

fig, ax1 = plt.subplots(1, 1)

ax1.axis('equal')

ax1.plot(X, Y)

ax1.set(xlim = [-5, 5], ylim = [-7, 7])

P, = ax1.plot(X[0], Y[0], marker = 'o')

anim\_R0, = ax1.plot(R0.get\_x(), R0.get\_y())

anim\_V, = ax1.plot(V.get\_x(), V.get\_y())

anim\_W, = ax1.plot(W.get\_x(), W.get\_y())

R0.set([0, 0], [Vx[0], Vy[0]])

anim\_R0.set\_data(R0.get\_x(), R0.get\_y())

def animate\_frame(i):

    P.set\_data(X[i], Y[i])

    R0.set([0, 0], [X[i], Y[i]])

    anim\_R0.set\_data(R0.get\_x(), R0.get\_y())

    V.set([X[i], Y[i]], [X[i] + Vx[i], Y[i] + Vy[i]])

    anim\_V.set\_data(V.get\_x(), V.get\_y())

    W.set([X[i], Y[i]], [X[i] + Wx[i], Y[i] + Wy[i]])

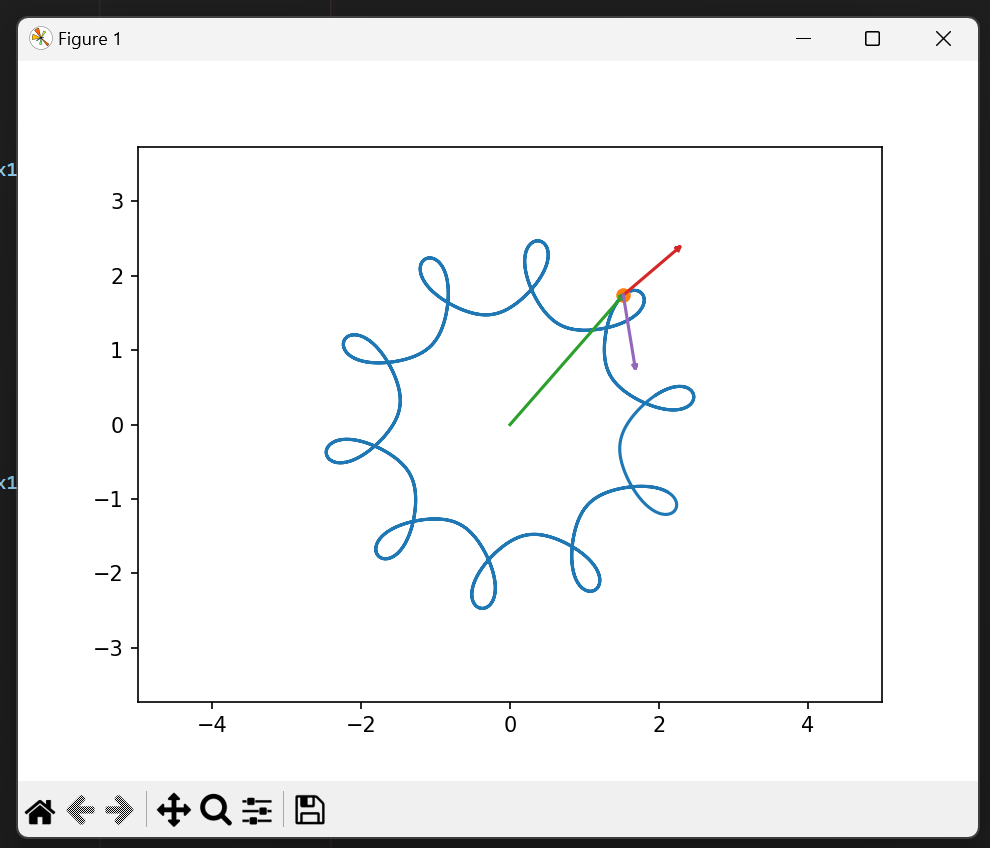
    anim\_W.set\_data(W.get\_x(), W.get\_y())

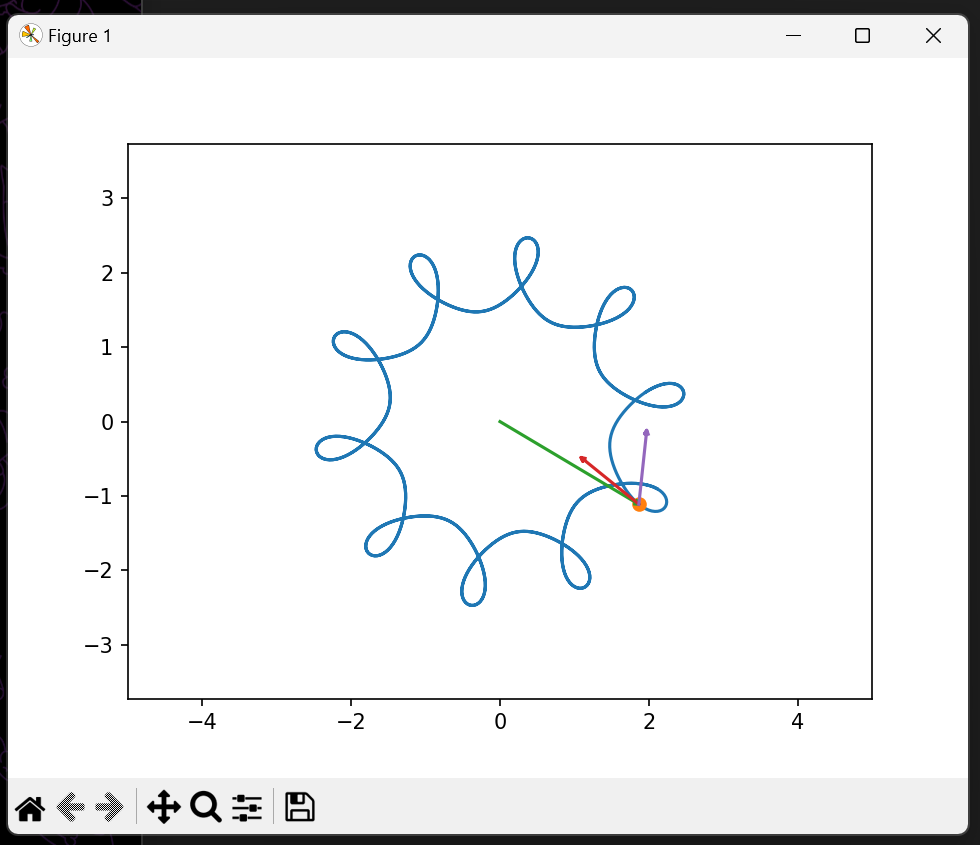
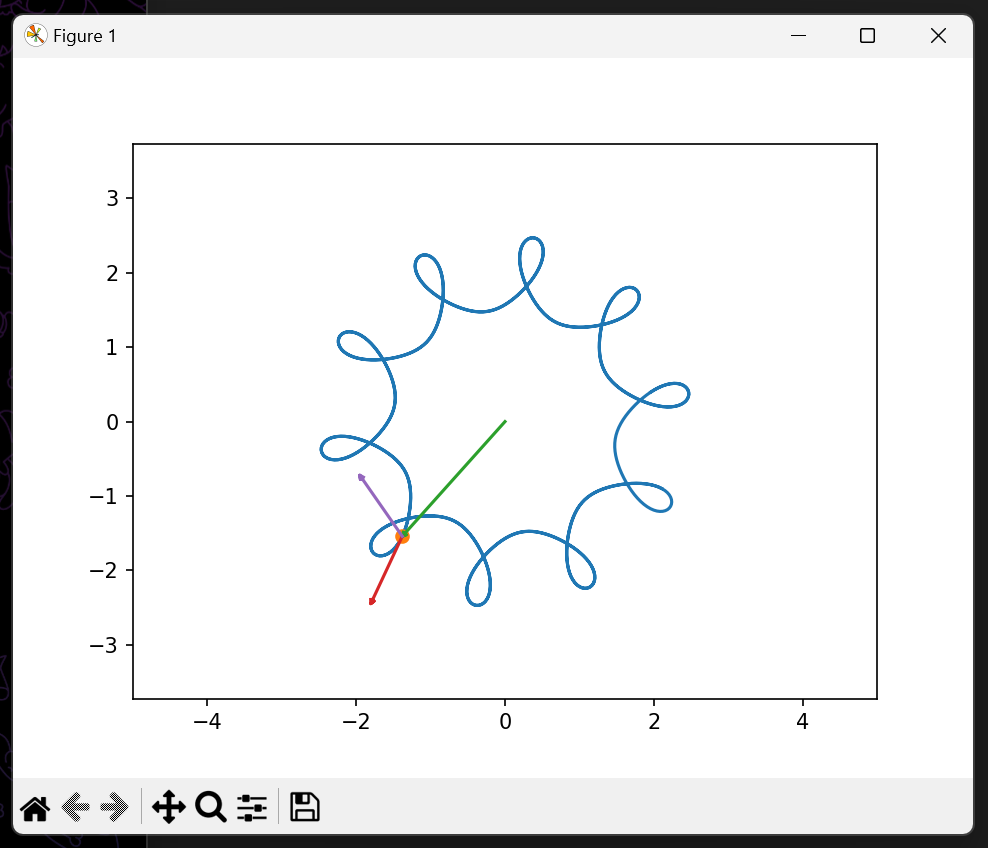
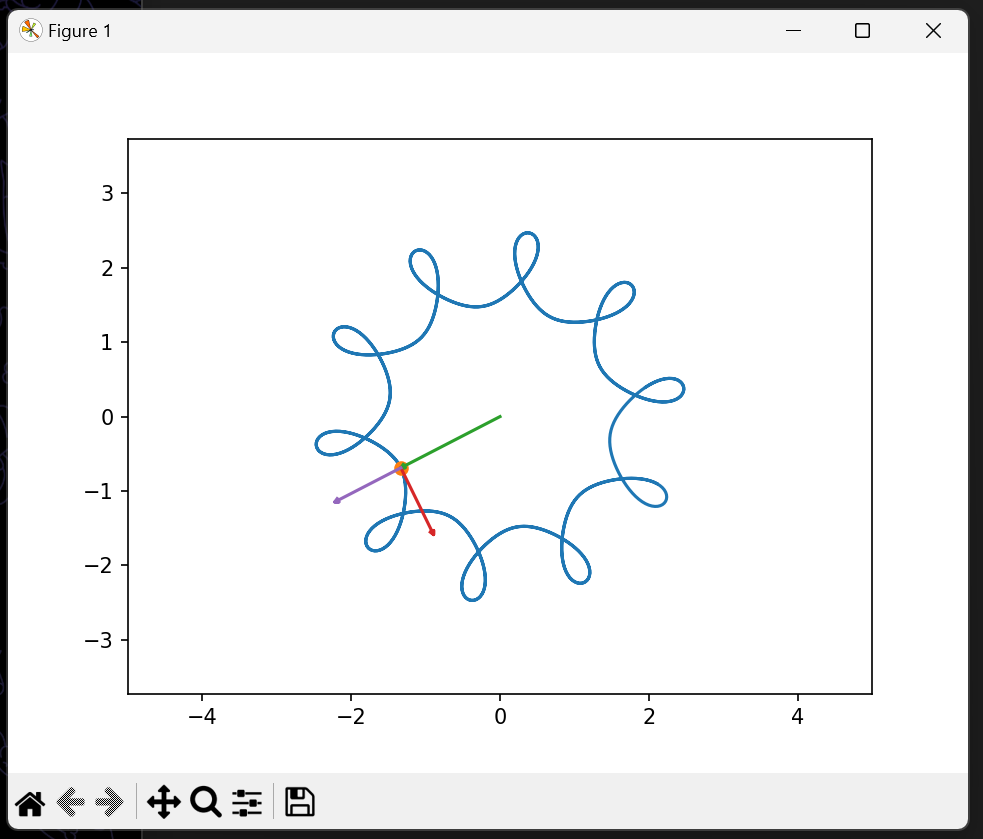
    return P, anim\_R0, anim\_V, anim\_W

animation = FuncAnimation(fig, animate\_frame, frames = 1000, interval = 2, repeat = True)

plt.show()

*Результат работы программы:*





*Вывод:*

Программирование на Python с применением научных библиотек облегчает моделирование и визуализацию разнообразных физических явлений. Создание анимации движения материальной точки может быть полезным инструментом для обучения и постижения физических законов, поскольку это обеспечивает наглядное отображение их функционирования. Использование библиотеки numpy для вычислений и matplotlib для визуализации позволяет создавать динамические анимации, демонстрирующие физические процессы. Визуализация движения материальной точки эффективно представляет изменения ее координат, скорости и ускорения в течение времени.