ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ ТОЧКИ»**

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

**ВАРИАНТ ЗАДАНИЯ №25**

Выполнил(а) студент группы М8О-201Б-22

Чибугаев И.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Волков Е.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

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

Москва, 2024

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

*Вариант:* r(t) = cos(3t), ф(t) = t

Код:

import math

import sympy as s

import matplotlib.pyplot as plot

import numpy as np

from matplotlib.animation import FuncAnimation

t = s.Symbol('t')

x = s.cos(3 \* t) \* s.cos(t)

y = s.cos(3 \* t) \* s.sin(t)

Vx = s.diff(x, t)

Vy = s.diff(y, t)

ax = s.diff(Vx, t)

ay = s.diff(Vy, t)

rx = -Vy / s.sqrt(Vx \* Vx + Vy \* Vy)

ry = Vx / s.sqrt(Vx \* Vx + Vy \* Vy)

step = 1000

T = np.linspace(0, 2 \* math.pi, step)

X = np.zeros\_like(T)

Y = np.zeros\_like(T)

VX = np.zeros\_like(T)

VY = np.zeros\_like(T)

AX = np.zeros\_like(T)

AY = np.zeros\_like(T)

RX = np.zeros\_like(T)

RY = np.zeros\_like(T)

for i in range(len(T)):

    X[i] = x.subs(t, T[i])

    Y[i] = y.subs(t, T[i])

    VX[i] = 0.5 \* Vx.subs(t, T[i])

    VY[i] = 0.5 \* Vy.subs(t, T[i])

    AX[i] = 0.2 \* ax.subs(t, T[i])

    AY[i] = 0.2 \* ay.subs(t, T[i])

    RX[i] = rx.subs(t, T[i])

    RY[i] = ry.subs(t, T[i])

fgr = plot.figure()

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

grf.axis('equal')

grf.set(xlim=[-6, 6], ylim=[-6, 6])

grf.plot(X, Y)

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

Vpl = grf.plot([X[0], X[0] + VX[0]], [Y[0], Y[0] + VY[0]], 'r')[0]

Apl = grf.plot([X[0], X[0] + AX[0]], [Y[0], Y[0] + AY[0]], 'g')[0]

Rpl = grf.plot([X[0], X[0] + RX[0]], [Y[0], Y[0] + RY[0]], 'b')[0]

R\_vec = grf.plot([0, X[0]], [0, Y[0]], 'm')[0]  *# Радиус-вектор*

def vect\_arrow(vec\_x, vec\_y, \_x, \_y):

    a = 0.15

    b = 0.1

    arr\_x = np.array([-a, 0, -a])

    arr\_y = np.array([b, 0, -b])

    phi = math.atan2(vec\_y, vec\_x)

    rot\_x = arr\_x \* np.cos(phi) - arr\_y \* np.sin(phi)

    rot\_y = arr\_x \* np.sin(phi) + arr\_y \* np.cos(phi)

    arr\_x = rot\_x + \_x + vec\_x

    arr\_y = rot\_y + \_y + vec\_y

    return arr\_x, arr\_y

ArVX, ArVY = vect\_arrow(VX[0], VY[0], X[0], Y[0])  *# velocity*

V\_arr = grf.plot(ArVX, ArVY, 'r')[0]

ArAX, ArAY = vect\_arrow(AX[0], AY[0], X[0], Y[0])  *# acceleration*

A\_arr = grf.plot(ArAX, ArAY, 'g')[0]

ArRX, ArRY = vect\_arrow(RX[0], RY[0], X[0], Y[0])  *# Радиус кривизны*

R\_arr = grf.plot(ArRX, ArRY, 'b')[0]

def anim(j):

    global ArVX, ArVY, ArAX, ArAY, ArRX, ArRY

    Pnt.set\_data([X[j]], [Y[j]])

    Vpl.set\_data([X[j], X[j] + VX[j]], [Y[j], Y[j] + VY[j]])

    ArVX, ArVY = vect\_arrow(VX[j], VY[j], X[j], Y[j])

    V\_arr.set\_data(ArVX, ArVY)

    Apl.set\_data([X[j], X[j] + AX[j]], [Y[j], Y[j] + AY[j]])

    ArAX, ArAY = vect\_arrow(AX[j], AY[j], X[j], Y[j])

    A\_arr.set\_data(ArAX, ArAY)

    Rpl.set\_data([X[j], X[j] + RX[j]], [Y[j], Y[j] + RY[j]])

    ArRX, ArRY = vect\_arrow(RX[j], RY[j], X[j], Y[j])

    R\_arr.set\_data(ArRX, ArRY)

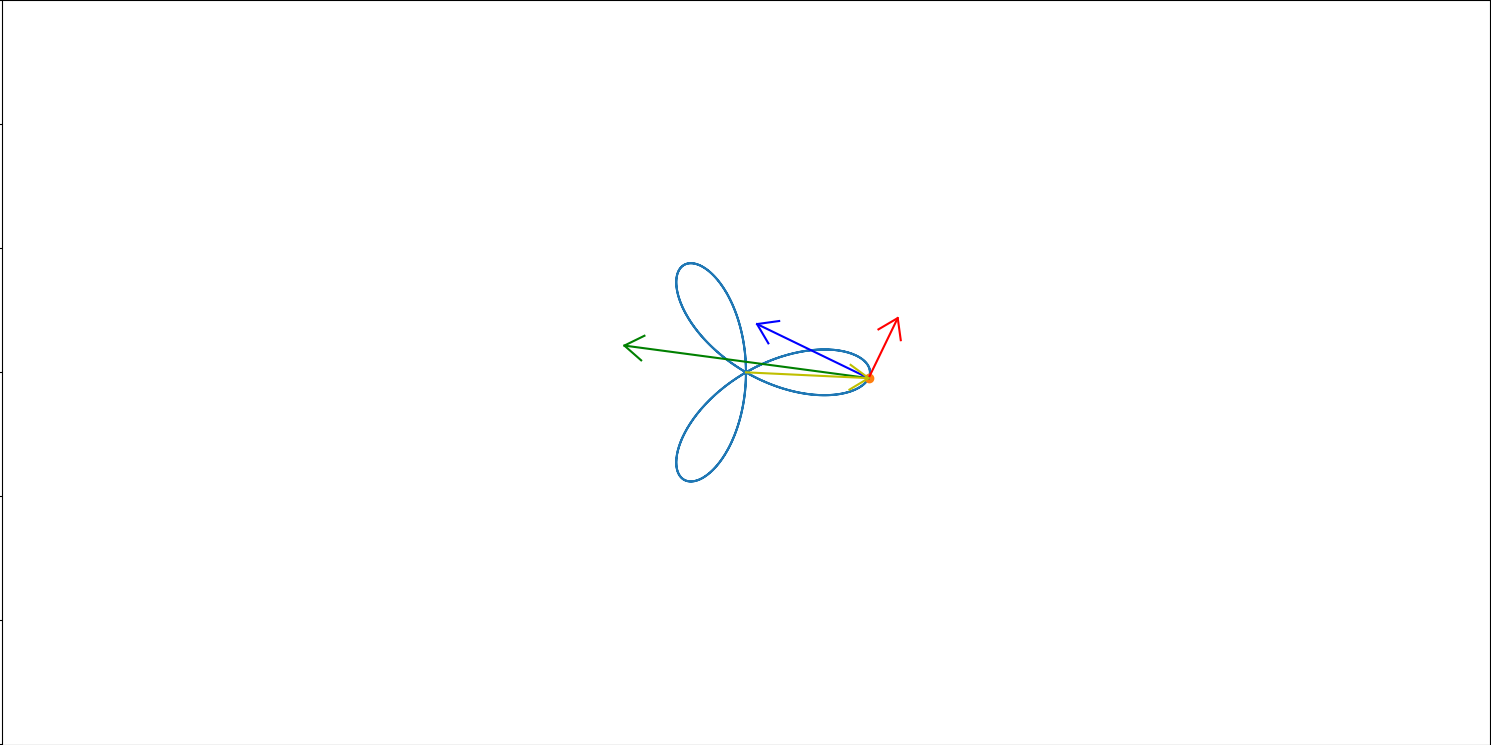
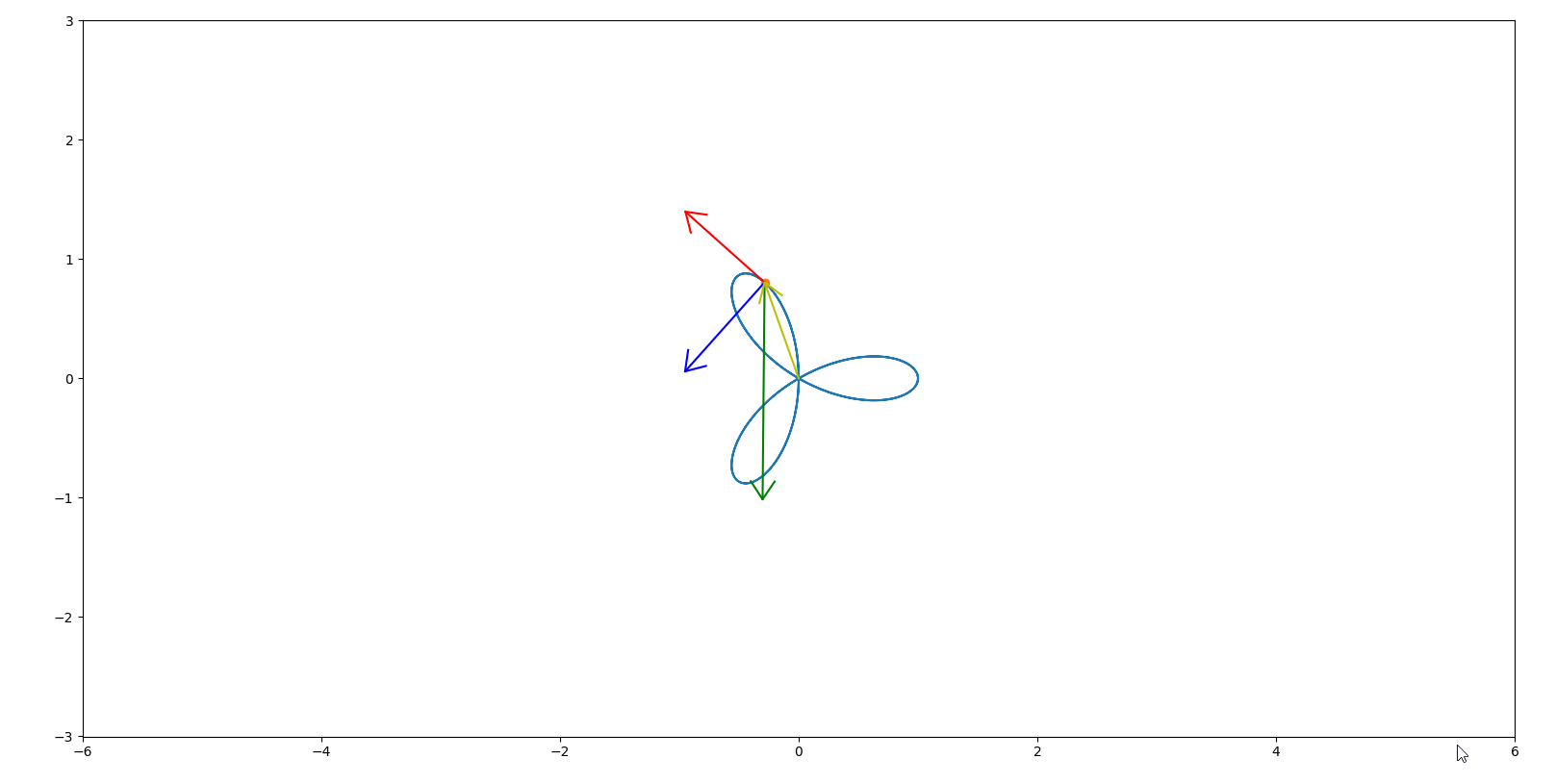
    R\_vec.set\_data([0, X[j]], [0, Y[j]])  *# Обновление радиус-вектора*

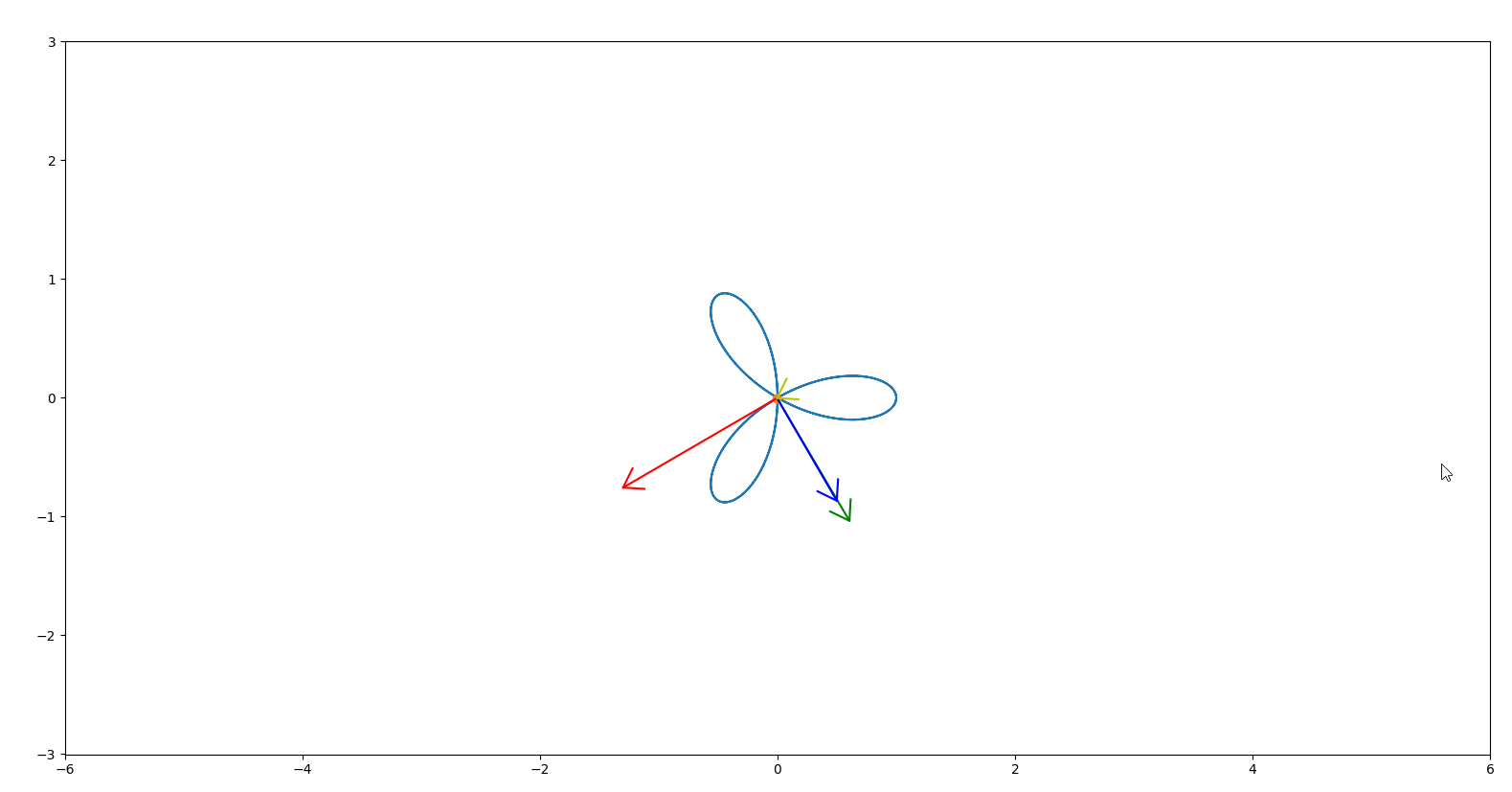
    return [Pnt, Vpl, V\_arr, Apl, A\_arr, Rpl, R\_arr, R\_vec]

an = FuncAnimation(fgr, anim, frames=step, interval=1)

plot.show()

Скриншот выполняющейся программы:



*Вывод:* Программирование на Python с применением библиотек для научных вычислений обеспечивает удобное моделирование и визуализацию разнообразных физических явлений. Создание анимации движения материальной точки становится полезным инструментом для обучения и постижения физических законов, поскольку она позволяет наглядно демонстрировать их действие. Использование библиотеки numpy для вычислений и matplotlib для визуализации открывает возможность создавать динамические анимации, наглядно отражающие физические процессы. Визуализация движения материальной точки эффективно демонстрирует изменение ее координат, скорости и ускорения в течение времени.