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 № 19**

Выполнил студент группы М8О-210Б-22

Сисенов Марат Мийрамович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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.

Вариант: r(t) = 1 + 1.5sin(12t) Phi(t) = 1.25t + 0.2cos(*12t*)

**Текст программы**

import matplotlib.pyplot as plt

import numpy as np

import math

from matplotlib.animation import FuncAnimation

import sympy as sp

def Rot2D(X, Y, Alpha):

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

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

return RX, RY

# time

t = sp.Symbol('t')

# 1000 dots in (1, 15)

T = np.linspace(1, 15, 1000)

Radius = 4

r = 1 + 1.5 \* sp.sin(12 \* t)

phi = 1.25 \* 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)

V = sp.sqrt(Vx \*\* 2 + Vy \*\* 2)

R = np.zeros\_like(T)

PHI = np.zeros\_like(T)

X = np.zeros\_like(T)

Y = np.zeros\_like(T)

VX = np.zeros\_like(T)

VY = np.zeros\_like(T)

# заполнение массивов

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

R[i] = sp.Subs(r, t, T[i])

PHI[i] = sp.Subs(phi, t, T[i])

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

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

VX[i] = sp.Subs(Vx, t, T[i])

VY[i] = sp.Subs(Vy, t, T[i])

# Начало координат

XX = [0 for i in range(1000)]

YY = [0 for i in range(1000)]

fig = plt.figure()

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

ax1.axis('equal')

ax1.set(xlim=[-2.5, 2.5], ylim=[-3, 3])

ax1.plot(X, Y)

# Точка, где находится тело

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

# рисуем вектор в нулевой момент времени (в anima будем рисовать с 1 до последнего момента времени)

Vline, = ax1.plot([X[0], X[0] + VX[0]], [Y[0], Y[0] + VY[0]], 'r')

Vline3, = ax1.plot([XX[0], X[0]], [YY[0], Y[0]], 'b')

# заполняем массивы векторов стрелочками

ArrowX = np.array([-0.2 \* Radius, 0, -0.2 \* Radius])

ArrowY = np.array([0.1 \* Radius, 0, -0.1 \* Radius])

ArrowWX = np.array([-Radius, 0, -Radius])

ArrowWY = np.array([Radius, 0, -Radius])

ArrowRX = np.array([-0.1 \* Radius, 0, -0.1 \* Radius])

ArrowRY = np.array([0.05 \* Radius, 0, -0.05 \* Radius])

RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(VY[0], VX[0]))

RArrowRX, RArrowRY = Rot2D(ArrowRX, ArrowRY, math.atan2(X[0], Y[0]))

VArrow, = ax1.plot(RArrowX + X[0] + VX[0], RArrowY + Y[0] + VY[0], 'r')

RArrow, = ax1.plot(ArrowRX + X[0], ArrowRY + Y[0], 'b')

# рисуем в каждый момент времени i нужные нам вектора

def anima(j):

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

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

Vline3.set\_data([XX[j], X[j]], [YY[j], Y[j]])

RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(VY[j], VX[j]))

VArrow.set\_data(RArrowX + X[j] + VX[j], RArrowY + Y[j] + VY[j])

RArrowRX, RArrowRY = Rot2D(ArrowRX, ArrowRY, math.atan2(Y[j], X[j]))

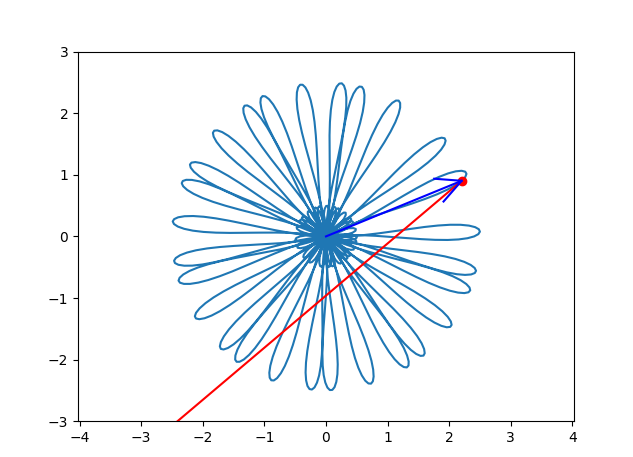
RArrow.set\_data(RArrowRX + X[j], RArrowRY + Y[j])

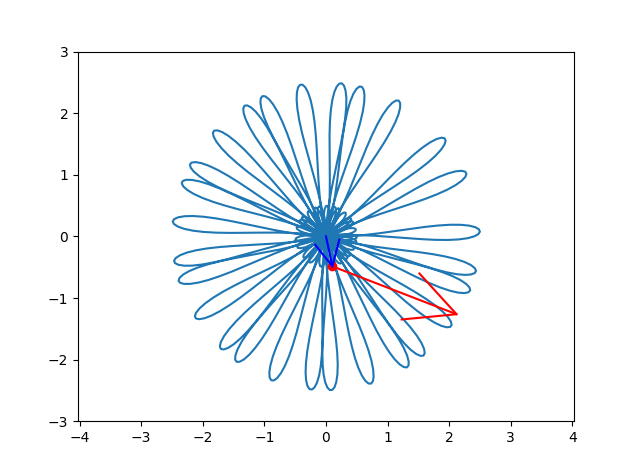
return P, Vline, VArrow, Vline3, RArrow

anim = FuncAnimation(fig, anima, frames=1000, interval=20, blit=True, repeat=True)

plt.show()

**Результат работы программы:**

****



**Вывод**

В ходе лабораторной работы я провёл изучение возможностей библиотек numpy, sympy и matplotlib для создания анимации точки на плоскости. Библиотека numpy позволила удобно работать с числовыми данными и выполнить необходимые операции. Sympy использовалась для символьного описания математических формул, делая процесс более гибким и понятным. Matplotlib предоставила мощные средства визуализации, позволяя создать информативную анимацию. В результате лабораторной работы было достигнуто понимание того, как объединить различные библиотеки Python для эффективного решения задач по визуализации анимации точки.