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*)

**Код программы(коммы добавь)**

import numpy as np

import sympy as sp

import math

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

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

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

t = sp.Symbol('t')

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)

Wx = sp.diff(Vx, t)

Wy = sp.diff(Vy, t)

X = np.zeros\_like(T)

Y = np.zeros\_like(T)

VX = np.zeros\_like(T)

VY = np.zeros\_like(T)

WX = np.zeros\_like(T)

WY = np.zeros\_like(T)

EP = np.zeros\_like(T)

PS = 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])

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

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

WX[i] = sp.Subs(Wx, t, T[i])

WY[i] = sp.Subs(Wy, t, T[i])

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

if(WY[i] > 0):

Eps = (x - (Vy \* ( 1 + (Vy\*\*2) ))/ (sp.Abs(Wy)) )

Ps = (y + (1 + (Vy\*\*2))/ (sp.Abs(Wy)) )

else:

Eps = (x + (Vy \* ( 1 + (Vy\*\*2) ))/ (sp.Abs(Wy)) )

Ps = (y - (1 + (Vy\*\*2))/ (sp.Abs(Wy)) )

EP[i] = sp.Subs(Eps, t, T[i])

PS[i] = sp.Subs(Ps, t, T[i])

fig = plt.figure()

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

ax1.axis('equal')

ax1.set(xlim=[-300, 300], ylim=[-300, 300])

ax1.plot(X, Y)

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

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

WLine, = ax1.plot([X[0],X[0] + WX[0]],[Y[0], Y[0] + WY[0]], 'g')

RLine, = ax1.plot([0 , X[0]], [0, Y[0]], 'k')

PLine, = ax1.plot( [X[0], EP[0]], [Y[0], PS[0]], 'm')

ArrowX = np.array([-0.2\*4, 0, -0.2\*4])

ArrowY = np.array([0.1\*4, 0, -0.1\*4])

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

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

WRArrowX, WRArrowY = Rot2D(ArrowX, ArrowY, math.atan2(WY[0], WX[0]))

WArrow, = ax1.plot(WRArrowX+X[0]+WX[0], WRArrowY+Y[0]+WY[0], 'g')

PRArrowX, PRArrowY = Rot2D(ArrowX, ArrowY, math.atan2(EP[0], PS[0]))

PArrow, = ax1.plot(PRArrowX+EP[0], PRArrowY+PS[0], 'm')

ArrowXR = np.array([-0.2, 0, -0.2])

ArrowYR = np.array([0.1, 0, -0.1])

RRArrowX, RRArrowY = Rot2D(ArrowXR, ArrowYR, math.atan2(Y[0], X[0]))

RRArrow, = ax1.plot(RRArrowX+X[0], RRArrowY+Y[0], 'k')

def anima(i):

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

VLine.set\_data([X[i], X[i]+VX[i]], [Y[i], Y[i]+VY[i]])

WLine.set\_data([X[i],X[i]+WX[i]], [Y[i], Y[i]+WY[i]])

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

PLine.set\_data([X[i], EP[i]], [Y[i], PS[i]])

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

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

WRArrowX, WRArrowY = Rot2D(ArrowX, ArrowY, math.atan2(WY[i], WX[i]))

WArrow.set\_data(WRArrowX+X[i]+WX[i], WRArrowY+Y[i]+WY[i])

RRArrowX, RRArrowY = Rot2D(ArrowXR, ArrowYR, math.atan2(Y[i], X[i]))

RRArrow.set\_data(RRArrowX+X[i], RRArrowY+Y[i])

PRArrowX, PRArrowY = Rot2D(ArrowX, ArrowY, math.atan2(EP[i], PS[i]))

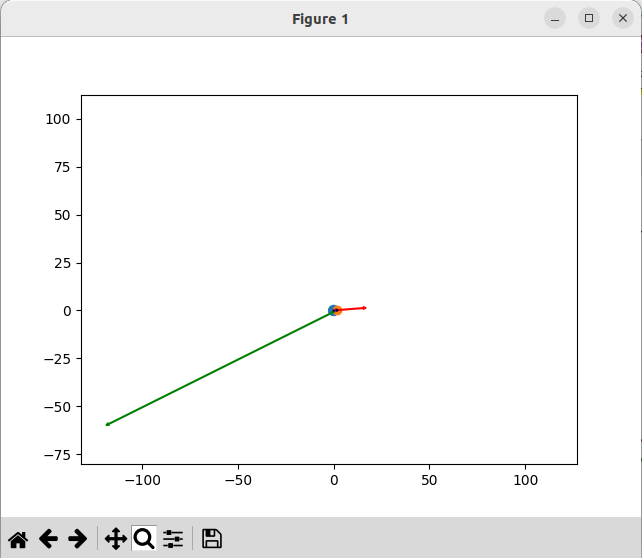
PArrow.set\_data(PRArrowX+EP[i], PRArrowY+PS[i])

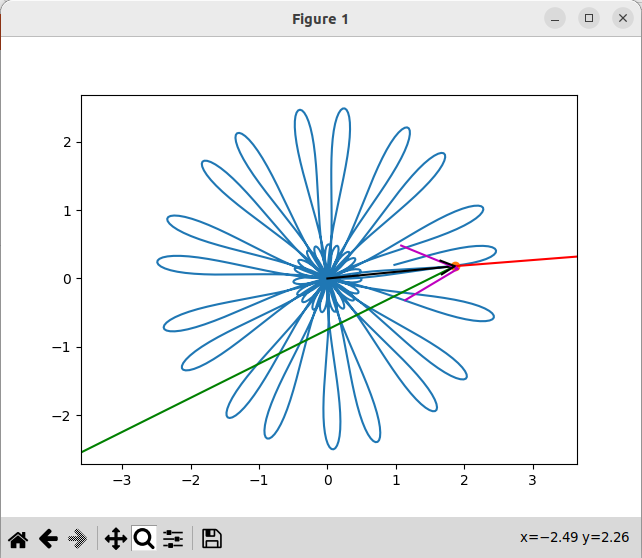
return P, VLine, VArrow, WLine, WArrow, RLine ,RRArrow ,PLine, PArrow

anim = FuncAnimation(fig, anima, frames=1000, interval=1, repeat = False)

plt.show()

**Результаты 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.