ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ ТОЧКИ»**

**ПО ДИСЦИПЛИНЕ «Основы Компьютерного Моделирования Математических Систем»**

**ВАРИАНТ ЗАДАНИЯ №18**

Выполнил(а) студент группы М8О-215Б-23

Самарский Я.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Ст. преп. каф. 802 Волков Е.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

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

Москва, 2024

**Вариант № 18**

**Задание:**

Построить заданную траекторию и анимацию движения точки, а также отобразить стрелки скорости и ускорения. Построить радиус кривизны траектории.

**Закон движения точки:**

r = 1 + 1.5sin(12t)

phi = 1.2t + 0.2cos(12t)

**Текст программы**

import matplotlib.pyplot as plt

import numpy as np

import sympy

from matplotlib.animation import FuncAnimation

def r(t):

    return 1 + 1.5 \* sympy.sin(12 \* t)

def phi(t):

    return 1.2 \* t + 0.2 \* sympy.cos(12 \* t)

def Rot2D(X, Y, phi):

    X\_r = X \* np.cos(phi) - Y \* np.sin(phi)

    Y\_r = X \* np.sin(phi) + Y \* np.cos(phi)

    return X\_r, Y\_r

t = sympy.Symbol("t")

# Переход от ПСК к ДСК

x = r(t) \* sympy.cos(phi(t))

y = r(t) \* sympy.sin(phi(t))

Vx = sympy.diff(x, t)

Vy = sympy.diff(y, t)

Wx = sympy.diff(Vx, t)

Wy = sympy.diff(Vy, t)

F\_x = sympy.lambdify(t, x, "numpy")

F\_y = sympy.lambdify(t, y, "numpy")

F\_Vx = sympy.lambdify(t, Vx, "numpy")

F\_Vy = sympy.lambdify(t, Vy, "numpy")

F\_Wx = sympy.lambdify(t, Wx, "numpy")

F\_Wy = sympy.lambdify(t, Wy, "numpy")

steps = 1000

T = np.linspace(0, 4 \* np.pi, steps)

X = F\_x(T)

Y = F\_y(T)

R\_phi = np.arctan2(Y, X)

VX = F\_Vx(T)

VY = F\_Vy(T)

V\_phi = np.arctan2(VY, VX)

WX = F\_Wx(T)

WY = F\_Wy(T)

W\_phi = np.arctan2(WY, WX)

V\_scale = 1/10

W\_scale = 1/100

Arrow\_scale = 1/10

fig = plt.figure()

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

ax1.axis("equal")

ax1.set(

    xlim=[-8, 8],

    ylim=[-8, 8],

    title=f"\

        Радиyс-вектор точки - серый вектор (масштаб 1.0)\n\

        Скорость (V) - зеленый вектор (масштаб {V\_scale})\n\

        Ускорение (W) - красный вектор (масштаб {W\_scale})",

)

ax1.plot(X, Y)

point, = ax1.plot(X[0], Y[0], marker="o")

# Добавление радиyс-вектора материальной точки

R\_color = [0.5, 0.5, 0.5]

R\_line, = ax1.plot(

    [0, X[0]],

    [0, Y[0]],

    color=R\_color,

)

# Добавление линии вектора скорости V

V\_color = [0, 0.7, 0]

V\_line, = ax1.plot(

    [X[0], X[0] + VX[0] \* V\_scale],

    [Y[0], Y[0] + VY[0] \* V\_scale],

    color=V\_color,

)

# Добавление линии вектора yскорения W

W\_color = [0.7, 0, 0]

W\_line, = ax1.plot(

    [X[0], X[0] + WX[0] \* W\_scale],

    [Y[0], Y[0] + WY[0] \* W\_scale],

    color=W\_color,

)

# Добавление стрелки радиyс-вектора материальной точки

X\_arr\_R = np.multiply(Arrow\_scale, np.array([-0.7, 0, -0.7])) # X координаты трех точек стрелки вектора V

Y\_arr\_R = np.multiply(Arrow\_scale, np.array([0.2, 0, -0.2])) # Y координаты трех точек стрелки вектора V

RX\_R, RY\_R = Rot2D(X\_arr\_R, Y\_arr\_R, R\_phi[0]) # Задание начального направления вектора V

R\_arrow, = ax1.plot(RX\_R, RY\_R, color=R\_color)

# Добавление стрелки вектора V

X\_arr\_V = np.multiply(Arrow\_scale, np.array([-0.7, 0, -0.7])) # X координаты трех точек стрелки вектора V

Y\_arr\_V = np.multiply(Arrow\_scale, np.array([0.2, 0, -0.2])) # Y координаты трех точек стрелки вектора V

RX\_V, RY\_V = Rot2D(X\_arr\_V, Y\_arr\_V, V\_phi[0]) # Задание начального направления вектора V

V\_arrow, = ax1.plot(RX\_V, RY\_V, color=V\_color)

# Добавление стрелки вектора W

X\_arr\_W = np.multiply(Arrow\_scale, np.array([-0.7, 0, -0.7])) # X координаты трех точек стрелки вектора W

Y\_arr\_W = np.multiply(Arrow\_scale, np.array([0.2, 0, -0.2])) # Y координаты трех точек стрелки вектора W

RX\_W, RY\_W = Rot2D(X\_arr\_W, Y\_arr\_W, W\_phi[0]) # Задание начального направления вектора W

W\_arrow, = ax1.plot(RX\_W, RY\_W, color=W\_color)

def animate(i):

    # Анимация материальной точки

    point.set\_data(X[i], Y[i])

    # Анимация радиyс-вектора материальной точки

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

    RX\_R, RY\_R = Rot2D(X\_arr\_V, Y\_arr\_V, R\_phi[i])

    R\_arrow.set\_data(X[i] + RX\_R , Y[i] + RY\_R)

    # Анимация вектора V

    V\_line.set\_data([X[i], X[i] + VX[i] \* V\_scale], [Y[i], Y[i] + VY[i] \* V\_scale])

    RX\_V, RY\_V = Rot2D(X\_arr\_V, Y\_arr\_V, V\_phi[i])

    V\_arrow.set\_data(X[i] + VX[i] \* V\_scale + RX\_V, Y[i] + VY[i] \* V\_scale + RY\_V)

    # Анимация вектора W

    W\_line.set\_data([X[i], X[i] + WX[i] \* W\_scale], [Y[i], Y[i] + WY[i] \* W\_scale])

    RX\_W, RY\_W = Rot2D(X\_arr\_W, Y\_arr\_W, W\_phi[i])

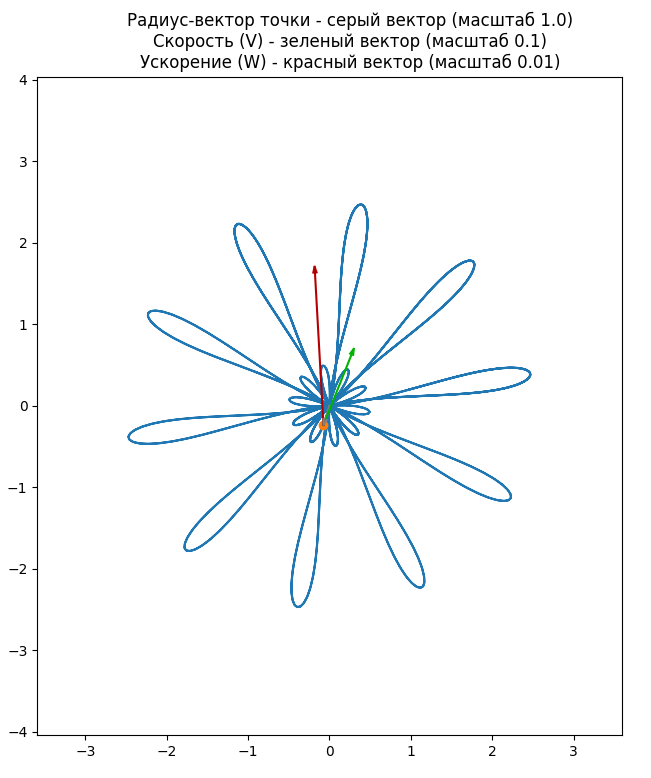
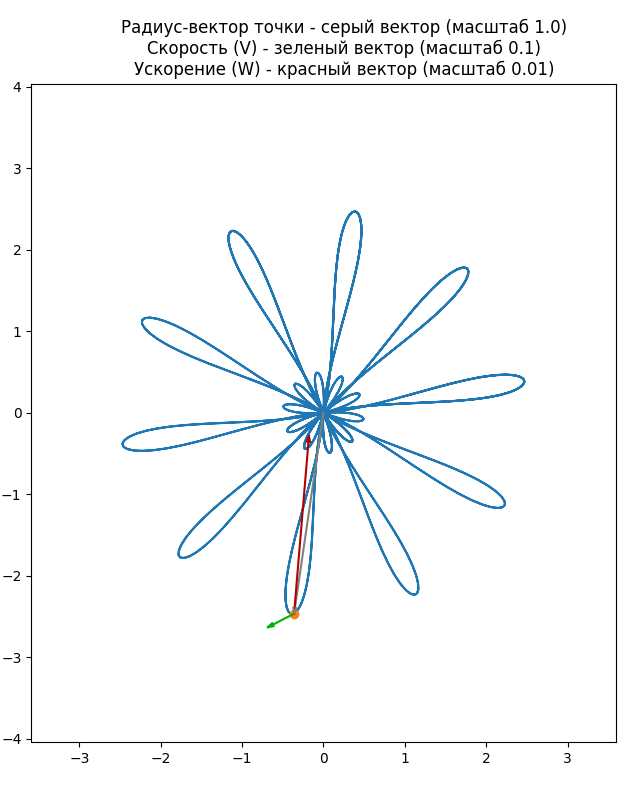
    W\_arrow.set\_data(X[i] + WX[i] \* W\_scale + RX\_W, Y[i] + WY[i] \* W\_scale + RY\_W)

    return point, R\_line, V\_line, V\_arrow, W\_line, W\_arrow

animation = FuncAnimation(fig, animate, frames=steps, interval=100)

plt.show()

**Результат работы программы:**



**Вывод:**

В процессе первой лабораторной работы я научился задавать движение точки по заданной траектории (например, в параметрической форме или в полярных координатах) и строить её анимацию. Мне удалось получить координаты x(t) и y(t) в каждый момент времени, а также вычислить скорость и ускорение точки. После этого я построил графики и создал анимацию, где были видны векторы скорости и ускорения. Это помогло мне лучше понять кинематику точки и увидеть преимущества использования символьных и численных методов в задаче о движении.