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 № 1**

Выполнил студент группы М8О-210Б-22

Андреев Иван Владимирович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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 + sin(t), Phi(t) = t

**Текст программы**

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

import sympy as sp

# Определение символьной переменной

t = sp.Symbol('t')

# Определение параметрических уравнений движущейся точки

r = 1 + sp.sin(t)

phi = t

x = r \* sp.cos(phi)

y = r \* sp.sin(phi)

# Вычисление производных по времени

Vx = sp.diff(x, t)

Vy = sp.diff(y, t)

Wx = sp.diff(Vx, t)

Wy = sp.diff(Vy, t)

# Определение центра вращения

cx = x - Vy \* ((Vx \* Vx + Vy \* Vy) / (Vx \* Wy - Wx \* Vy))

cy = y + Vx \* ((Vx \* Vx + Vy \* Vy) / (Vx \* Wy - Wx \* Vy))

# Преобразование символьных выражений в функции для численного вычисления

F\_x = sp.lambdify(t, x)

F\_y = sp.lambdify(t, y)

F\_Vx = sp.lambdify(t, Vx)

F\_Vy = sp.lambdify(t, Vy)

F\_Wx = sp.lambdify(t, Wx)

F\_Wy = sp.lambdify(t, Wy)

F\_cx = sp.lambdify(t, cx)

F\_cy = sp.lambdify(t, cy)

# Генерация временных значений

t = np.linspace(0, 10, 5000)

x = F\_x(t)

y = F\_y(t)

Vx = F\_Vx(t)

Vy = F\_Vy(t)

Wx = F\_Wx(t)

Wy = F\_Wy(t)

cx = F\_cx(t)

cy = F\_cy(t)

Alpha\_V = np.arctan2(Vy, Vx)

Alpha\_W = np.arctan2(Wy, Wx)

# Параметры для анимации

k\_V = 0.1

k\_W = 0.01

a = 0.1

b = 0.03

x\_arr = np.array([-a, 0, -a])

y\_arr = np.array([b, 0, -b])

# Функция для поворота точек вокруг начала координат

def Rot2D(X, Y, Alpha):

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

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

return RotX, RotY

# Создание графика

fig = plt.figure(figsize=[5, 5])

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

ax.axis('equal')

ax.set(xlim=[-2, 2], ylim=[-1, 3])

ax.plot(x, y)

# Инициализация элементов анимации

V\_line = ax.plot([x[0], x[0] + k\_V \* Vx[0]], [y[0], y[0] + k\_V \* Vy[0]], color=[1, 0, 0])[0]

Rot\_Vx, Rot\_Vy = Rot2D(x\_arr, y\_arr, Alpha\_V[0])

V\_arr = ax.plot(x[0] + k\_V \* Vx[0] + Rot\_Vx, y[0] + k\_V \* Vy[0] + Rot\_Vy, color=[1, 0, 0])[0]

W\_line = ax.plot([x[0], x[0] + k\_W \* Vx[0]], [y[0], y[0] + k\_W \* Vy[0]], color=[0, 0, 1])[0]

Rot\_Wx, Rot\_Wy = Rot2D(x\_arr, y\_arr, Alpha\_W[0])

W\_arr = ax.plot(x[0] + k\_W \* Wx[0] + Rot\_Wx, y[0] + k\_W \* Wy[0] + Rot\_Wy, color=[0, 0, 1])[0]

c\_line = ax.plot([x[0], cx[0]], [y[0], cy[0]], color=[0, 1, 0])[0]

P = ax.plot(x[0], y[0], marker='o')[0]

c = ax.plot(cx[0], cy[0], marker='o')[0]

# Функция для обновления анимации на каждом кадре

def kadr(i):

P.set\_data(x[i], y[i])

c.set\_data(cx[i], cy[i])

V\_line.set\_data([x[i], x[i] + k\_V \* Vx[i]], [y[i], y[i] + k\_V \* Vy[i]])

Rot\_Vx, Rot\_Vy = Rot2D(x\_arr, y\_arr, Alpha\_V[i])

V\_arr.set\_data(x[i] + k\_V \* Vx[i] + Rot\_Vx, y[i] + k\_V \* Vy[i] + Rot\_Vy)

W\_line.set\_data([x[i], x[i] + k\_W \* Wx[i]], [y[i], y[i] + k\_W \* Wy[i]])

Rot\_Wx, Rot\_Wy = Rot2D(x\_arr, y\_arr, Alpha\_W[i])

W\_arr.set\_data(x[i] + k\_W \* Wx[i] + Rot\_Wx, y[i] + k\_W \* Wy[i] + Rot\_Wy)

c\_line.set\_data([x[i], cx[i]], [y[i], cy[i]])

return [P, c, V\_line, V\_arr, W\_line, W\_arr, c\_line]

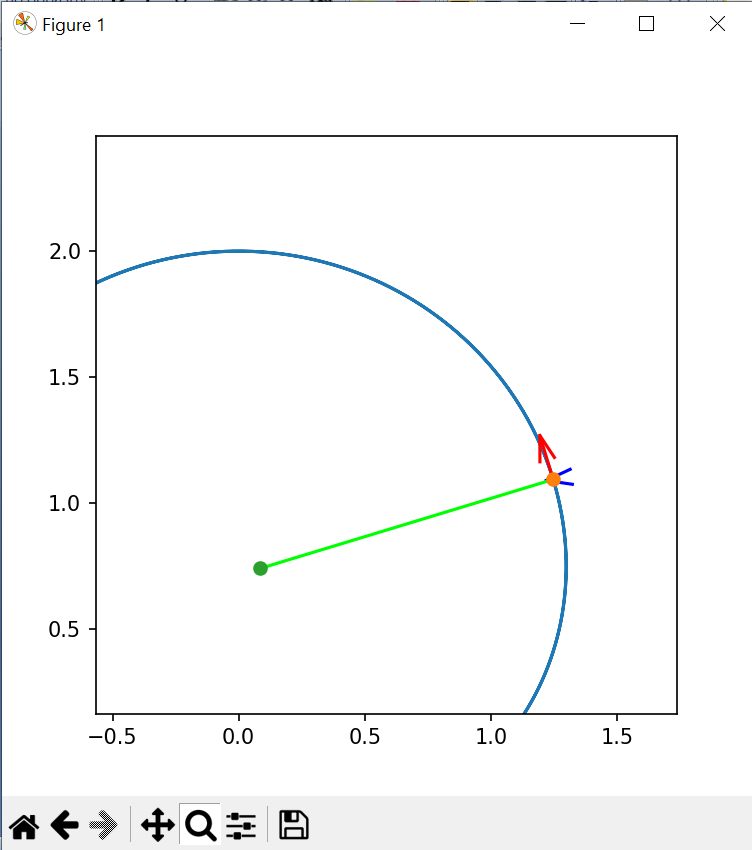
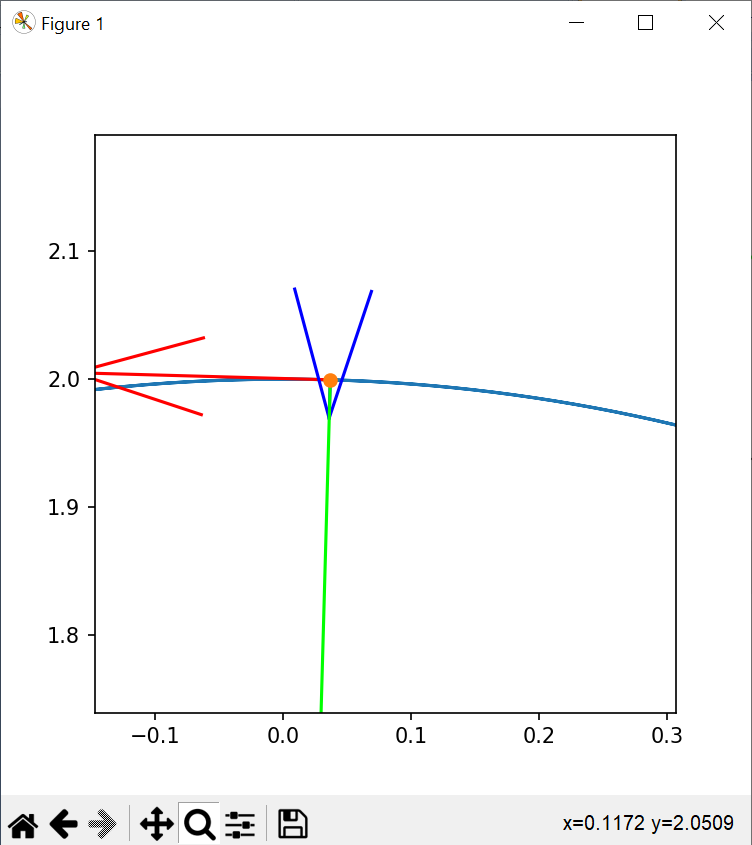
# Создание объекта анимации

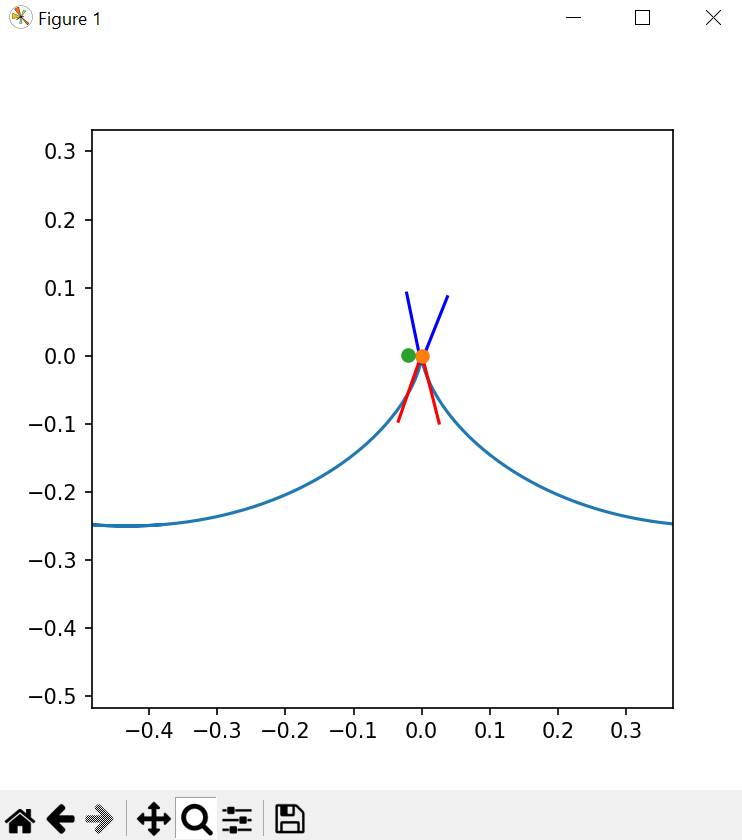
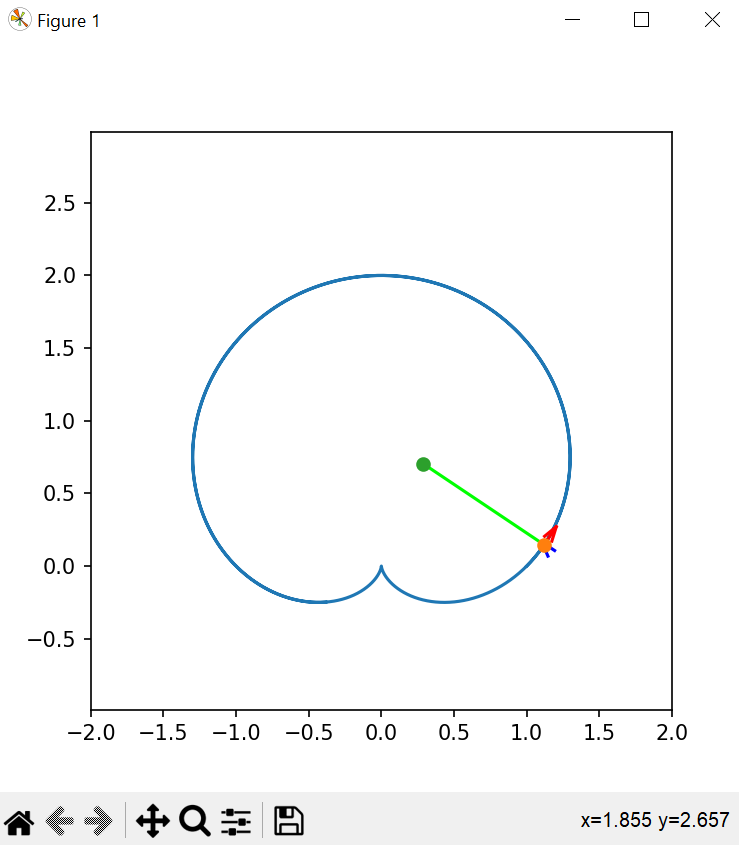
kino = FuncAnimation(fig, kadr, frames=len(t), interval=30)

# Отображение анимации

plt.show()

**Результат работы программы:**





**Вывод**

В ходе выполнения лабораторной работы были исследованы возможности библиотек numpy, sympy и matplotlib с целью создания анимации движения точки на плоскости. Библиотека numpy была использована для удобной работы с числовыми данными и выполнения необходимых математических операций. Sympy была привлечена для символьного описания математических формул, что сделало процесс более гибким и понятным. Библиотека matplotlib предоставила мощные инструменты для создания информативной анимации.

В результате работы было достигнуто понимание того, как эффективно объединять различные библиотеки Python с целью успешного решения задач визуализации анимации точки на плоскости.