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 № 15**

Выполнил(а) студент группы М8О-212Б-22

Куценко М.Д.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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 15 (r(t) и ф(t)):*

*Код:*

import numpy as np

import sympy as sp

import math

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation, FFMpegWriter

def Rot2D(X, Y, Alpha): # функция вращения стрелки

RX = X\*np.cos(Alpha) - Y\*np.sin(Alpha) # умножение на матрицу вращения

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

return RX, RY

t = sp.Symbol('t') #символьная переменная

x = (2+sp.sin(12\*t))\*sp.cos(1.8\*t+0.2\*sp.cos(12\*t)) # координата x

y = (2+sp.sin(12\*t))\*sp.sin(1.8\*t+0.2\*sp.cos(12\*t)) # координата y

Vx = sp.diff(x, t) # скорость по оси x || следующие выражения получаем дифференцированием

Vy = sp.diff(y, t) # скорость по оси y

Wx = sp.diff(Vx, t) # ускорение по оси x

Wy = sp.diff(Vy, t) # ускорение по оси y

minim = 10\*\*-5 # переменная для предотвращения деления на ноль

Dy = (Wy\*Vx-Wx\*Vy)/(Vy\*\*3+minim) # находим вторую параметрическую производную

V = (Vx\*\*2+Vy\*\*2)\*\*0.5 # находим общую скорость

W = (Wx\*\*2+Wy\*\*2)\*\*0.5 # находим общее ускорение

Wt = sp.diff(V,t) # находим тангенциальное ускорение

Wn = (W\*\*2-Wt\*\*2)\*\*0.5 # нахоидм нормальное ускорение

Rk = V\*\*2/(Wn+minim) # находим радиус кривизны

T = np.linspace(0, 10, 1000) # создаём массив из 1000 символов с равномерным распределением 10-ти (секунд) для точного моделирования

X = np.zeros\_like(T) # создаём пустые массивы такого же размера как 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)

DY = np.zeros\_like(T)

AN = np.zeros\_like(T)

RKX = np.zeros\_like(T)

RKY = np.zeros\_like(T)

for i in np.arange(len(T)): # проход по индексам на основе длины T

X[i] = sp.Subs(x, t, T[i]) # в массивах заменяем символьную переменную на отдельное значение времени

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])

DY[i] = sp.Subs(Dy, t, T[i])

AN[i] = DY[i]/(abs(DY[i])+minim)\*math.pi/2 + math.atan2(VY[i], VX[i]) # считаем угол для вектора кривизны

RKX[i] = sp.Subs(Rk, t, T[i]) # находим проекции вектора кривизны на оси

RKY[i] = RKX[i]

RKX[i] \*= sp.cos(AN[i])

RKY[i] \*= sp.sin(AN[i])

VX, VY, WX, WY, RKX, RKY = VX\*0.25, VY\*0.25, WX\*0.05, WY\*0.05, RKX\*0.2, RKY\*0.2 # укорачиваем величины

fig = plt.figure() # задаём пространство(?) для отрисовки фигур

ax1 = fig.add\_subplot(1, 1, 1) # задаём оси

ax1.axis('equal') # задаём равенство размерности осей

ax1.set(xlim=[-10, 10], ylim=[-10, 10]) # задаём предельные значения осей

ax1.plot(X, Y) # отрисовываем тракеторию движения

P, = ax1.plot(X[0], Y[0], marker='o') # точка, двигающаяся по траектории

VLine, = ax1.plot([X[0], X[0]+VX[0]], [Y[0], Y[0]+VY[0]], 'r') # "отрезок" вектора скорости

WLine, = ax1.plot([X[0], X[0]+WX[0]], [Y[0], Y[0]+WY[0]], 'g') # "отрезок" вектора ускорения

RLine, = ax1.plot([0, X[0]], [0, Y[0]], 'b') # "отрезок" радиус-вектора

RKLine, = ax1.plot([X[0], X[0]+RKX[0]], [Y[0], Y[0]+RKY[0]], 'y') # "отрезок" вектора кривизны

ArrowX = np.array([-0.5, 0, -0.5]) # кооординаты крайних точек стрелки

ArrowY = np.array([0.5, 0, -0.5])

VArrowX, VArrowY = Rot2D(ArrowX, ArrowY, math.atan2(VY[0], VX[0])) # позиция вращения стрелки скорости в нулевой момент времени

VArrow, = ax1.plot(VArrowX+X[0]+VX[0], VArrowY+Y[0]+VY[0], 'r') # размещение стрелки скорости по координатам в 0 с

WArrowX, WArrowY = Rot2D(ArrowX, ArrowY, math.atan2(WY[0], WX[0])) # позиция вращения стрелки ускорения в нулевой момент времени

WArrow, = ax1.plot(WArrowX+X[0]+WX[0], WArrowY+Y[0]+WY[0], 'g') # размещение стрелки ускорения по координатам в 0 с

RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(Y[0], X[0])) # позиция вращения стрелки радиус-вектора в нулевой момент времени

RArrow, = ax1.plot(RArrowX+X[0], RArrowY+Y[0], 'b') # размещение стрелки радиус-вектора по координатам в 0 с

RKArrowX, RKArrowY = Rot2D(ArrowX, ArrowY, AN[0]) # позиция вращения стрелки вектора кривизны в нулевой момент времени

RKArrow, = ax1.plot(RKArrowX+X[0]+RKX[0], RKArrowY+Y[0]+RKY[0], 'y') # размещение стрелки вектора кривизны по координатам в 0 с

def anima(i):

P.set\_data(X[i], Y[i]) # задаём координаты точки в каждый момент времени

VLine.set\_data([X[i], X[i]+VX[i]], [Y[i], Y[i]+VY[i]]) # задаём координаты для вектора скорости

VArrowX, VArrowY = Rot2D(ArrowX, ArrowY, math.atan2(VY[i], VX[i]))

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

WLine.set\_data([X[i], X[i]+WX[i]], [Y[i], Y[i]+WY[i]]) # задаём координаты для вектора ускорения

WArrowX, WArrowY = Rot2D(ArrowX, ArrowY, math.atan2(WY[i], WX[i]))

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

RLine.set\_data([0, X[i]], [0, Y[i]]) # задаём координаты для радиус-вектора

RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(Y[i], X[i]))

RArrow.set\_data(RArrowX+X[i], RArrowY+Y[i])

RKLine.set\_data([X[i], X[i]+RKX[i]], [Y[i], Y[i]+RKY[i]]) # задаём координаты для вектора кривизны

RKArrowX, RKArrowY = Rot2D(ArrowX, ArrowY, AN[i])

RKArrow.set\_data(RKArrowX+X[i]+RKX[i], RKArrowY+Y[i]+RKY[i])

return P, VLine, VArrow, WLine, WArrow, RLine, RArrow, RKLine, RKArrow # возвращаем величины для анимации

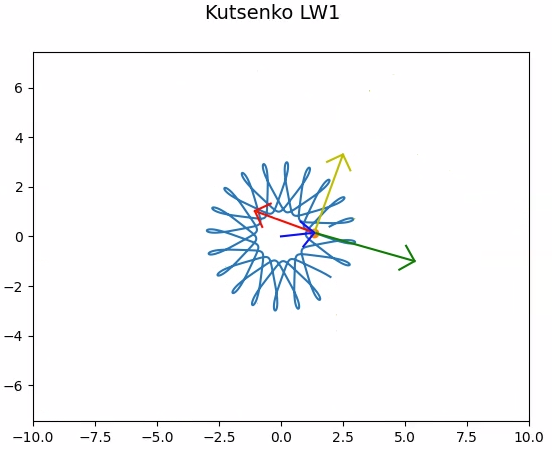
anim = FuncAnimation(fig, anima, frames=1000, interval=50, repeat=False) # создаём разовую анимацию

fig.suptitle('Kutsenko LW1', fontsize=14) # добавляем название

anim.save("./Animation.mp4", writer="ffmpeg") # сохраняем анимацию

plt.close()

*Результат работы программы:*

**

*Вывод:* научился использованию средств языка Python для моделирования движения точки по некоторой траектории в течение промежутка времени. Применил данные знания на практике при выполнении лабораторной работы.