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 № 6**

Выполнил(а) студент группы М8О-208Б-20

Зинин Владислав Владимирович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Доцент каф. 802, Чекина Е.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

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

Москва, 2021

**Задание:**

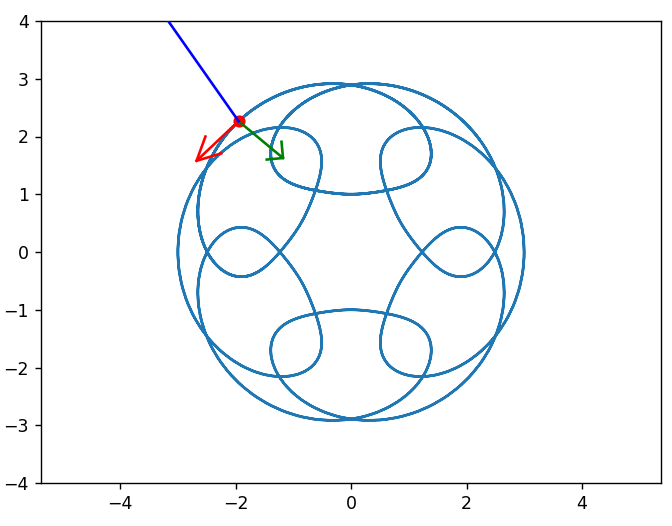
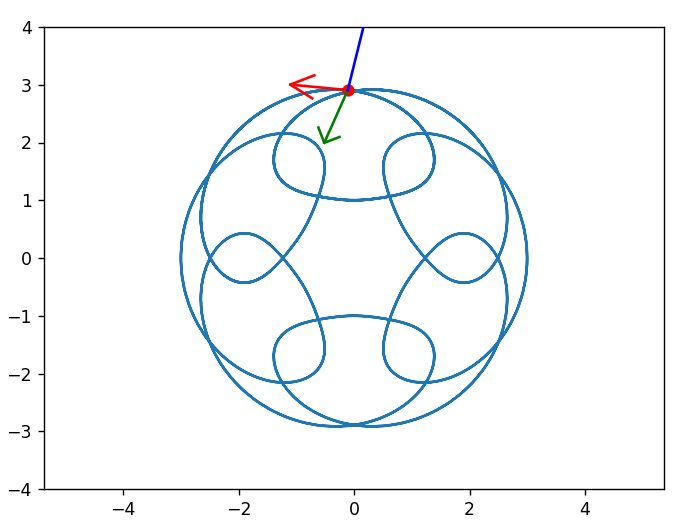
Построить заданную траекторию и анимацию движения точки, а также отобразить стрелки скорости и ускорения

**Закон движения:**

r = 2 + cos(6\*t)

phi = t + sin(6\*t)

**Результат работы программы**



Зеленая стрелочка указывает направление ускорения точки, красная – скорости.

Исходный код 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

def anima(j):

a = math.sqrt(VX[j] \*\* 2 + VY[j] \*\* 2)

b = math.sqrt(WX[j] \*\* 2 + WY[j] \*\* 2)

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

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

Vline2.set\_data([X[j], X[j] + WX[j] / b], [Y[j], Y[j] + WY[j] / b])

# Vline3.set\_data([X\_[j], X[j]], [Y\_[j], Y[j]])

Vline4.set\_data([X[j], X[j] + (Y[j] + VY[j]) \* Ro[j]/((Y[j] + VY[j])\*\*2 +

(X[j] + VX[j])\*\*2)\*\*0.5], [Y[j], Y[j] - (X[j] + VX[j]) \*

Ro[j]/((Y[j] + VY[j])\*\*2 + (X[j] + VX[j])\*\*2)\*\*0.5])

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

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

RArrowWX, RArrowWY = Rot2D(ArrowWX, ArrowWY, math.atan2(WY[j], WX[j]))

WArrow.set\_data(RArrowWX + X[j] + WX[j] / b, RArrowWY + Y[j] + WY[j] / b)

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

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

return P, Vline, VArrow, Vline2, WArrow, Vline4,

#Vline3,RArrow,

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

t = sp.Symbol('t')

R\_ = 4

Omega = 1

#functions

r = 2 + sp.cos(6\*t)

phi = t + sp.sin(6\*t)

x = r \* sp.cos(phi)

y = r \* sp.sin(phi)

Vx = sp.diff(x, t)

Wx = sp.diff(Vx, t)

Vy = sp.diff(y, t)

Wy = sp.diff(Vy, t)

W\_ = sp.sqrt(Wx \* Wx + Wy \* Wy)

W\_t = sp.diff(sp.sqrt(Vx\*\*2 + Vy\*\*2),t)

ro = (Vx\*\*2 + Vy\*\*2)/sp.sqrt((Wx \* Wx + Wy \* Wy) - sp.diff(sp.sqrt(Vx\*\*2 + Vy\*\*2), t)\*\*2)

#filling arrays with zeros

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)

WX = np.zeros\_like(T)

WY = np.zeros\_like(T)

W = np.zeros\_like(T)

W\_T = np.zeros\_like(T)

Ro = np.zeros\_like(T)

X\_ = [0 for i in range(1000)]

Y\_ = [0 for i in range(1000)]

#filling arrays

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])

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

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

W[i] = sp.Subs(W\_, t, T[i])

W\_T[i] = sp.Subs(W\_t, t, T[i])

Ro[i] = sp.Subs(ro, t, T[i])

#drawing

fig = plt.figure()

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

ax1.axis('equal')

ax1.set(xlim=[-R\_, R\_], ylim=[-R\_, R\_])

ax1.plot(X, Y)

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

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

Vline2, = ax1.plot([X[0], X[0] + WX[0]], [Y[0], Y[0] + WY[0]], 'g') # vector of acceleration

#Vline3, = ax1.plot([X\_[0], X[0]], [Y\_[0], Y[0]], 'b') # vector of radius vector

Vline4, = ax1.plot([X[0], X[0] + (Y[0] + VY[0]) \* Ro[0]/((Y[0] + VY[0])\*\*2 +

(X[0] + VX[0])\*\*2)\*\*0.5], [Y[0], Y[0] - (X[0] + VX[0]) \* Ro[0]/

((Y[0] + VY[0])\*\*2 + (X[0] + VX[0])\*\*2)\*\*0.5], 'b') # vector of radius of curvature

ArrowX = np.array([-0.1 \* R\_, 0, -0.1 \* R\_]) # arrow of speed

ArrowY = np.array([0.05 \* R\_, 0, -0.05 \* R\_])

ArrowWX = np.array([-0.05 \* R\_, 0, 0.05 \* -R\_]) # arrow of acceleration

ArrowWY = np.array([0.05 \* R\_, 0, 0.05 \* -R\_])

ArrowRX = np.array([-0.05 \* R\_, 0, -0.05 \* R\_]) # arrow of radius vector

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

# drawing an arrow at the end of a vector

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

RArrowWX, RArrowWY = Rot2D(ArrowWX, ArrowWY, math.atan2(WY[0], WX[0]))

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

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

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

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

anim = FuncAnimation(fig, anima, frames=2000, interval=80, blit=True)

plt.show()

**Вывод**

В результате проведенной лабораторной работы я познакомился с такими библиотеками, как matplotlib, numpy, sympy, научился анимировать математическую модель и повторил основные математические и физические уравнения.