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 №15**

Выполнил(а) студент группы М8О-207Б-22

КРУТЫХ РОМАН ВЯЧЕСЛАВОВИЧ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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.

*Условия задачи для 15 варианта:*

r(t) = 2 + sin(12t)

φ(t) =1.8t + 0.2cos(12t)

*Код программы:*

import numpy as np

import sympy as sp

import math

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

# Constant for decreasing the size of the arrows

decrease = 7

# Функция для вращения точек

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 = sp.Symbol('t')

T = np.linspace(0, 20, 1000)

r = 2 + sp.sin(12 \* t)

phi = 1.8 \* t + 0.2 \* sp.cos(12 \* t)

x = r \* sp.cos(phi)

y = r \* sp.sin(phi)

# Скорость

Vx = sp.diff(x, t) / decrease

Vy = sp.diff(y, t) / decrease

# Ускорение

Wx = sp.diff(Vx, t) / decrease

Wy = sp.diff(Vy, t) / decrease

# Тангециальное ускорение

t\_x = Vx / (Vx \*\* 2 + Vy \*\* 2) \*\* (1 / 2)

t\_y = Vy / (Vx \*\* 2 + Vy \*\* 2) \*\* (1 / 2)

wt\_x = sp.diff((Vx \*\* 2 + Vy \*\* 2) \*\* (1 / 2), t) \* t\_x / decrease

wt\_y = sp.diff((Vx \*\* 2 + Vy \*\* 2) \*\* (1 / 2), t) \* t\_y / decrease

# Центр кривизны

xc = x - Vy \* (Vx \*\* 2 + Vy \*\* 2) / (Vx \* Wy - Wx \* Vy)

yc = y + Vx \* (Vx \*\* 2 + Vy \*\* 2) / (Vx \* Wy - Wx \* Vy)

# Нормальное ускорение

wn\_x = (Wx - wt\_x) /decrease

wn\_y = (Wy - wt\_y)/decrease

X = np.zeros\_like(T)

Y = np.zeros\_like(T)

XC = np.zeros\_like(T)

YC = np.zeros\_like(T)

VX = np.zeros\_like(T)

VY = np.zeros\_like(T)

WY = np.zeros\_like(T)

WX = np.zeros\_like(T)

WT\_X = np.zeros\_like(T)

WT\_Y = np.zeros\_like(T)

WN\_X = np.zeros\_like(T)

WN\_Y = 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])

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

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

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

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

YC[i] = sp.Subs(yc, t, T[i])

XC[i] = sp.Subs(xc, t, T[i])

WT\_X[i] = sp.Subs(wt\_x, t, T[i])

WT\_Y[i] = sp.Subs(wt\_y, t, T[i])

WN\_Y[i] = sp.Subs(wn\_y, t, T[i])

WN\_X[i] = sp.Subs(wn\_x, t, T[i])

fig = plt.figure()

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

ax1.axis('equal')

ax1.set(xlim=[-6, 6], ylim=[-6, 6])

ax1.plot(X, Y, linestyle="--", color="gray")

point, = ax1.plot(X[0], Y[0], marker="o", color="#00B873")

# Кривизна

evoluta, = ax1.plot(XC[0], YC[0], marker=".", color="#210881")

curve\_line, = ax1.plot([XC[0], X[0]], [YC[0], Y[0]], linestyle="--", color="#7253E2", label="$радиус\\ кривизны$")

Rad = math.sqrt((XC[0] - X[0]) \*\* 2 + (YC[0] - Y[0]) \*\* 2)

curvature\_circle = plt.Circle((XC[0], YC[0]), Rad, alpha=0.3, edgecolor="#210881", facecolor="#7253E2")

ax1.add\_patch(curvature\_circle)

# Вектор скорости

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

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

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

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')

# Радиус-вектор

R\_Vline, = ax1.plot([0, X[0]], [0, Y[0]], color="orange", label="$\\rho\\ (радиус-вектор)$")

R\_RArrowX, R\_RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(Y[0], X[0]))

R\_VArrow, = ax1.plot(R\_RArrowX + X[0], R\_RArrowY + Y[0], 'r', color="#41E1A5")

# Вектор ускорения

W\_Vline, = ax1.plot([X[0], X[0] + WX[0]], [Y[0], Y[0] + WY[0]], color="#00D939", label="$Вектор \\ ускорения$")

W\_RArrowX, W\_RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(WY[0], WX[0]))

W\_VArrow, = ax1.plot(W\_RArrowX + X[0] + WX[0], W\_RArrowY + Y[0] + WY[0], color="#00D939")

# Вектор тангенциального ускорения

WT\_Vline, = ax1.plot([X[0], X[0] + WT\_X[0]], [Y[0], Y[0] + WT\_Y[0]], color="purple",

label="$тангенциальное \\ уравнение$")

WT\_RArrowX, WT\_RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(WT\_Y[0], WT\_X[0]))

WT\_VArrow, = ax1.plot(WT\_RArrowX + X[0] + WT\_X[0], WT\_RArrowY + Y[0] + WT\_Y[0], color="purple")

# Вектор нормального ускорения

WN\_Vline, = ax1.plot([X[0], X[0] + WN\_X[0]], [Y[0], Y[0] + WN\_Y[0]], color="green", label="$нормальное \\ ускорение$",

alpha=0.3)

WN\_RArrowX, WN\_RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(WN\_Y[0], WN\_X[0]))

WN\_VArrow, = ax1.plot(WN\_RArrowX + X[0] + WN\_X[0], WN\_RArrowY + Y[0] + WN\_Y[0], color="green", alpha=0.3)

def anima(i):

# Точка

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

# Кривизна

curve\_line.set\_data([XC[i], X[i]], [YC[i], Y[i]])

evoluta.set\_data(XC[i], YC[i])

Radius = ax1.plot((VX[i] \*\* 2 + VY[i] \*\* 2) / (

math.sqrt(WX[i] \*\* 2 + WY[i] \*\* 2 - sp.diff(math.sqrt(VX[i] \*\* 2 + VY[i] \*\* 2), t) \*\* 2)))

global curvature\_circle

curvature\_circle.remove()

Rad = math.sqrt((XC[i] - X[i]) \*\* 2 + (YC[i] - Y[i]) \*\* 2)

curvature\_circle = plt.Circle((XC[i], YC[i]), Rad, alpha=0.3, edgecolor="#210881", facecolor='#7253E2')

ax1.add\_patch(curvature\_circle)

# Вектор скорости

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

# Радиус-вектор

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

R\_RArrowX, R\_RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(Y[i], X[i]))

R\_VArrow.set\_data(R\_RArrowX + X[i], R\_RArrowY + Y[i])

# Вектор ускорения

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

W\_RArrowX, W\_RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(WY[i], WX[i]))

W\_VArrow.set\_data(R\_RArrowX + X[i] + WX[i], R\_RArrowY + Y[i] + WY[i])

# Вектор тангециального ускорения

WT\_Vline.set\_data([X[i], X[i] + WT\_X[i]], [Y[i], Y[i] + WT\_Y[i]])

WT\_RArrowX, WT\_RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(WT\_Y[i], WT\_X[i]))

WT\_VArrow.set\_data(WT\_RArrowX + X[i] + WT\_X[i], WT\_RArrowY + Y[i] + WT\_Y[i])

# Вектор нормального ускорения

WN\_Vline.set\_data([X[i], X[i] + WN\_X[i]], [Y[i], Y[i] + WN\_Y[i]])

WN\_RArrowX, WN\_RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(WN\_Y[i], WN\_X[i]))

WN\_VArrow.set\_data(WN\_RArrowX + X[i] + WN\_X[i], WN\_RArrowY + Y[i] + WN\_Y[i])

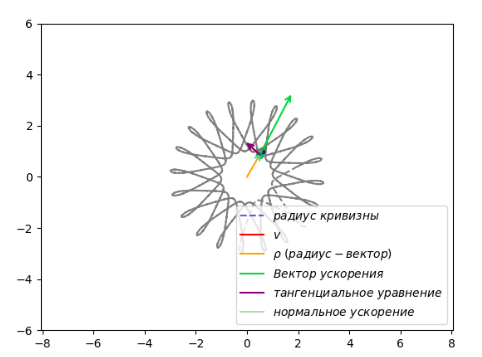
return point, evoluta, curve\_line, curvature\_circle, VLine, VArrow, R\_VArrow, R\_Vline, W\_Vline, W\_VArrow, WT\_Vline, WT\_VArrow, WN\_Vline, WN\_VArrow

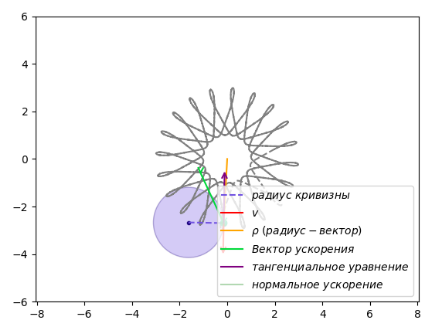
anim = FuncAnimation(fig, anima, frames=1000, interval=50, blit=True)

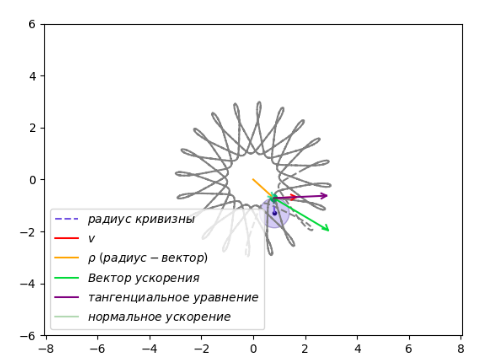
plt.legend()

plt.show()

*Рисунки получившейся физической модели:*





**

*Вывод:*

*В ходе выполнения данной работы я написал код на языке программирования Python, который строит заданную траекторию и запускает анимацию движения точки. Также были построены стрелки радиус вектора, вектора скорости и вектора ускорения.  
  
Эта работа помогла мне разобраться в основах теоретической механики и применении ее в программировании. Мне стало понятно , как использовать математические формулы для решения задач движения тела и как визуализировать результаты с помощью анимации.  
  
Также, я научился работать с графическими библиотеками Python, такими как Matplotlib и NumPy. Эти инструменты позволили мне создать графики и анимации. Я считаю, что полученные знания пригодятся мне в будущем, если я столкнусь со схожей задачей в работе.*