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 №16**

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

Лебедько Платон 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.

*1)Условия задачи 16 варианта:*

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

φ(t) = 1.8t + 0.2cos2(12t)

*2)Рисунок получившейся физической модели:*

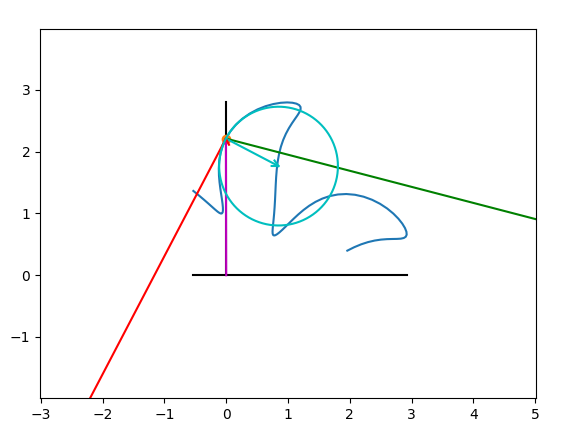
5

4

3

2

1



* «1» - радиус вектор
* «2» - вектор скорости
* «3» - вектор ускорения
* «4» – вектор кривизны
* «5» – окружность, по которой в данный момент движется точка

*3)Код программы:*

import numpy as np

import sympy as sp

import math

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

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

frames = 100

eps = 1e-18

T = np.linspace(0, 1, frames)

t = sp.Symbol('t')

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

phi = 1.8 \* t + 0.2 \* sp.cos(12 \* t) \*\* 2

x = r \* sp.cos(phi)

y = r \* sp.sin(phi)

Vx = sp.diff(x, t)

Vy = sp.diff(y, t)

v = (Vx \*\* 2 + Vy \*\* 2) \*\* 0.5

Wx = sp.diff(Vx, t)

Wy = sp.diff(Vy, t)

w = (Wx \*\* 2 + Wy \*\* 2) \*\* 0.5

Wtan = sp.diff(v, t)

Wnor = (w \*\* 2 - Wtan \*\* 2) \*\* 0.5

ro = v \*\* 2 / Wnor

WTanx = Vx / v \* Wtan

WTany = Vy / v \* Wtan

NotNormNx = Wx - WTanx

NotNormNy = Wy - WTany

NotNormn = (NotNormNx \*\* 2 + NotNormNy \*\* 2) \*\* 0.5

Nx = NotNormNx / NotNormn

Ny = NotNormNy / NotNormn

Curvax = Nx \* ro

Curvay = Ny \* ro

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)

RO = np.zeros\_like(T)

CurvaX = np.zeros\_like(T)

CurvaY = 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])

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])

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

CurvaX[i] = sp.Subs(Curvax, t, T[i])

CurvaY[i] = sp.Subs(Curvay, t, T[i])

fig = plt.figure()

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

ax1.axis('equal')

ax1.set(xlim=[int(X.min()) - 1, int(X.max()) + 1], ylim=[int(Y.min()) - 1, int(Y.max()) + 1])

ax1.plot(X, Y)

ax1.plot([min(0, X.min()), max(0, X.max())], [0, 0], 'black')

ax1.plot([0, 0], [min(0, Y.min()), max(0, Y.max())], 'black')

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

RLine, = ax1.plot([0, X[0]], [0, Y[0]], 'm')

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

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

CurvaLine, = ax1.plot([X[0], X[0]+CurvaX[0]], [Y[0], Y[0]+CurvaY[0]], 'c')

arrowMult = 0.5

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

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

RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(Y[0], X[0]))

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

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')

RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(WY[0], WX[0]))

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

RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(CurvaY[0], CurvaX[0]))

CurvaArrow, = ax1.plot(RArrowX+X[0]+CurvaX[0], RArrowY+Y[0]+CurvaY[0], 'c')

Phi = np.linspace(0, 2\*math.pi, 100)

CircCurva, = ax1.plot(X[0] + CurvaX[0] + RO[0] \* np.cos(Phi), Y[0] + CurvaY[0] + RO[0] \* np.sin(Phi), 'c')

def anima(i):

print(f"{i}: Vx = {VX[i]}, Vy = {VY[i]}, Wx = {WX[i]}, Wy = {WY[i]}")

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

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

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

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

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

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

RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(WY[i], WX[i]))

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

if (abs(VX[i] \*\* 2 + VY[i] \*\* 2 - WX[i] \*\* 2 - WY[i] \*\* 2) < eps):

CurvaLine.set\_data([0], [0])

CircCurva.set\_data([0], [0])

CurvaArrow.set\_data([0], [0])

print("Ужас!")

else:

CurvaLine.set\_data([X[i], X[i]+CurvaX[i]], [Y[i], Y[i]+CurvaY[i]])

CircCurva.set\_data(X[i] + CurvaX[i] + RO[i] \* np.cos(Phi), Y[i] + CurvaY[i] + RO[i] \* np.sin(Phi))

RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(CurvaY[i], CurvaX[i]))

CurvaArrow.set\_data(RArrowX+X[i]+CurvaX[i], RArrowY+Y[i]+CurvaY[i])

return P

anim = FuncAnimation(fig, anima, frames=frames, interval=200)

anim\_running = True

def onClick(event):

global anim\_running

if anim\_running:

anim.event\_source.stop()

anim\_running = False

else:

anim.event\_source.start()

anim\_running = True

fig.canvas.mpl\_connect('button\_press\_event', onClick)

plt.show()

*4) Пояснения:*

Основной задачей было нахождение радиуса кривизны, ведь остальные пункты были фактически реализованы на лекции.

Как буду искать вектор радиуса кривизны?

Очевидно, вектор радиуса кривизны сонаправлен с нормальным ускорением. Зная его и сам радиус – решим задачу.

Вектор полного ускорения - это сумма векторов нормального и тангенциального ускорений. Следовательно, вектор нормального ускорения - это разность векторов полного и тангенциального ускорений.

Но у нас нет вектора тангенциального ускорения - только его модуль (посчитанный, как производная модуля скорости по времени). Но оно сонаправлено со скоростью. Значит нужна нормировка вектора скорости, затем умножим его на известный модуль тангенциального ускорения. Таким образом получаем вектор тангенциального ускорения, а вектор полного ускорения легко найти, дифференцируя координаты скорости по времени. Следовательно, можем найти вектор нормального ускорения.

Нормируем вектор нормального ускорения, получим вектор n.

Теперь зная направляющий вектор (n) и радиус кривизны (посчитанный через нормальное ускорение и скорость) сможем отрисовать вектор радиуса кривизны.

*4) Вывод:*

Я успешно выполнил лабораторную работу по теоретической механике. С помощью языка Python и библиотек matplotlib, numpy и sympy я построил заданную траекторию, а также запустил анимацию движения точки по этой траектории. Для каждого момента времени я отрисовал векторы скорости, ускорения, радиус-вектора, вектора радиуса кривизны.

Эта лабораторная работа позволила мне лучше разобраться в теме движения точки, понять как связаны между собой разные характеристики движения точки – скорость и ускорения.