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 №5

Выполнил(а) студент группы М8О-201Б-22

Шляхтуров Александр Викторович_____
подпись, дата

Проверил и принял

Авдюшкин А.Н._____
подпись, дата

с оценкой _____

Москва, 2023

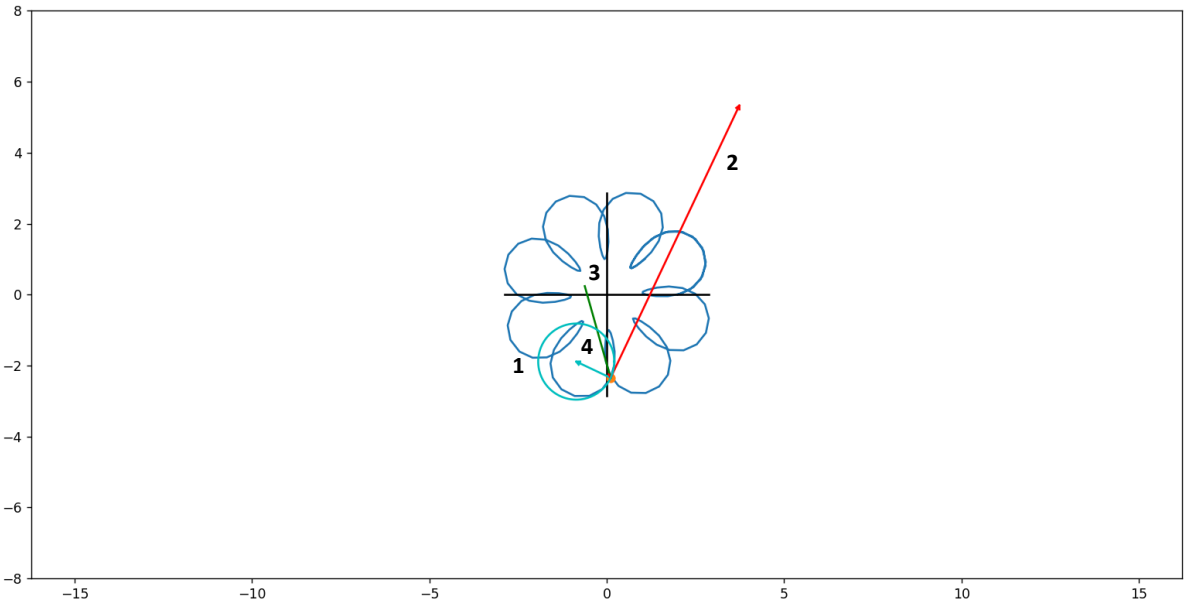
Задание: построить заданную траекторию, запустить анимацию движения точки, построить стрелки радиус-вектора, вектора скорости, вектора ускорения и радиуса кривизны.

1)Условия задачи 16 варианта:

$r(t) = 2 + \sin(8t)$

$\varphi(t) = t + 0.5\sin^2(4t)$

2)Рисунок получившейся физической модели:



1 - окружность, по которой движется точка
2 - вектор скорости
3 - вектор ускорения
4 – вектор кривизны

3)Код:

```
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

# Задаем количество разбиений
frames = 150
eps = 1e-18

# Разбиваем промежуток от 0 до 7 на 250 частей
T = np.linspace(0, 7, frames)

# Символическая переменная - время
t = sp.Symbol('t')

# Наши уравнения движения
r = 2 + sp.sin(8 * t)
phi = t + 0.5 * sp.sin(4 * t) ** 2

#Координаты x, y через r, phi
x = r * sp.cos(phi)
y = r * sp.sin(phi)

# Первые производные координат x, y по времени - составляющие вектора скорости
Vx = sp.diff(x, t)
Vy = sp.diff(y, t)
# модуль скорости по теореме пифагора
v = (Vx ** 2 + Vy ** 2) ** 0.5

# Вторые производные координат по времени - составляющие ускорения
Wx = sp.diff(Vx, t)
Wy = sp.diff(Vy, t)

# модуль полного ускорения
w = (Wx ** 2 + Wy ** 2) ** 0.5

# модуль тангенциального ускорения
Wtan = sp.diff(v, t)

# модуль нормального ускорения
Wnor = (w ** 2 - Wtan ** 2) ** 0.5

# Радиус кривизны
ro = v ** 2 / Wnor

# Радиус кривизны сонаправлен с нормальным ускорением
# 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
    # Но оно сонаправлено со скоростью - отнормируем вектор скорости, умножим на
модуль тангенциального ускорения
# Нормируем вектор нормального ускорения, получим вектор n
# Зная направляющий вектор и радиус кривизны сможем отрисовать кривизну



# Нормируем составляющие вектора скорости и умножаем на модуль тан.уск. Получаем
компоненты вектора тангенциального ускорения
# потому что тан.уск сонаправлено со скоростью
WTanx = Vx / v * Wtan
```

```
WTany = Vy / v * Wtan

# Получаем ненормированные компоненты нормального ускорения, вычитая из компонент
полного ускорения компоненты тангенциального
NotNormNx = Wx - WTanx
NotNormNy = Wy - WTany
NotNormn = (NotNormNx ** 2 + NotNormNy ** 2) ** 0.5

# Компоненты нормированного вектора нормального ускорения
Nx = NotNormNx / NotNormn
Ny = NotNormNy / NotNormn

# Компоненты вектора кривизны
Curvax = Nx * ro
Curvay = Ny * ro

# Инициализируем массивы для величин нулями
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)

# Заполняем массивы величин для построения графиков. Подставляем значение T[i] вместо
переменной t в выражение
for i in np.arange(len(T)):
    # радиус вектор точки
    X[i] = sp.Subs(x, t, T[i])
    Y[i] = sp.Subs(y, t, T[i])
    # вектор скорости
    VX[i] = sp.Subs(Vx, t, T[i])
    VY[i] = sp.Subs(Vy, t, T[i])
    # вектор ускорения
    WX[i] = sp.Subs(Wx, t, T[i])
    WY[i] = sp.Subs(Wy, t, T[i])
    # радиус кривизны
    RO[i] = sp.Subs(ro, t, T[i])
    # вектор кривизны
    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')

# Пределы для осей икс и игрек
ax1.set(xlim=[int(X.min()) - 6, int(X.max()) + 6], ylim=[int(Y.min()) - 6,
int(Y.max()) + 6])

# Рисуем траекторию движения
ax1.plot(X, Y)

# Рисуем координатные оси
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')

# Точка
P, = ax1.plot(X[0], Y[0], marker='o')
# Радиус вектор
# RLine, = ax1.plot([0, X[0]], [0, Y[0]], 'm')
# Вектора скорости, ускорения и кривизны
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])
# стрелка для радиус вектора
RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(Y[0], X[0]))
RArrow, = ax1.plot(RArrowX+X[0], RArrowY+Y[0], 'r')
# стрелка для вектора скорости
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')
# стрелка для вектора ускорения
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')
# стрелка для вектора кривизны
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')

# Отображение кривизны в виде окружности
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]}")
    # устанавливаем положение точки
    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], 0.01*(X[i]+WX[i]) ], [Y[i], 0.01*(Y[i]+WY[i]) ])

    # Рисуем стрелки для скорости в конкретной точке
    # радиус вектор
    RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(Y[i], X[i]))
    RArrow.set_data(RArrowX+X[i], RArrowY+Y[i])
    # вектор скорости
    RArrowX, RArrowY = Rot2D(ArrowX, ArrowY, math.atan2(VY[i], VX[i]))
    VArrow.set_data(RArrowX+X[i]+VX[i], RArrowY+Y[i]+VY[i])
    # вектор ускорения
    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

# Создаем анимацию
anim = FuncAnimation(fig, anima, frames=frames, interval=200)

anim_running = True

# делаем паузу на клик мышкой
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)
# Рисуем график
plt.show()
```

4) Пояснения:

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:

С помощью языка Python и библиотек matplotlib, numpy и sympy была создана анимация движения точки по траектории, заданной в параметрическом виде. В каждой точке движения тела были отрисованы
-окружность, по которой движется точка вектор скорости
-вектор ускорения
-вектор кривизны
-вектор скорости
Благодаря этой лабе, я научился пользоваться базовым функционалом библиотеки matplotlib для отрисовки линий, графиков и создания анимаций