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 № 7

Выполнил(а) студент группы М8О-208Б-23

Денисов Константин Дмитриевич _____
подпись, дата

Проверил и принял

Ст. преп. каф. 802 Волков Е.В_____
подпись, дата

с оценкой _____

Москва, 2024

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


Код программы

```
import math  # Импортируем библиотеку для работы с математическими функциями.
import sympy as s  # Импортируем библиотеку Sympy для символьных вычислений.
import matplotlib.pyplot as plot  # Импортируем библиотеку для построения графиков.
import numpy as np  # Импортируем библиотеку NumPy для работы с массивами.
from matplotlib.animation import FuncAnimation  # Импортируем класс для создания анимаций.

# Определяем функцию поворота точки в 2D-пространстве на заданный угол.
def rotation2D(x, y, angle):
    Rot_x = x * np.cos(angle) - y * np.sin(angle)  # Вычисляем новую координату x.
    Rot_y = x * np.sin(angle) + y * np.cos(angle)  # Вычисляем новую координату y.
    return Rot_x, Rot_y  # Возвращаем новые координаты.

# Определяем функцию для построения стрелки вектора.
def Vect_arrow(VecX, VecY, X, Y):
    a = 0.3  # Задаём длину стрелки.
    b = 0.2  # Задаём ширину стрелки.
    arrow_x = np.array([-a, 0, -a])  # Координаты контура стрелки по оси x.
    arrow_y = np.array([b, 0, -b])  # Координаты контура стрелки по оси y.

    phi = math.atan2(VecY, VecX)  # Вычисляем угол наклона вектора.

    RotX, RotY = rotation2D(arrow_x, arrow_y, phi)  # Поворачиваем стрелку на угол вектора.

    arrow_x = RotX + X + VecX  # Смещаем стрелку к концу вектора по оси x.
    arrow_y = RotY + Y + VecY  # Смещаем стрелку к концу вектора по оси y.

    return arrow_x, arrow_y  # Возвращаем координаты стрелки.

# Определяем функцию, которая будет обновлять данные для анимации на каждом кадре.
def anim(i):
    Pnt.set_data(X[i], Y[i])  # Обновляем положение точки на траектории.

    RVector.set_data([0, X[i]], [0, Y[i]])  # Обновляем радиус-вектор от начала координат до точки.
    RArrow.set_data(Vect_arrow(X[i], Y[i], 0, 0))  # Обновляем стрелку радиус-вектора.

    VVector.set_data([X[i], X[i] + X_velocity[i]], [Y[i], Y[i] + Y_velocity[i]])  # Обновляем вектор скорости.
```

```
    VArrow.set_data(Vect_arrow(X_velocity[i], Y_velocity[i], X[i], Y[i]))  # Обновляем стрелку скорости.

    AVector.set_data([X[i], X[i] + X_acceleration[i]], [Y[i], Y[i] + Y_acceleration[i]])  # Обновляем вектор ускорения.
    AArrow.set_data(Vect_arrow(X_acceleration[i], Y_acceleration[i], X[i], Y[i]))  # Обновляем стрелку ускорения.

    RCVector.set_data([X[i], X[i] + X_rcurvature[i]], [Y[i], Y[i] + Y_rcurvature[i]])  # Обновляем радиус кривизны.
    RCArrow.set_data(Vect_arrow(X_rcurvature[i], Y_rcurvature[i], X[i], Y[i]))  # Обновляем стрелку радиуса кривизны.

    return  # Завершаем обновление данных.

# Определяем символьные выражения для траектории движения точки.
t = s.Symbol('t')  # Символьная переменная для параметра времени.

r = 2 + s.cos(6 * t)  # Радиальная координата (расстояние от центра).
phi = t + 1.2 * s.cos(6 * t)  # Угловая координата (угол поворота).

x = r * s.cos(phi)  # Преобразуем радиальную и угловую координаты в x.
y = r * s.sin(phi)  # Преобразуем радиальную и угловую координаты в y.

# Вычисляем производные для определения скорости и ускорения.
x_velocity = s.diff(x)  # Производная по x — проекция скорости на ось x.
y_velocity = s.diff(y)  # Производная по y — проекция скорости на ось y.

x_acceleration = s.diff(x_velocity)  # Производная скорости по x — проекция ускорения на ось x.
y_acceleration = s.diff(y_velocity)  # Производная скорости по y — проекция ускорения на ось y.

Velocity = s.sqrt(x_velocity ** 2 + y_velocity ** 2)  # Модуль скорости.
Acceleration = s.sqrt(x_acceleration ** 2 + y_acceleration ** 2)  # Модуль ускорения.
Acceleration_t = s.diff(Velocity)  # Тангенциальное ускорение.
Acceleration_n = s.sqrt(Acceleration ** 2 - Acceleration_t ** 2)  # Нормальное ускорение.
RCurvature = (Velocity ** 2) / Acceleration_n  # Радиус кривизны.

# Задаём временной интервал и создаём массивы для хранения данных.
step = 2000  # Количество шагов (кадров) анимации.
T = np.linspace(0, 10, step)  # Временные значения от 0 до 10.

# Создаём пустые массивы для координат и векторов.
X = np.zeros_like(T)
Y = np.zeros_like(T)
X_velocity = np.zeros_like(T)
Y_velocity = np.zeros_like(T)
X_acceleration = np.zeros_like(T)
Y_acceleration = np.zeros_like(T)
X_rcurvature = np.zeros_like(T)
```

```
Y_rcurvature = np.zeros_like(T)

# Заполняем массивы вычисленными значениями.
for i in np.arange(len(T)):
    X[i] = s.Subs(x, t, T[i])  # Вычисляем x(t) для текущего времени.
    Y[i] = s.Subs(y, t, T[i])  # Вычисляем y(t) для текущего времени.

    X_velocity[i] = s.Subs(x_velocity, t, T[i])    # Вычисляем проекцию скорости на x.
    Y_velocity[i] = s.Subs(y_velocity, t, T[i])    # Вычисляем проекцию скорости на y.

    X_acceleration[i] = s.Subs(x_acceleration, t, T[i])  # Вычисляем проекцию ускорения на x.
    Y_acceleration[i] = s.Subs(y_acceleration, t, T[i])  # Вычисляем проекцию ускорения на y.

    # Вычисляем угол скорости и ускорения, чтобы найти радиус кривизны.
    Veloctity_angle = math.atan2(Y_velocity[i], X_velocity[i])    # Угол скорости.
    Acceleration_angle = math.atan2(Y_acceleration[i], X_acceleration[i])  # Угол ускорения.
    RCurvature_angle = Veloctity_angle - math.pi / 2 if Veloctity_angle - Acceleration_angle > 0 else Veloctity_angle + math.pi / 2  # Угол радиуса кривизны.

    # Вычисляем радиус кривизны как вектор.
    X_rcurvature[i] = RCurvature.subs(t, T[i]) * math.cos(RCurvature_angle)  # Проекция на x.
    Y_rcurvature[i] = RCurvature.subs(t, T[i]) * math.sin(RCurvature_angle)  # Проекция на y.

# Создаём фигуру и добавляем на неё графики.
fgr = plot.figure()  # Создаём окно графика.
grf = fgr.add_subplot(1, 1, 1)  # Создаём оси.
grf.axis('equal')  # Сохраняем одинаковый масштаб по осям.
grf.set(xlim=[-10, 10], ylim=[-10, 10])  # Устанавливаем границы графика.
grf.plot(X, Y)  # Рисуем траекторию движения точки.

# Добавляем на график начальные элементы анимации.
Pnt = grf.plot(X[0], Y[0], marker='o')[0]  # Точка на траектории.

# Радиус-вектор.
X_RArrow, Y_RArrow = Vect_arrow(X[0], Y[0], 0, 0)
RArrow = grf.plot(X_RArrow, Y_RArrow, '
```

Результат выполнения программы