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 №21

Выполнил(а) студент группы М80-201Б-22

Парфенов Михаил Максимович_____
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.

Закон движения: r = cos(6t), phi = t + 0.2*cos(3t).

Текст программы:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import sympy as sp
import math

#вариант 21
#r = cos(6t)
#phi = t + 0.2*cos(3t)

#поворот на угол alpha
def rotate(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 angle(a_x, a_y, b_x, b_y):
    ab = a_x * b_x + a_y * b_y
    mod_a = math.sqrt(a_x * a_x + a_y * a_y)
    mod_b = math.sqrt(b_x * b_x + b_y * b_y)
    return ab / (mod_a * mod_b)


t = sp.Symbol('t')
#перевод полярных координат в Декартовы
x = (sp.cos(6*t))*sp.cos(t + 0.2*sp.cos(3*t))
y = (sp.cos(6*t))*sp.sin(t + 0.2*sp.cos(3*t))
phi = t + 0.2*sp.cos(3*t)

#скорость
v_x = sp.diff(x, t)
v_y = sp.diff(y, t)
v_mod = sp.sqrt(v_x*v_x + v_y*v_y)

#ускорение
w_x = sp.diff(v_x, t)
w_y = sp.diff(v_y, t)
w_mod = sp.sqrt(w_x*w_x + w_y*w_y)

#эволюта (радиус кривизны)
evo_x = -sp.diff(y, t)*(sp.diff(x)**2 + sp.diff(y)**2)/(sp.diff(x, t)*sp.diff(y, t, 2) -
sp.diff(x, t, 2)*sp.diff(y, t))
evo_y = sp.diff(x, t)*(sp.diff(x)**2 + sp.diff(y)**2)/(sp.diff(x, t)*sp.diff(y, t, 2) -
sp.diff(x, t, 2)*sp.diff(y, t))
evo_mod = sp.sqrt(evo_x*evo_x + evo_y*evo_y)

#тангенсальное ускорение
cos_w_v = (v_x * w_x + v_y * w_y) / (v_mod * w_mod)
```

```
w_tau_x = (v_x / v_mod) * w_mod * cos_w_v
w_tau_y = (v_y / v_mod) * w_mod * cos_w_v

#нормальное ускорение
cos_w_evo = sp.sqrt(1 - cos_w_v*cos_w_v)
w_nor_x = (evo_x / evo_mod) * w_mod * cos_w_evo
w_nor_y = (evo_y / evo_mod) * w_mod * cos_w_evo

#1000 чисел от 1 до 10
T = np.linspace(0, 10, 1000)

#куча нулевых массивов "как" массив T
X = np.zeros_like(T)
Y = np.zeros_like(T)
V_X = np.zeros_like(T)
V_Y = np.zeros_like(T)
W_X = np.zeros_like(T)
W_Y = np.zeros_like(T)
W_N_X = np.zeros_like(T)
W_N_Y = np.zeros_like(T)
Phi = np.zeros_like(T)


#вычисление значений функций в каждой точке T
for i in np.arange(len(T)):
    X[i] = sp.Subs(x, t, T[i])
    Y[i] = sp.Subs(y, t, T[i])
    V_X[i] = sp.Subs(v_x, t, T[i])
    V_Y[i] = sp.Subs(v_y, t, T[i])
    W_X[i] = sp.Subs(w_tau_x, t, T[i])
    W_Y[i] = sp.Subs(w_tau_y, t, T[i])
    W_N_X[i] = sp.Subs(w_nor_x, t, T[i])
    W_N_Y[i] = sp.Subs(w_nor_y, t, T[i])
    Phi[i] = sp.Subs(phi, t, T[i])


#создание окна, поля с графиком и его параметров
fig = plt.figure(figsize=(10, 5))

ax1 = fig.add_subplot(1, 1, 1)
ax1.axis('equal')
ax1.set_title("Модель движения точки")
ax1.set_xlabel('X')
ax1.set_ylabel('Y')

ax1.plot(X, Y)


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

V_Line, = ax1.plot([X[0], X[0] + V_X[0]], [Y[0], X[0] + V_X[0]], 'r', label = 'Скорость')
W_Line, = ax1.plot([X[0], X[0] + W_X[0]], [Y[0], Y[0] + W_Y[0]], 'g', label = 'Тангенсальное
ускорение')
R_Line, = ax1.plot([0, X[0]], [0, Y[0]], 'black', label = 'Радиус-вектор')
W_N_Line, = ax1.plot([X[0], X[0] + W_N_X[0]], [Y[0], Y[0] + W_N_Y[0]], 'y', label =
'Нормальное ускорение')

R = math.sqrt(math.pow(X[0], 2) + math.pow(Y[0], 2))
```

```
#построение стрелочек
#для скорости
Arrow_X = np.array([-0.2*R, 0, -0.2*R])
Arrow_Y = np.array([0.1*R, 0, -0.1*R])
R_Arrow_X, R_Arrow_Y = rotate(Arrow_X, Arrow_Y, math.atan2(V_Y[0], V_X[0]))
V_Arrow, = ax1.plot(R_Arrow_X + X[0] + V_X[0], R_Arrow_Y + Y[0] + V_Y[0], 'r')

#для тангенсального ускорения
W_Arrow_X = np.array([-0.2*R, 0, -0.2*R])
W_Arrow_Y = np.array([0.1*R, 0, -0.1*R])
R_W_Arrow_X, R_W_Arrow_Y  = rotate(W_Arrow_X, W_Arrow_Y, math.atan2(W_Y[0], W_X[0]))
W_Arrow, = ax1.plot(R_W_Arrow_X + X[0] + W_X[0], R_W_Arrow_Y + Y[0] + W_Y[0], 'g')

#для радиус вектора
Arrow_R_X = np.array([-0.2*R, 0, -0.2*R])
Arrow_R_Y = np.array([0.1*R, 0, -0.1*R])
R_Arrow_RX, R_Arrow_RY = rotate(Arrow_R_X, Arrow_R_Y, math.atan2(Y[0], X[0]))
R_Arrow, = ax1.plot(R_Arrow_RX + X[0], R_Arrow_RY + Y[0], 'black')

#для нормального ускорения
W_N_Arrow_X = np.array([-0.2*R, 0, -0.2*R])
W_N_Arrow_Y = np.array([0.1*R, 0, -0.1*R])
R_W_N_Arrow_X, R_W_N_Arrow_Y  = rotate(W_N_Arrow_X, W_N_Arrow_Y, math.atan2(W_N_Y[0],
W_N_X[0]))
W_N_Arrow, = ax1.plot(R_W_N_Arrow_X + X[0] + W_N_X[0], R_W_N_Arrow_Y + Y[0] + W_N_Y[0], 'y')

ax1.legend(loc='lower center', ncol = 5, edgecolor = 'b')
ax1.set(xlim=[-4, 4], ylim=[-4, 4])


#функция анимацииУ
def anima(i):
    P.set_data(X[i], Y[i])

    W_Line.set_data([X[i], X[i] + W_X[i]], [Y[i],Y[i] + W_Y[i]])
    V_Line.set_data([X[i], X[i] + V_X[i]], [Y[i], Y[i] + V_Y[i]])
    R_Line.set_data([0, X[i]], [0, Y[i]])
    W_N_Line.set_data([X[i], X[i] + W_N_X[i]], [Y[i], Y[i] + W_N_Y[i]])

    R_Arrow_X, R_Arrow_Y = rotate(Arrow_X, Arrow_Y, math.atan2(V_Y[i], V_X[i]))
    R_W_Arrow_X, R_W_Arrow_Y = rotate(W_Arrow_X, W_Arrow_Y, math.atan2(W_Y[i], W_X[i]))
    R_Arrow_RX, R_Arrow_RY = rotate(Arrow_R_X, Arrow_R_Y, math.atan2(Y[i], X[i]))
    R_W_N_Arrow_X, R_W_N_Arrow_Y = rotate(W_N_Arrow_X, W_N_Arrow_Y, math.atan2(W_N_Y[i],
W_N_X[i]))

    V_Arrow.set_data(R_Arrow_X + X[i] + V_X[i], R_Arrow_Y + Y[i] + V_Y[i])
    W_Arrow.set_data(R_W_Arrow_X + X[i] + W_X[i], R_W_Arrow_Y + Y[i] + W_Y[i])
    R_Arrow.set_data(R_Arrow_RX + X[i], R_Arrow_RY + Y[i])
    W_N_Arrow.set_data(R_W_N_Arrow_X + W_N_X[i], R_W_N_Arrow_Y + Y[i] + W_N_Y[i])


    return P, V_Line, W_Line, R_Line, W_N_Line, V_Arrow, W_Arrow, R_Arrow, W_N_Arrow


#вывести график
anim = FuncAnimation(fig, anima, frames=1000, interval=100, blit=True)
plt.show()
```

Результат работы программы: