```python
#! python3.7
# -*- coding: utf-8 -*-
from numpy import zeros, sqrt, sum
from matplotlib.pyplot import style, figure, axes

# Функция f подготавливает массив, содержащий элементы вектор-функции,
# определяющей правую часть решаемой системы ОДУ
def f(u,m_sun,G) :
    f = zeros(5)
    # Определение исходной правой части (до перехода к длине дуги кривой)
    f[0] = u[2]
    f[1] = u[3]
    f[2] = - G*m_sun*u[0]/(u[0]**2 + u[1]**2)**(3/2)
    f[3] = - G*m_sun*u[1]/(u[0]**2 + u[1]**2)**(3/2)
    # Добавление ещё одной компоненты вектор-функции, которая отвечает за t
    # (выполнение формальной процедуры автономизации)
    f[4] = 1
    # Переопределение правой части (реализация перехода к длине дуги кривой)
    f = f/sqrt(1 + f[0]**2 + f[1]**2 + f[2]**2 + f[3]**2)
    return f

# Определение входных данных задачи
t_0 = 0.; T = 75.3*365.25*24*60*60
x_0 = 5.24824*10**12; y_0 = 0.
v_x_0 = 0.; v_y_0 = 0.9*10**3
G = 6.674301515151515*10**(-11)
m_sun = 1.98847*10**30

# Определение шага сетки вдоль интегральной кривой
dl = 2.*10**10

# Выделение памяти под массив сеточных значений решения системы ОДУ
# В строке с номером j этого массива хранятся сеточные значения решения,
# соответствующие j-ому узлу сетки вдоль дуги интегральной кривой
# Число J, определеяющее число строк массива, задаём с запасом
J = 10000
u = zeros((J,5))

# Задание начальных условий
# (записываются в строку с номером 0 массива u)
u[0] = [x_0, y_0, v_x_0, v_y_0, t_0]

# Порядок точности основной схемы ERK3
p = 3

# Оценка желаемой ошибки
eps = 0.000001*sqrt(x_0**2 + y_0**2 + v_x_0**2 + v_y_0**2)

j = 0
while u[j,4] < T :

    w_1 = f(u[j],m_sun,G)
    w_2 = f(u[j] + dl*1/2*w_1,m_sun,G)
    w_3 = f(u[j] + dl*3/4*w_2,m_sun,G)
    # Поиск решения по основной схеме ERK3
    u_main = u[j] + dl*(2/9*w_1 + 1/3*w_2 + 4/9*w_3)
    # Поиск решения по вложенной схеме ERK2
    u_embedded = u[j] + dl*w_2

    # Пересчёт оптимального шага dl
    norm = sqrt(sum((u_main - u_embedded)**2))
    dl = dl*(eps*dl/T/norm)**(1/(p - 1))

    # Пересчёт решения по основной схеме ERK3 с оптимальным шагом
```

```
        w_1 = f(u[j],m_sun,G)
        w_2 = f(u[j] + dl*1/2*w_1,m_sun,G)
        w_3 = f(u[j] + dl*3/4*w_2,m_sun,G)
        u[j + 1] = u[j] + dl*(2/9*w_1 + 1/3*w_2 + 4/9*w_3)

        j = j + 1

    # Отрисовка решения
    style.use('dark_background')

    fig = figure()
    ax = axes(xlim=(-1*10**12,6*10**12), ylim=(-3.5*10**12,3.5*10**12))
    ax.set_aspect('equal'); ax.set_xlabel('x'); ax.set_ylabel('y');
    ax.plot(0,0,'yo',markersize=10)
    ax.plot(u[0:j+1,0],u[0:j+1,1],'-ow',markersize=5)
    ax.plot(u[j,0],u[j,1], color='w', marker='o', markersize=7)
    ax.set_title('Траектория движения кометы Галлея')

    # Листинг программы, реализущей решение системы ОДУ
    # с автоматическим выбором шага
    # (на примереме моделирования движения кометы Галлея вокург Солнца)
    # (результатом яляется траектория кометы Галлея)
```