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

# Функция f подготавливает массив, содержащий элементы вектор-функции,
# определяющей правую часть решаемой системы ОДУ
def f(u,t,m_sun,G) :
    f = zeros(4)
    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)
    return f

# Определение входных данных задачи
t_0 = 0.; T = 365.25*24*60*60
x_0 = 147098291*10**3; y_0 = 0.
v_x_0 = 0.; v_y_0 = 30.4*10**3
G = 6.674301515151515*10**(-11)
m_sun = 1.98847*10**30

# Определение числа интервалов сетки,
# на которой будет искаться приближённое решение
M = 365

# Определение схемы ERKs, которая будет использоваться для расчётов
# (далее необходимо раскомментировать требуемый набор коэффициентов)

# s = 1 # ERK1
# b = zeros(s); a = zeros((s,s)); c = zeros(s)
# b[0] = 1.; c[0] = 1/2

s = 2 # ERK2
b = zeros(s); a = zeros((s,s)); c = zeros(s)
b[0] = 1/4; b[1] = 3/4;
a[1,0] = 2/3; c[0] = 0.; c[1] = 2/3

# s =3 # ERK3
# b = zeros(s); a = zeros((s,s)); c = zeros(s)
# b[0] = 2/9; b[1] = 1/3; b[2] = 4/9
# a[1,0] = 1/2; a[2,0] = 0.; a[2,1] = 3/4; c[0] = 0.; c[1] = 1/2; c[2] = 3/4

# s = 4 # ERK4
# b = zeros(s); a = zeros((s,s)); c = zeros(s)
# b[0] = 1/6; b[1] = 1/3; b[2] = 1/3; b[3] = 1/6
# a[1,0] = 1/2; a[2,0] = 0.; a[2,1] = 1/2; a[3,0] = 0.; a[3,1] = 0.; a[3,2] = 1.
# c[0] = 0.; c[1] = 1/2; c[2] = 1/2; c[3] = 1.

# Определение сетки
tau = (T - t_0)/M
t = linspace(t_0,T,M + 1)

# Выделение памяти под массив сеточных значений решения системы ОДУ
# В строке с номером m этого массива хранятся сеточные значения решения,
# соответствующие моменту времени t_m
u = zeros((M + 1,4))

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

# Реализация схемы ERKs
# (отметим, что во вспомогательных массивах adjustment_1,2 и w
# размерность 4 соответствует числу компонент вектор-функции f)
for m in range(M) :
    w = zeros((s,4))
    for k in range(s) :
        adjustment_1 = zeros(4)
```

```
            for l in range(k) :
                adjustment_1 = adjustment_1 + a[k,l]*w[l]
            w[k] = f(u[m] + tau*adjustment_1,t[m] + tau*c[k],m_sun,G)
        adjustment_2 = zeros(4)
        for k in range(s) :
            adjustment_2 = adjustment_2 + b[k]*w[k]
        u[m + 1] = u[m] + tau*adjustment_2

# Отрисовка решения
style.use('dark_background')

fig = figure()
ax = axes(xlim=(-2*10**11,2*10**11), ylim=(-2*10**11,2*10**11))
ax.set_aspect('equal'); ax.set_xlabel('x'); ax.set_ylabel('y');
ax.plot(0,0,'yo',markersize=15)
ax.plot(u[:,0],u[:,1],'-w',markersize=5)
ax.plot(u[M,0],u[M,1], color='w', marker='o', markersize=7)
ax.set_title('Траектория движения Земли')

# Листинг программы, реализущей решение системы ОДУ
# с помощью одной из схем семейства схем Рунге-Кутты ERKs
# (на примереме моделирования движения Земли вокург Солнца)
# (результатом яляется траектория Земли)
```