Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Казанский (Приволжский) Федеральный университет

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра прикладной математики и искусственного интеллекта

Направление подготовки: 01.03.04 - «Прикладная математика»

КУРСОВАЯ РАБОТА

по дисциплине: «Численные методы»

Ограниченная задача трех тел.

Студент 2 курса
группы 09-222
«____» ________ 2024 г. ______________ И.И. Романов
Научный руководитель
ассистент б. с.
«____» ________ 2024 г. ______________ О.В. Глазырина

Казань, 2024

Содержание

Постановка задачи

Рассмотрим два тела с массами m (Луна) и $M = 1 - m$ (Земля), участвующие в совместном круговом движении в плоскости xOy и расположенные в точках с координатами $(1, 0)$ и $(0, 0)$ соответственно. Пусть далее близ этих тел в той же плоскости движется третье тело пренебрежимо малой массы и $(x(t), y(t))$ - его координаты в момент времени t. Траектория движения этого тела описывается уравнениями:

$$
\begin{aligned}
&x'' = x + 2y' - \frac{M(x+m)}{R_1} - \frac{m(x-M)}{R_2}, \\
&y'' = y - 2x' - M\frac{y}{R_1} - m\frac{y}{R_2}, \\
&R_1 = \left((x+m)^2 + y^2\right)^{3/2}, R_2 = \left((x-M)^2 + y^2\right)^{3/2}.
\end{aligned} \tag{1}
$$

Уравнения (1) дополняются начальными условиями:

$$
x(0) = 0.994,\ y(0) = 0,\ x'(0) = 0,\ y'(0) = -2.031732629557337. \tag{2}
$$

При начальных условиях (2) и $m = 0.012277471$ орбита будет периодической с периодом обращения, равным $T = 11.124340337$ (такие орбиты называют "орбитами Аренсторфа").

Для решения полученной задачи свести её к задаче Коши для системы уравнений первого порядка вида

$$
y' = f(t, y),\ y(0) = y_0,\ y(t) \in R^n
$$

и использовать метод Рунге-Кутты 4-го порядка точности:

$$
\begin{aligned}
&k_1 = f(t_n, y_n), && k_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{hk_1}{2}\right), \\
&k_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{hk_2}{2}\right), && k_4 = f\left(t_n + h, y_n + hk_3\right), \\
&y_{n+1} = y_n + h(k_1 + 2k_2 + 2k_3 + k_4)/6.
\end{aligned} \tag{3}
$$

Для проверки правильности работы программы решить тестовую задачу из двух уравнений

$$
\begin{aligned}
&y_1' = -y_2 + y_1(y_1^2 + y_2^2 - 1), \\
&y_2' = y_1 + y_2(y_1^2 + y_2^2 - 1)
\end{aligned} \tag{4}
$$

на отрезке $[0, 5]$ с точным решением

$$
y_1 = \frac{\cos(x)}{\sqrt{1 + e^{2x}}}, \quad y_2 = \frac{\sin(x)}{\sqrt{1 + e^{2x}}}.
$$

В ходе работы необходимо выполнить следующие действия:

1. Проверить правильность тестового решения.

2. Написать процедуру интегрирования задачи Коши для системы из n обыкновенных дифференциальных уравнений второго порядка по формулам (3) на произвольном отрезке $[a, b]$ с постоянным шагом h.

3. Для тестовой задачи (4) построить графики зависимости максимальной погрешности решения e и e/h^4 от выбранного шага h.

4. Рассчитать орбиту Аренсторфа. Учесть, что решения бесконечно дифференцируемы всюду за исключением двух точке $(-m, 0)$, $(M, 0)$. Поэтому в окрестности начала и конца отрезка интегрирования необходимо выбирать существенно меньший шаг интегрирования h, чем в другие моменты времени. Построить график орбиты в координатах (x, y)

Ход работы

Для проверки тестовой задачи реализуем метод решения системы уравнений (4) с помощью формул (3). Так как система (4) является системой дифференциальных уравнений первого порядка, то никаких дополнительных замен переменных не потребуется.

В качестве начальных значений для нашей системы возьмём значения точных решений в начале отрезка $[0, 5]$ – точке 0. Откуда получим, что

$$y_1(0) = \frac{1}{\sqrt{2}}, \quad y_2(0) = 0.$$

Далее, выбрав значение шага $h = 0,01$, решим систему (4) методом Рунге-Кутты 4 порядка точности на отрезке $[0, 5]$ и сравним с точными решением системы. Получим данные графики:

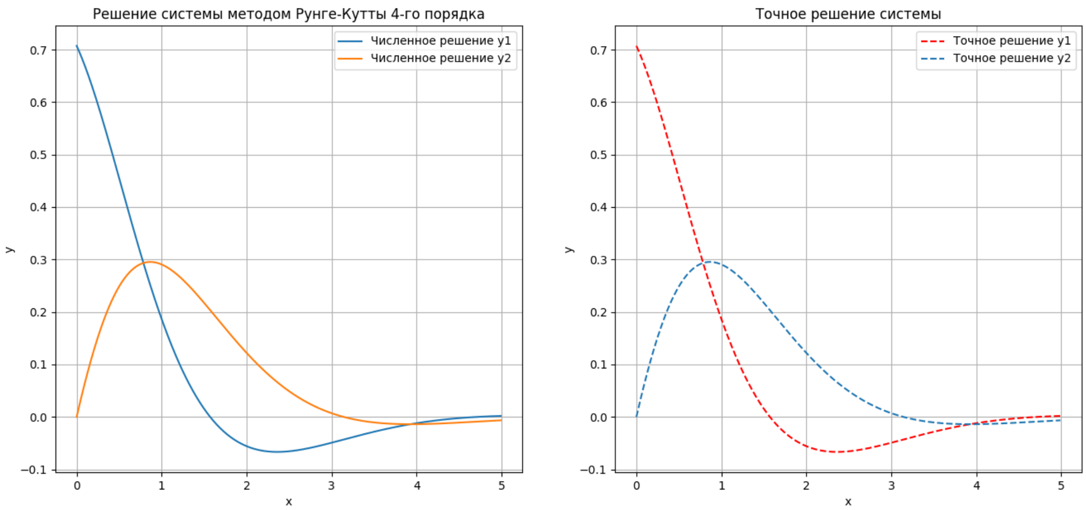

Рис.1 – графики решения тестовой задачи методом Рунге-Кутты и точного решения

Наложив данные графики друг на друга, убедимся, что решения сходятся:

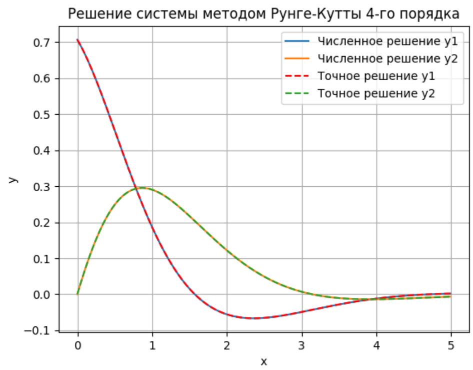

Рис.2 – наложенные графики решения методом Рунге-Кутты и точного решения

Убедившись в правильности решения метода Рунге-Кутты 4 порядка точности, реализуем процедуру интегрирования задачи Коши для системы из n уравнений второго порядка на произвольном отрезке $[a, b]$ с постоянным шагом h.

Пусть дана система обыкновенных дифференциальных уравнений второго порядка вида:

$$
\begin{aligned}
\frac{d^2y_1}{dt^2} &= F_1(t, y_1, y_1', y_2, y_2', \dots, y_n, y_n') \\
\frac{d^2y_2}{dt^2} &= F_2(t, y_1, y_1', y_2, y_2', \dots, y_n, y_n') \\
&\vdots \\
\frac{d^2y_n}{dt^2} &= F_n(t, y_1, y_1', y_2, y_2', \dots, y_n, y_n')
\end{aligned}
$$

где $y_i' = \dfrac{dy_i}{dt}$.

Данную систему можно преобразовать в эквивалентную систему из $2n$ обыкновенных дифференциальных уравнений первого порядка, сделав замену переменных. Введём новые переменные $z_i = \dfrac{dy_i}{dt}$, чтобы сделать эквивалентное преобразование. Получим систему вида:

$$
\begin{aligned}
\frac{dy_1}{dt} &= z_1 \\
\frac{dz_1}{dt} &= F_1(t, y_1, z_1, y_2, z_2, \dots, y_n, z_n) \\
\frac{dy_2}{dt} &= z_2 \\
\frac{dz_2}{dt} &= F_2(t, y_1, z_1, y_2, z_2, \dots, y_n, z_n) \\
&\vdots \\
\frac{dy_n}{dt} &= z_n \\
\frac{dz_n}{dt} &= F_n(t, y_1, z_1, y_2, z_2, \dots, y_n, z_n)
\end{aligned}
$$

Получив систему из $2n$ обыкновенных дифференциальных уравнений первого порядка, применим начальные значения системы второго порядка для получившейся системы. Далле для каждого получившегося уравнения системы обыкновенных дифференциальных уравнений первого порядка применим метод Рунге-Кутты 4 порядка точности на отрезке $[a, b]$ с точным шагом h.

Правильность решения метода Рунге-Кутты 4 порядка точности во многом зависит от выбранного значения h. На примере тестовой задачи рассмотрим поведение максимальной ошибки вычисления в зависимоти от выбранного значения h. Для этого проведём вычисления тестовой задачи для $h = 0,001,\ \dots,\ 0,1$, постепенно увеличивая шаг. Рассмотрим значение максимальной ошибки вычисления e и e/h^4 для данных шагов.

Построим графики зависимостей: Данные графики показывают, что значение максимальной вычислительной ошибки e возрастают с уравнением значения шага h.

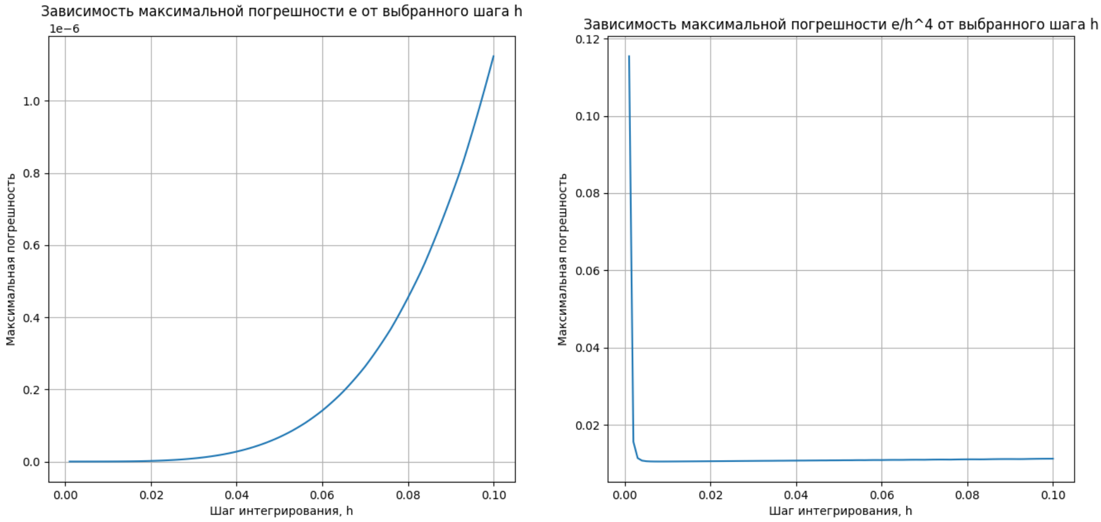

Рис.3 – графики зависимости значения максимальной ошибки e и e/h^4 от выбранного шага h

Теперь расчитаем орбиту Аренсторфа, решив систему (1) с начальными условиями (2) при помощи формул (3) метода Рунге-Кутты 4 порядка точности. Пусть $u_1 = x$, $u_2 = x'$, $u_3 = y$, и $u_4 = y'$.

Мы можем записать систему (1) с помощью обыкновенных дифференциальных уравнений первого порядка в следующем виде:

$$\begin{cases} u_1' = u_2 \\ u_2' = u_1 + 2u_4 - M\frac{(u_1 - m)}{R_1^{\frac{3}{2}}} - m\frac{(u_1 - M)}{R_2^{\frac{3}{2}}} \\ u_3' = u_4 \\ u_4' = u_3 - 2u_2 - M\frac{u_3}{R_1^{\frac{3}{2}}} - m\frac{u_3}{R_2^{\frac{3}{2}}} \end{cases} \tag{5}$$

Теперь определим R_1 и R_2:

$$R_1 = \left((u_1 + m)^2 + u_3^2\right)^{\frac{3}{2}}, \quad R_2 = \left((u_1 - M)^2 + u_3^2\right)^{\frac{3}{2}}$$

Определим начальные условия:

$$u_1(0) = 0,994, \; u_2(0) = 0, \; u - 3(0) = 0, \; u_4(0) = -2,031732629557337$$

Далее определимся с отрезком интегрирования. Поскольку нужная нам орбита Аренсторфа имеет период обращения $T = 11.124340337$, то будем считать, что отрезком интегрирования будет служить $[0, T]$. Шаг возьмём равным $h = 0.01$.

Определившись с отрезком интегрирования и шагом, реализуем метод Рунге-Кутты 4 порядка точности по формулам (3) для получившейся системы (5).

Вычисления строим таким образом, чтобы учесть тот факт, что решения задачи бесконечно дифференцируемы всюду, кроме точек $(-m, 0)$ $(M, 0)$. В окрестности начала и конца отрезка интегрирования выберем существенно меньший шаг интегрирования h. Построим график орбиты Аренсторфа.

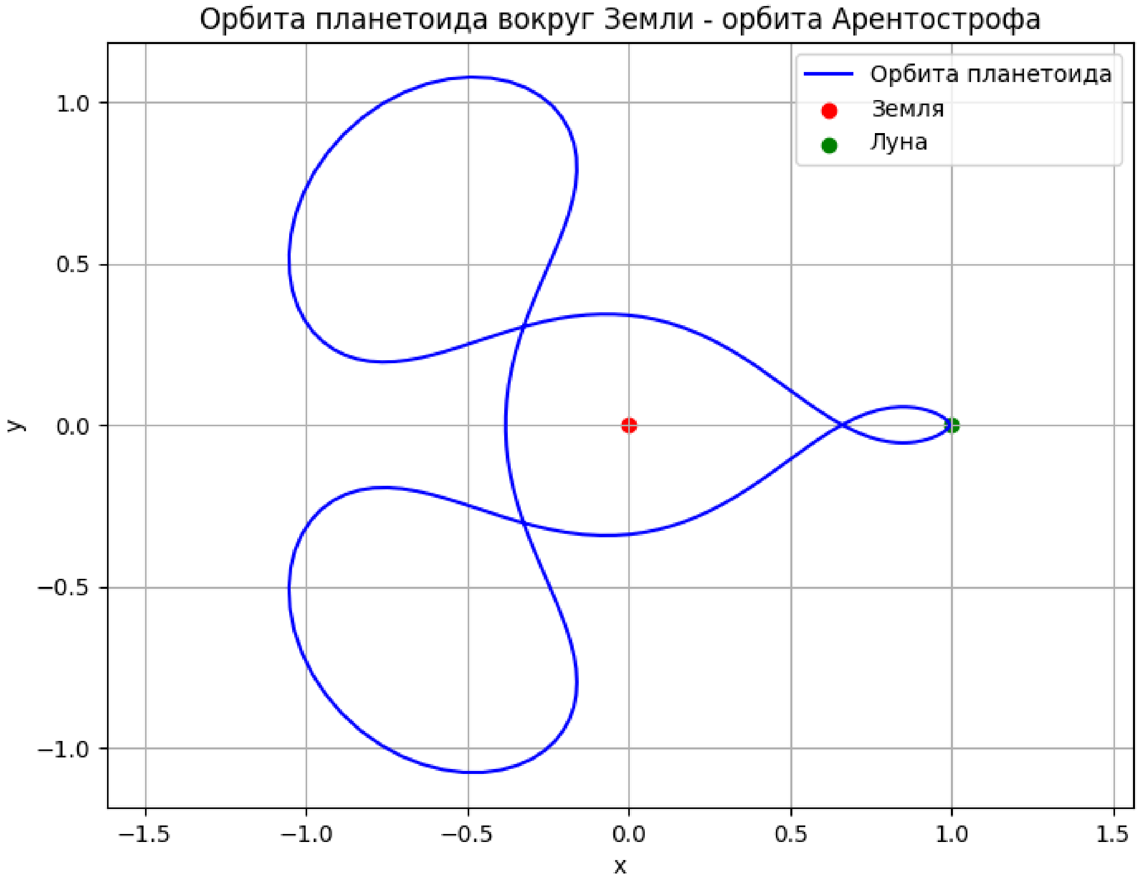

Рис.4 – график орбиты Аренсторфа

График описывает орбиту планетоида, который движется под влиянием гравитации Земли и Луны. Синяя линия показывает траекторию движения планетоида. Орбита имеет сложную форму и включает в себя несколько замкнутых петель. Данная орбита характеризуются тем, что планетоид периодически приближается и удаляется от Земли и Луны.

На графике видно три основные петли орбиты планетоида: две крупные петли выше и ниже оси x, и одна маленькая петля справа от Земли и Луны. Петли показывают, что планетоид проходит близко к Земле и Луне, а затем удаляется от них на большие расстояния. Когда планетоид находится на петлях, он испытывает сильное гравитационное влияние как Земли, так и Луны.

Движение планетоида описывается периодическим характером: он возвращается в определенные точки своей орбиты спустя фиксированные промежутки времени. Орбита не является стабильной и регулярной.

Вывод

В ходе выполнении курсовой работы, была решена тестовая задача и реализована программа интегрирования задачи Коши с помощью Рунге- Кутты 4-го порядка точности с постоянным шагом h для системы, состоящей из n обыкновенных дифференциальных уравнений. Были построены графики зависимости максимальной погрешности решения e и e/h^4 от выбранного шага h. По графикам можно сделать вывод, что максимальная погрешность решения e возрастает при увеличении шага h. Был построен график орбиты Аренсторфа.

Список литературы

1. Даутов Р. З. Практикум по дисциплине “Численные методы”. Решение задачи Коши для системы ОДУ. - Учебное пособие, Казань, 2014.

2. Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. - М.: Мир, 1990.

3. Самарский А.А., Гулин А.В. Численные методы. - М.: Наука, 1989.

4. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. - М.: Наука, 1987.

Листинг

```python
import numpy as np
import matplotlib.pyplot as plt

def f(y1, y2):
    dy1 = -y2 + y1 * (y1**2 + y2**2 - 1)
    dy2 = y1 + y2 * (y1**2 + y2**2 - 1)
    return dy1, dy2

def rk4_step(y1, y2, h):
    k1_y1, k1_y2 = f(y1, y2)
    k2_y1, k2_y2 = f(y1 + 0.5 * h * k1_y1, y2 + 0.5 * h * k1_y2)
    k3_y1, k3_y2 = f(y1 + 0.5 * h * k2_y1, y2 + 0.5 * h * k2_y2)
    k4_y1, k4_y2 = f(y1 + h * k3_y1, y2 + h * k3_y2)
    y1_new = y1 + (h / 6) * (k1_y1 + 2 * k2_y1 + 2 * k3_y1 + k4_y1)
    y2_new = y2 + (h / 6) * (k1_y2 + 2 * k2_y2 + 2 * k3_y2 + k4_y2)
    return y1_new, y2_new

def exact_solution(x):
    y1_exact = np.cos(x) / (np.sqrt(1 + 2.71828182846**(2*x)))
    y2_exact = np.sin(x) / (np.sqrt(1 + 2.71828182846**(2*x)))
    return y1_exact, y2_exact

y1_0 = 1 / np.sqrt(2)
y2_0 = 0
t0 = 0
t_end = 5
h = 0.01


t_values = np.arange(t0, t_end, h)
y1_values = np.zeros_like(t_values)
y2_values = np.zeros_like(t_values)

y1 = y1_0
y2 = y2_0
for i, t in enumerate(t_values):
    y1_values[i] = y1
    y2_values[i] = y2
    y1, y2 = rk4_step(y1, y2, h)

y1_exact, y2_exact = exact_solution(t_values)
```

```

plt.figure(figsize=(16, 7))

plt.subplot(1, 2, 1)
plt.plot(t_values, y1_values, label='Численное решение y1')
plt.plot(t_values, y2_values, label='Численное решение y2')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Решение системыметодомРунгеКутты  - го4- порядка')
plt.legend()
plt.grid(True)
plt.subplot(1, 2, 2)
plt.plot(t_values, y1_exact, label='Точное решение y1', linestyle='--', color='red')
plt.plot(t_values, y2_exact, label='Точное решение y2', linestyle='--')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Точное решениесистемы ')
plt.legend()
plt.grid(True)
plt.savefig('plot_test_task_exact_and_method.png')
```

```
import numpy as np
import matplotlib.pyplot as plt

def f(y1, y2):
    dy1 = -y2 + y1 * (y1**2 + y2**2 - 1)
    dy2 = y1 + y2 * (y1**2 + y2**2 - 1)
    return dy1, dy2

def rk4_step(y1, y2, h):
    k1_y1, k1_y2 = f(y1, y2)
    k2_y1, k2_y2 = f(y1 + 0.5 * h * k1_y1, y2 + 0.5 * h * k1_y2)
    k3_y1, k3_y2 = f(y1 + 0.5 * h * k2_y1, y2 + 0.5 * h * k2_y2)
    k4_y1, k4_y2 = f(y1 + h * k3_y1, y2 + h * k3_y2)
    y1_new = y1 + (h / 6) * (k1_y1 + 2 * k2_y1 + 2 * k3_y1 + k4_y1)
    y2_new = y2 + (h / 6) * (k1_y2 + 2 * k2_y2 + 2 * k3_y2 + k4_y2)
    return y1_new, y2_new

def exact_solution(x):
    y1_exact = np.cos(x) / (np.sqrt(1 + 2.71828182846**(2*x)))
    y2_exact = np.sin(x) / (np.sqrt(1 + 2.71828182846**(2*x)))
    return y1_exact, y2_exact
```

```

def max_error(y_num, y_exact):
    return np.max(np.abs(y_num - y_exact))

def solve_system_e(h_values):
    max_errors = []
    for h in h_values:
        t_values = np.arange(t0, t_end, h)
        y1_values = np.zeros_like(t_values)
        y2_values = np.zeros_like(t_values)
        y1 = y1_0
        y2 = y2_0
        for i, t in enumerate(t_values):
            y1_values[i] = y1
            y2_values[i] = y2
            y1, y2 = rk4_step(y1, y2, h)
        y1_exact, y2_exact = exact_solution(t_values)
        max_errors.append(max(max_error(y1_values, y1_exact), max_error(y2_values, y2_exact)))
    return max_errors

def solve_system_eh4(h_values):
    max_errors = []
    for h in h_values:
        t_values = np.arange(t0, t_end, h)
        y1_values = np.zeros_like(t_values)
        y2_values = np.zeros_like(t_values)
        y1 = y1_0
        y2 = y2_0
        for i, t in enumerate(t_values):
            y1_values[i] = y1
            y2_values[i] = y2
            y1, y2 = rk4_step(y1, y2, h)
        y1_exact, y2_exact = exact_solution(t_values)
        max_errors.append(max(max_error(y1_values, y1_exact), max_error(y2_values, y2_exact))/h**4)
    return max_errors

y1_0 = 1 / np.sqrt(2)
y2_0 = 0
t0 = 0
t_end = 5
h_values = np.linspace(0.001, 0.1, 100)
```

```
max_errors_e = solve_system_e(h_values)
max_errors_eh4 = solve_system_eh4(h_values)

plt.figure(figsize=(16, 7))

plt.subplot(1, 2, 1)
plt.plot(h_values, max_errors_e)
plt.xlabel('Шаг интегрирования, h')
plt.ylabel('Максимальная погрешность')
plt.title('Зависимость максимальнойпогрешности e отвыбранногошага h')
plt.grid(True)
plt.subplot(1, 2, 2)
plt.plot(h_values, max_errors_eh4)
plt.xlabel('Шаг интегрирования, h')
plt.ylabel('Максимальная погрешность')
plt.title('Зависимость максимальнойпогрешности e/h^4 отвыбранногошага h')
plt.grid(True)
plt.savefig('plot_max_err.png')
plt.show()
```

```
import numpy as np
import matplotlib.pyplot as plt

def aren(t, u):
    x, y, vx, vy = u
    m = 0.012277471; M = 1 - m
    r1 = ((x+m)**2+y**2)**(1.5);
    r2 = ((x-M)**2+y**2)**(1.5)
    return np.array([vx, vy,x+2*vy-M*(x+m)/r1-m*(x-M)/r2,y-2*vx-M* y /r1-m* y /r2])

aren_init = np.array([0.994, 0, 0, -2.031732629557337])
aren_tmax = 11.124340337

def rk4(f, tau, t, u):
    k1 = f(t, u)
    k2 = f(t + tau/2, u + tau/2*k1)
    k3 = f(t + tau/2, u + tau/2*k2)
    k4 = f(t + tau , u + tau *k3)
    return u + tau * (k1 + 2*k2 + 2*k3 + k4) / 6

```

```python
def adaptive_stepsize(f, y0, tmax, method, tol, tau=0.01):
    t = 0; u = y0
    T = [0]; Y = [y0]
    failed = 0 # Числонеудачныхшагов
    while t < tmax:
        if t + tau > tmax: tau = tmax - t
        u1 = method(f, tau, t, u) # Целыйшаг
        u2 = method(f, tau/2, t, u)
        u2 = method(f, tau/2, t+tau/2, u2) # Дваполушага
        err = np.linalg.norm(u1-u2)/(1-2**-4) # ПравилоРунге
        fac = (tol/err)**(1 / (4+1)) # Подстраиваем tau
        taunew = tau * min(2, max(0.25, 0.8 * fac))
        if err < tol: # Ошибкамала, принимаемшаг
            t += tau; u = u1
            T.append(t); Y.append(u)
        else: # Еслиошибкавелика, повторяемшагсновым tau
            failed += 1
        tau = taunew
    return np.array(T), np.array(Y)

T, Y = adaptive_stepsize(aren, aren_init, aren_tmax, rk4, 1e-6)
x_values = Y[:, 0]
y_values = Y[:, 1]

# Построениеорбиты
plt.figure(figsize=(8, 6))
plt.plot(x_values, y_values, label='Орбита планетоида', color='blue')
plt.scatter([0], [0], color='red', label='Земля')
plt.scatter([1], [0], color='green', label='Луна')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Орбита планетоидавокругЗемли - орбитаАрентострофа ')
plt.legend()
plt.grid(True)
plt.axis('equal')
plt.savefig('plot_main_task.png')
plt.show()
```