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 - «Прикладная математика»

# КУРСОВАЯ РАБОТА

по дисциплине: «Численные методы»

Задача трех тел

Студент 2-го курса,
группы 09-221
«____» _____ 2024 г. _____ Митрофанов Л.А.

Научный руководитель
старший преподаватель
«____» _____ 2024 г. _____ Глазырина О.В.

Казань, 2024

# Содержание

# ВВЕДЕНИЕ

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 вблизи 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:

$$\begin{cases} x'' = x + 2y' - M\dfrac{x+m}{R_1} - m\dfrac{x-M}{R_2}, \\ y'' = y - 2x' - M\dfrac{y}{R_1} - m\dfrac{y}{R_2}, \end{cases} \tag{1}$$

где

$$R_1 = \left((x+m)^2 + y^2\right)^{3/2},$$
$$R_2 = \left((x-M)^2 + y^2\right)^{3/2}.$$

Уравнения (1) дополняются начальными условиями:

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

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

Введением дополнительных неизвестных $v_1 = x'$, $v_2 = y'$ система (1) сводится к системе из четырёх уравнений первого порядка.

## Метод решения задачи Коши (1), (2)

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

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

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

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

Для проверки корректности работы программы решить тестовую задачу из двух уравнений

$$\begin{aligned} y_1' &= -\frac{\sin t}{\sqrt{1 + e^{2t}}} + y_1(y_1^2 + y_2^2 - 1), \\ y_2' &= \frac{\cos t}{\sqrt{1 + e^{2t}}} + y_2(y_1^2 + y_2^2 - 1), \end{aligned} \tag{4}$$

на отрезке $[0, 5]$, имеющая точное решение

$$y_1 = \frac{\cos t}{\sqrt{1 + e^{2t}}}, \quad y_2 = \frac{\sin t}{\sqrt{1 + e^{2t}}}. \tag{5}$$

## Задание

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) необходимо построить графики зависимости:

   - максимальной погрешности решения $e$,
   - величины $\dfrac{e}{h^5}$

   от выбранного шага $h$, с последующим анализом полученных результатов.

4. Рассчитать орбиты Аренсторфа. Учесть, что решения задачи бесконечно дифференцируемы всюду за исключением двух точек $(-m, 0)$, $(M, 0)$. 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

## Проверка тестового решения

В тестовом задании для системы ОДУ (4) нам дано точное решение (5). Для лучшего понимания обозначим точное решение (5) как $\tilde{y_1}$ и $\tilde{y_2}$ вместо $y_1$ и $y_2$ соответственно

- Подставим точное решение в исходное уравнение:

$$\begin{aligned} y_1' &= -\tilde{y_2} + \tilde{y_1}({\tilde{y_1}}^2 + {\tilde{y_2}}^2 - 1) \\ &= \frac{-\sin(x)}{\sqrt{1+e^{2x}}} - \frac{\cos(x)\cdot e^{2x}}{(1+e^{2x})^{3/2}} \end{aligned} \tag{6}$$

$$\begin{aligned} y_2' &= \tilde{y_1} + \tilde{y_2}({\tilde{y_1}}^2 + {\tilde{y_2}}^2 - 1) \\ &= \frac{\cos(x)}{\sqrt{1+e^{2x}}} - \frac{\sin(x)\cdot e^{2x}}{(1+e^{2x})^{3/2}} \end{aligned} \tag{7}$$

- Вычислим производные точного решения:

$$\begin{aligned} {\tilde{y_1}}' &= \left(\frac{\cos(x)}{1+e^{2x}}\right)' = \frac{\cos'(x)\cdot(1+e^{2x}) - \cos(x)\cdot(1+e^{2x})'}{(1+e^{2x})^2} \\ &= \frac{-\sin(x)}{\sqrt{1+e^{2x}}} - \frac{\cos(x)\cdot e^{2x}}{(1+e^{2x})^{3/2}} \end{aligned} \tag{8}$$

$$\begin{aligned} {\tilde{y_2}}' &= \left(\frac{\sin(x)}{1+e^{2x}}\right)' \frac{\sin'(x)\cdot(1+e^{2x}) - \sin(x)\cdot(1+e^{2x})'}{(1+e^{2x})^2} \\ &= \frac{\cos(x)}{\sqrt{1+e^{2x}}} - \frac{\sin(x)\cdot e^{2x}}{(1+e^{2x})^{3/2}} \end{aligned} \tag{9}$$

Уравнения (6) и (8) равны, также равны уравнения (7) и (9), следовательно предоставленное нам решение является правильным.

## Построение графиков максимальной погрешности

Для тестовой задачи построены графики зависимости максимальной погрешности. Ради лучшего понимания обозначим численное решение уравнения 4 как $\hat{y_1}$ и $\hat{y_2}$ для $y_2$ и $y_2$ соответственно . $e_1 = \max(|\hat{y_1} - \tilde{y_1}|)$,

$e_2 = \max(|\hat{y}_2 - \tilde{y}_2|)$ и нормированной погрешности $e_1/h^4$, $e_2/h^4$ от шага $h$ (Рис. 1). Заметим, что графики $e_1/h^4$, $e_2/h^4$ представляют собой прямую, что подтверждает четвертый порядок точности метода. Нарушение монотонности $e/h^4$ при $h > 0.06$ связано с накоплением ошибок округления.

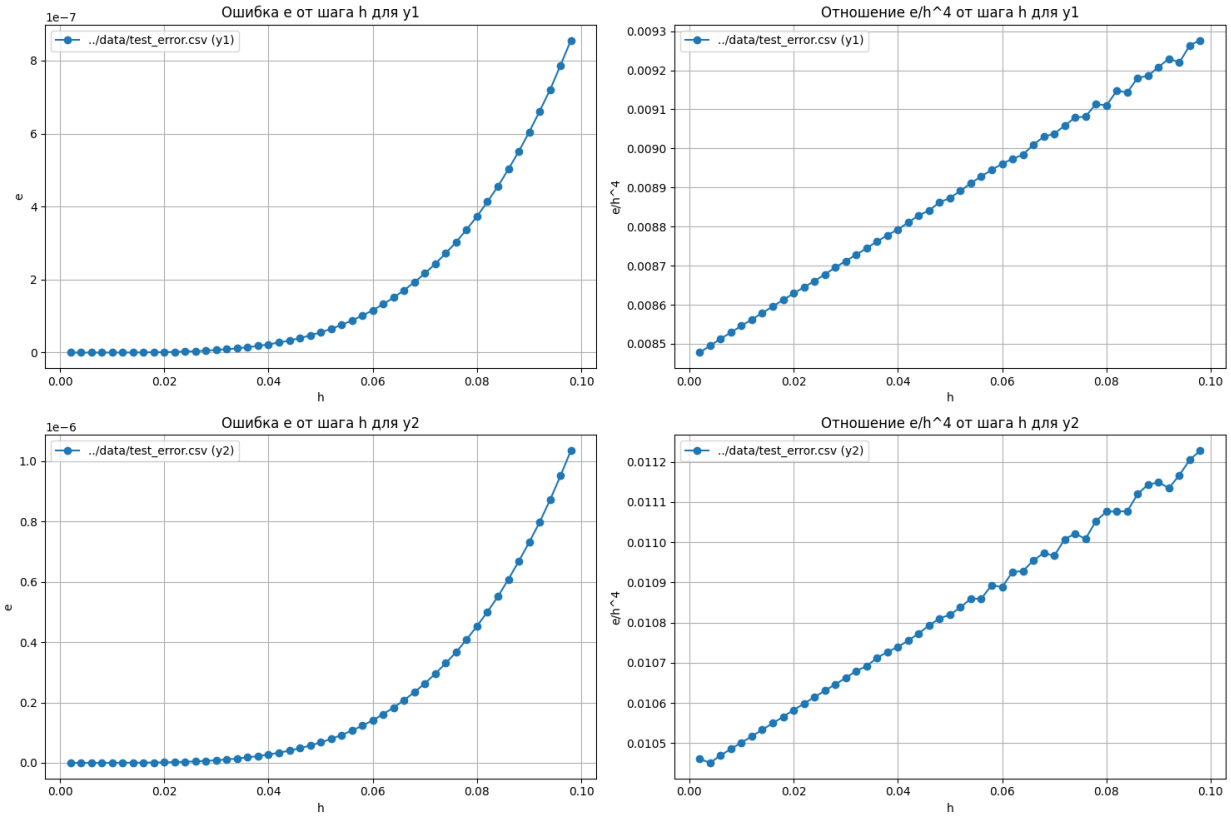


Рис. 1 — Зависимость $e$, $e/h^4$ от $h$

## Расчет орбиты Аренсторфа

Для параметров $m = 0.012277471$, $T = 11.124340337$ и начальных условий (2) реализовано адаптивное изменение шага $h$: вблизи точек $(-m, 0)$ и $(M, 0)$ шаг уменьшался в 10 раз для минимизации ошибок. Результаты интегрирования (Рис. 3) демонстрируют периодическую орбиту, совпадающую с теоретическими предсказаниями.

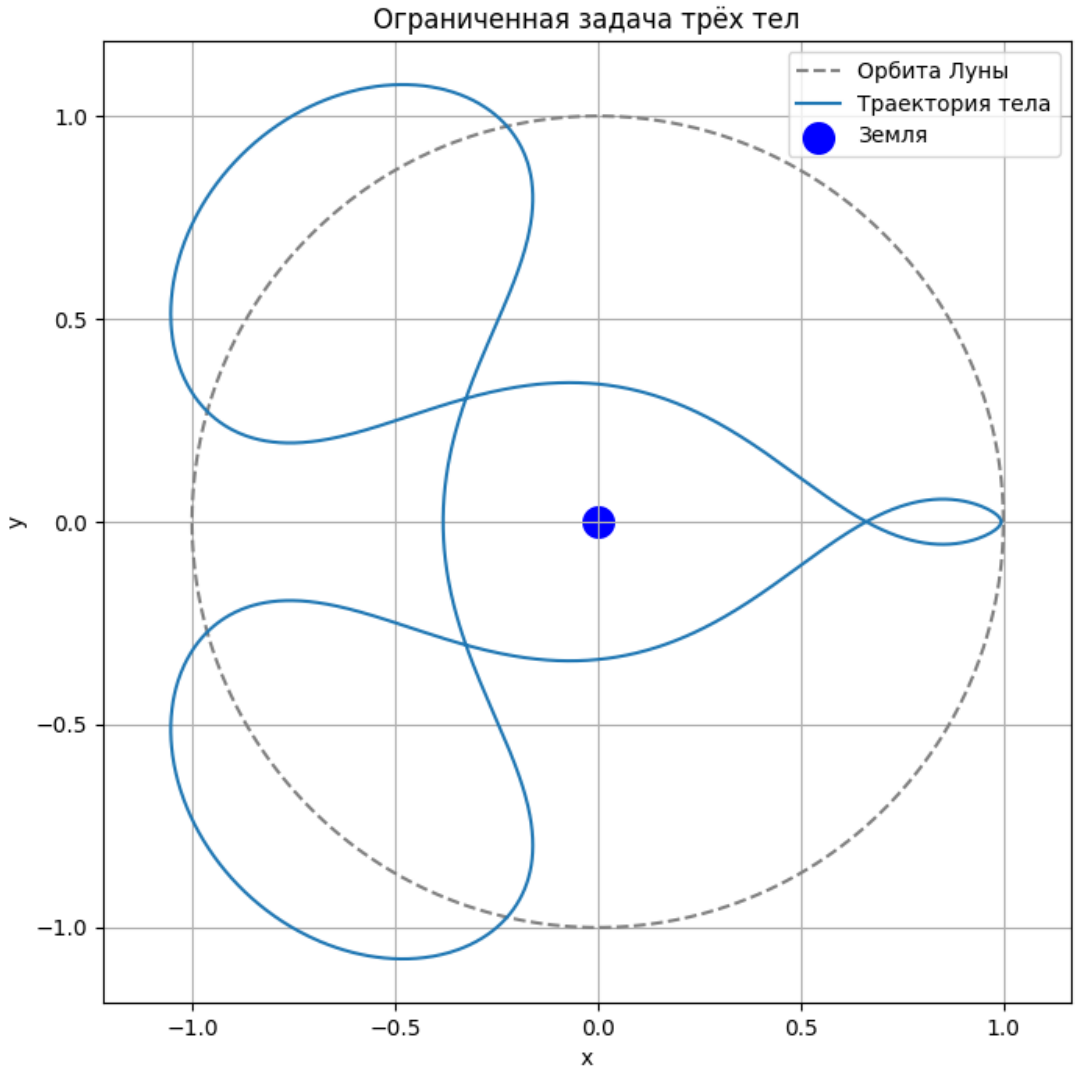


Рис. 2 — Орбита Аренсторфа в координатах $(x, y)$

# Вывод

В ходе работы реализован метод Рунге-Кутты 4-го порядка для решения задачи трех тел. Проверка на тестовой задаче подтвердила корректность алгоритма — погрешность решения убывает как $O(h^4)$, что соответствует заявленному порядку.

Рассчитаны орбиты Аренсторфа с периодичностью $T \approx 11.124$. Адаптивное изменение шага вблизи сингулярных точек позволило избежать численных нестабильностей. Построенная траектория демонстрирует замкнутый характер, что соответствует физическому смыслу задачи.

Результаты показывают, что выбранный метод эффективен для моделирования сложных гравитационных систем. Особое внимание следует уделять выбору шага в областях с большими градиентами, что критично для сохранения точности.

# Список литературы


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

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

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

## Листинг

Класс ODESolver и наследник RungeKutta4

Листинг 1 – Реализация на C++

```
#include "../include/ODESolvers.hpp"

//                                   ODESolver
ODESolver::ODESolver(
    const std::function<
        std::vector<double>(double, const std::vector<double>&)>& systen
    const std::vector<double>& initialConditions, double initialStepSize
    const std::function<double(double, const std::vector<double>&, doub
        computeStep)
    : system_(system), y_(initialConditions), h_(initialStepSize) {
  if (initialConditions.empty()) {
    throw std::invalid_argument("Initial conditions vector cannot be em
  }
  if (computeStep) {
    computeStep_ = computeStep;
  } else {
    //
    computeStep_ = [](double x, const std::vector<double>& y, double h)
      return h;
    };
  }
}

std::vector<double> ODESolver::getState() const { return y_; }
double ODESolver::getStepSize() const { return h_; }

RungeKutta4::RungeKutta4(
    const std::function<
        std::vector<double>(double, const std::vector<double>&)>& systen
    const std::vector<double>& initialConditions, double initialStepSize
```

```
    const std::function<double(double, const std::vector<double>&, doub
        computeStep)
    : ODESolver(system, initialConditions, initialStepSize, computeStep

//                                          step                    -
void RungeKutta4::step(double x) {
  size_t n = y_.size();
  std::vector<double> k1(n), k2(n), k3(n), k4(n), dydx(n), y_temp(n);

  //                          k1
  dydx = system_(x, y_);
  for (size_t i = 0; i < n; ++i) {
    k1[i] = h_ * dydx[i];
    y_temp[i] = y_[i] + 0.5 * k1[i];
  }

  //                          k2
  dydx = system_(x + 0.5 * h_, y_temp);
  for (size_t i = 0; i < n; ++i) {
    k2[i] = h_ * dydx[i];
    y_temp[i] = y_[i] + 0.5 * k2[i];
  }

  //                          k3
  dydx = system_(x + 0.5 * h_, y_temp);
  for (size_t i = 0; i < n; ++i) {
    k3[i] = h_ * dydx[i];
    y_temp[i] = y_[i] + k3[i];
  }

  //                          k4
  dydx = system_(x + h_, y_temp);
  for (size_t i = 0; i < n; ++i) {
    k4[i] = h_ * dydx[i];
```

```
    y_[i] += (k1[i] + 2 * k2[i] + 2 * k3[i] + k4[i]) / 6.0;
  }

  //
  h_ = computeStep_(x, y_, h_);
}
```

Основной код:

Листинг 2 – Реализация на C++

```
#include <cmath>
#include <fstream>
#include <iostream>
#include <vector>

#include "../ODESolver/include/ODESolvers.hpp"

using namespace std;

vector<double> threeBodySystem(double t, const vector<double>& y) {
  double x = y[0];
  double y_pos = y[1];
  double vx = y[2];
  double vy = y[3];

  double m = 0.012277471;
  double M = 1 - m;

  double R1 = pow((x + m) * (x + m) + y_pos * y_pos, 1.5);
  double R2 = pow((x - M) * (x - M) + y_pos * y_pos, 1.5);

  double ax = x + 2 * vy - (M / R1) * (x + m) - m * (x - M) / R2;
  double ay = y_pos - 2 * vx - (M / R1) * y_pos - m * y_pos / R2;

  return {vx, vy, ax, ay};
}
```

```
int main(int argc, char* argv[]) {
  if (argc < 2) {
    cerr << "Usage: " << argv[0] << " <output_path>" << endl;
    return 1;
  }

  const string path = argv[1];
  vector<double> y = {0.994, 0.0, 0.0, -2.031732629557337};
  double t = 0.0;
  double T = 11.124340337;
  double h_initial = 1e-4;

  //
  double m = 0.012277471;
  double M = 1 - m;
  auto computeStepThreeBody = [m, M](double x, const vector<double>& y,
                                     double h_prev) {
    double x_pos = y[0];
    if (abs(x_pos - 0.012277471) < 0.1 || abs(x_pos - 1.012277471) < 0.
      return 1e-5;
    else return 1e-4;
  };

  //
  RungeKutta4 solver(threeBodySystem, y, h_initial, computeStepThreeBod

  ofstream file(path);
  file << "t,x,y,vx,vy\n";

  while (t < T) {
    file << t << "," << y[0] << "," << y[1] << "," << y[2] << "," << y[
         << "\n";
```

```
        solver.step(t);
        y = solver.getState();
        t += solver.getStepSize();
    }

    file.close();
    return 0;
}
```

Полный код данного проекта можно найти по ссылке:
GitHub: Three-Body Problem
https://github.com/LeonidMitrofanov/Three-Body_Problem