Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №4
Задание №1 по курсу «Численные методы»
Вариант №12

Студент:	С. М. Власова
Преподаватель:	И. Э. Иванов
Группа:	М8О-306Б
Дата:	
Оценка:	
Подпись:	

Москва, 2020

Задание №4.1

Реализовать методы Эйлера, Рунге-Кутты и Адамса 4-го порядка в виде программ, задавая в качестве входных данных шаг сетки h. С использованием разработанного программного обеспечения решить задачу Коши для ОДУ 2-го порядка на указанном отрезке. Оценить погрешность численного решения с использованием метода Рунге-Ромберга и путем сравнения с точным решением.

Вариант: 12

Задача Коши:

$$\begin{cases} (x^2+1)\cdot y'' - 2x\cdot y' + 2y = 0, \\ y(0) = 1, \\ y'(0) = 1, \\ x \in [0,1], h = 0.1. \end{cases} \tag{1}$$

Точное решение: $y = x - x^2 + 1$.

1 Описание метода решения

1 ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

1.1 ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧИ КОШИ

Задача Коши для обыкновенного дифференциального уравнения.

Рассматривается задача Коши для одного дифференциального уравнения первого порядка, разрешенного относительно производной

$$\begin{cases} y' = f(x, y) \\ y(x_0) = y_0 \end{cases} \tag{2}$$

Требуется найти решение на отрезке $[a, b]$, где $x_0 = a$.
Введем разностную сетку на отрезке $[a, b]: \ \Omega^{(k)} = \{x_k = x_0 + hk\}, \ k = 0, 1, \ldots, N, \quad h = |b-a|/N$. Точки x_k называются *узлами* разностной сетки, расстояния между узлами — *шагом* разностной сетки (h), а совокупность значений какой-либо величины, заданных в узлах сетки называется *сеточной функцией* $y^{(h)} = \{y_k, k = 0, 1, \ldots, N\}$.
Приближенное решение задачи Коши будем искать *численно* в виде сеточной функции $y^{(h)}$. Для оценки погрешности приближенного численного решения $y^{(h)}$ будем рассматривать это решение как элемент $N+1-$ мерного линейного векторного пространства с какой-либо нормой.В качестве погрешности решения принимается норма элемента этого пространства $\delta^{(h)} = y^{(h)} - [y]^{(h)}$, где $[y]^{(h)}$ — точное решение задачи Коши в узлах расчетной сетки. Таким образом, $\varepsilon_h = ||\delta^{(h)}||$.

Одношаговые методы.
Метод Эйлера (явный).

Метод Эйлера играет важную роль в теории численных методов решения ОДУ, хотя и не часто используется в практических расчетах из-за невысокой точности. Вывод расчетных соотношений для этого метода может быть произведен несколькими способами: с помощью геометрической интерпретации, с использованием разложения в ряд Тейлора, конечно разностным методом (с помощью разностной аппроксимации производной), квадратурным способом (использованием эквивалентного интегрального уравнения).

Рассмотрим вывод соотношений метода Эйлера геометрическим способом. Решение в узле x_0 известно из начальных условий, рассмотрим процедуру получения решения в узле x_1 рис.4.1.

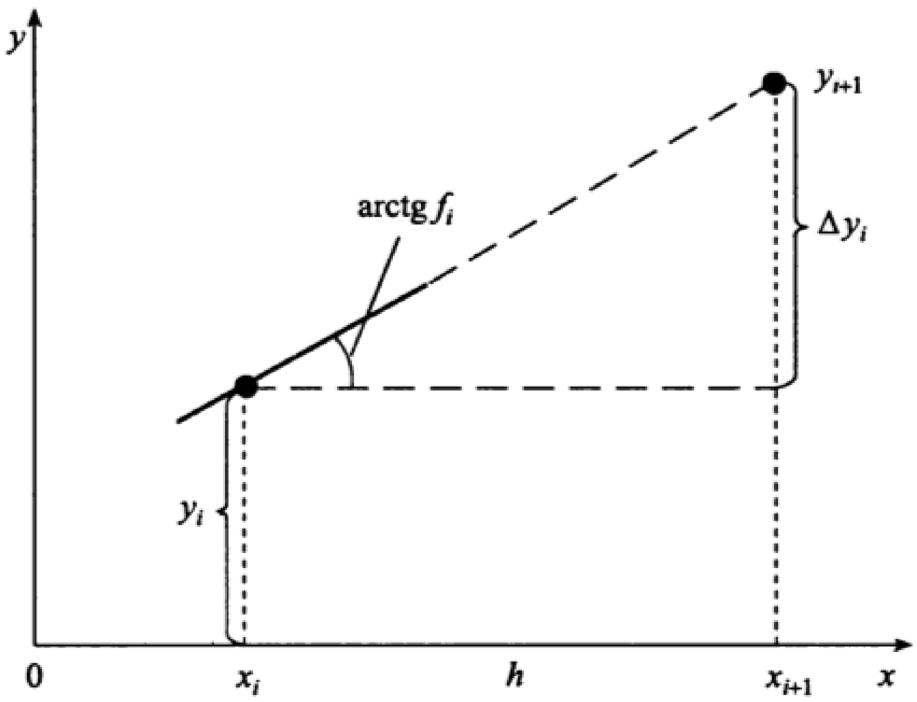

Рис. 4.1. Геометрический смысл метода Эйлера

График функции $y^{(h)}$, которая является решением задачи Коши, представляет собой гладкую кривую, проходящую через точку (x_0, y_0) согласно условию

$$y(x_0) = y_0,$$

и имеет в этой точке касательную. Тангенс угла наклона касательной к оси O_x равен значению производной от решения в точке x_0 и равен значению правой части дифференциального уравнения в точке (x_0, y_0) согласно выражению

$$y'(x_0) = f(x_0, y_0).$$

В случае небольшого шага разностной сетки h график функции и график касательной не успевают сильно разойтись друг от друга, и можно в качестве значения решения в узле x_1 принять значение касательной y_1, вместо значения неизвестного точного решения y_1. При этом допускается погрешность $y_1 - y_{1.}$.
Из прямоугольного треугольника (рис.4.1) находим

$$\Delta y = h \cdot y'(x_0).$$

Учитывая, что $\Delta y = y_1 - y_0$ и заменяя производную $y'(x_0)$ на правую часть дифференциального уравнения, получаем соотношение

$$y_1 = y_0 + h \cdot f(x_0, y_0).$$

Считая теперь точку (x_1, y_1) начальной и повторяя все предыдущие рассуждения, получим значение y_2 в узле x_2.

Переход к произвольным индексам дает **формулу Эйлера:**

$$y_{k+1} = y_k + h \cdot f(x_k, y_k).$$

Погрешность метода Эйлера.

На каждом шаге метода Эйлера допускается *локальная* погрешность по отношению к точному решению, график которого проходит через крайнюю левую точку отрезка. Кроме того, на каждом шаге, начиная со второго, накапливается *глобальная* погрешность, представляющая собой разность между численным решением и точным решением исходной начальной задачи (а не локальной).
Локальная ошибка на каждом шаге выражается соотношением

$$\varepsilon_k^h = \frac{y''(\xi)}{2} \cdot h^2,$$

где $\xi \in [x_{k-1}, x_k]$. Глобальная погрешность метода Эйлера $\varepsilon^h = C \cdot h$ в окрестности $h = 0$ ведет себя, как линейная функция и, следовательно, метод Эйлера имеет первый порядок точности относительно шага h.

Модификации метода Эйлера.
Неявный метод Эйлера.

Если на правой границе интервала использовать точное значение производной от решения (т.е. тангенса угла наклона касательной), то получается неявный метод Эйлера первого порядка точности.

$$y_{k+1} = y_k + h \cdot f(x_{k+1}, y_{k+1}).$$

В общем случае нелинейное относительно y_{k+1} уравнение численно решается с помощью одного из методов раздела 2, например, методом Ньютона или его модификациями.

Метод Эйлера-Коши.

В данном методе на каждом интервале расчет проводится в два этапа. На первом (этап прогноза) определяется приближенное решение на правом конце интервала по методу Эйлера, на втором (этап коррекции) уточняется значение решения на правом конце с использованием полусуммы тангенсов углов наклона на концах интервала

$$\widetilde{y}_{k+1} = y_k + h \cdot f(x_k, y_k),$$

$$y_{k+1} = y_k + h \cdot \frac{f(x_k, y_k) + f(x_{k+1}, \widetilde{y}_{k+1})}{2},$$

$$x_{k+1} = x_k + h.$$

Этот метод имеет второй порядок точности.

Неявный метод Эйлера-Коши.

Если на правой границе интервала использовать точное значение производной к решению (т.е. тангенса угла наклона касательной), то получается неявный метод Эйлера-Коши (метод трапеций) второго порядка точности.

$$y_{k+1} = y_k + h \cdot \frac{f(x_k, y_k) + f(x_{k+1}, y_{k+1})}{2},$$

$$x_{k+1} = x_k + h.$$

Метод Эйлера-Коши с итерационной обработкой.

Комбинация предыдущих модификационных методов дает метод формально второго порядка точности, но более точного в смысле абсолютной величины погрешности приближенного решения, чем исходные методы.

$$y_{k+1}^{(0)} = y_k + h \cdot f(x_k, y_k),$$

$$y_{k+1}^{(i)} = y_k + h \cdot \frac{f(x_k, y_k) + f(x_{k+1}, y_{k+1}^{(i-1)})}{2},$$

$$x_{k+1} = x_k + h.$$

В формуле правые верхние индексы в круглых скобках обозначают номер итерации, при этом начальное приближение $y_{k+1}^{(0)}$ определяется по методу Эйлера. Метод Эйлера-Коши с итерационной обработкой представляет собой реализацию метода простой итерации для решения нелинейного уравнения в неявном методе Эйлера. Выполнять простые итерации до полной сходимости нет смысла, поэтому рекомендуется выполнять 3-4 итерации.

Первый улучшенный метод Эйлера.

Данный метод использует расчет приближенного значения производной от решения в точке на середине расчетного интервала. Значение производной в середине получают применением явного метода Эйлера на половинном шаге по x.

$$y_{k+1/2} = y_k + \frac{h}{2} \cdot f(x_k, y_k),$$

$$y_{k+1} = y_k + h \cdot f(x_{k+1/2}, y_{k+1/2}),$$
$$x_{k+1} = x_k + h,$$
$$x_{k+1/2} = x_k + \frac{h}{2}.$$

Данная модификация Эйлера имеет второй порядок точности.

Методы Рунге-Кутты.

Все рассмотренные выше явные методы являются вариантами методом Рунге-Кутты. Семейство явных методов Рунге-Кутты $p-$го порядка записывается в виде совокупности формул:

$$y_{k+1} = y_k + \Delta y_k,$$

$$\Delta y_k = \sum_{i=1}^{p} c_i K_i^k,$$

$$K_i^k = h \cdot f(x_k + a_i \cdot h, y_k + h \cdot \sum_{j=1}^{i-1} b_{ij} K_j^k), \quad i = 2, 3, \ldots, p.$$

Параметры a_i, b_{ij}, c_i подбираются так, чтобы значение y_{k+1} совпадало со значением разложения в точке x_{k+1} точного решения в ряд Тейлора с погрешностью $O(h^{p+1})$.

Метод Рунге-Кутты третьего порядка точности.

Один из методов Рунге-Кутты третьего порядка ($p = 3, a_1 = 0, a_2 = \frac{1}{3}, a_3 = \frac{2}{3}, b_{21} = \frac{1}{3}, b_{31} = 0, b_{32} = \frac{2}{3}, c_1 = \frac{1}{4}, c_2 = 0, , c_3 = \frac{3}{4}$) имеет вид:

$$y_{k+1} = y_k + \Delta y_k,$$

$$\Delta y_k = \frac{1}{4}(K_1^k + 3 \cdot K_3^k),$$

$$K_1^k = h \cdot f(x_k, y_k),$$

$$K_2^k = h \cdot f(x_k + \frac{1}{3}h, y_k + \frac{1}{3}K_1^k),$$

$$K_3^k = h \cdot f(x_k + \frac{2}{3}h, y_k + \frac{2}{3}K_2^k).$$

Метод Рунге-Кутты четвертого порядка точности.

Методов Рунге-Кутты четвертого порядка ($p = 4, a_1 = 0, a_2 = \frac{1}{2}, a_3 = \frac{1}{2}, a_4 = 1, b_{21} = \frac{1}{2}, b_{31} = 0, b_{32} = \frac{1}{2}, b_{41} = 0, b_{42} = 0, b_{43} = \frac{1}{2}, c_1 = \frac{1}{6}, c_2 = \frac{1}{3}, c_3 = \frac{1}{3}, c_4 = \frac{1}{6}$) является одним из самых широко используемых методов решения задачи Коши:

$$y_{k+1} = y_k + \Delta y_k,$$

$$\Delta y_k = \frac{1}{6}(K_1^k + 2 \cdot K_2^k + 2 \cdot K_3^k + K_4^k),$$

$$K_1^k = h \cdot f(x_k, y_k),$$

$$K_2^k = h \cdot f(x_k + \frac{1}{2}h, y_k + \frac{1}{2}K_1^k),$$

$$K_3^k = h \cdot f(x_k + \frac{1}{2}h, y_k + \frac{1}{2}K_2^k).K_4^k = h \cdot f(x_k + h, y_k + K_3^k).$$

Контроль точности на каждом шаге h.

Основным способом контроля точности получаемого численного решения при решении задачи Коши является методы основанные на принципе Рунге-Ромберга-Ричардсона. Пусть y^h решение задачи Коши, полученное методом Рунге-Кутты $p-$го порядка точности с шагом h в точке $x + 2h$. Пусть y^{2h} решение той же задачи в точке $x + 2h$, полученное тем же методом, но с шагом $2h$. Тогда выражение

$$\tilde{y} = y^h + \frac{y^h - y^{2h}}{2^p - 1}$$

аппроксимирует точное решение в точке $x + 2h$ $y(x + 2h)$ с $p + 1-$м порядком. Второе слагаемое в этом выражении оценивает главный член в погрешности решения y^h, то есть

$$R^h = \frac{y^h - y^{2h}}{2^p - 1}.$$

Контроль точности может быть организован следующим образом. Выбирается значение шага h и дважды рассчитывается решение в точке $x + 2h$ — один раз с шагом h, другой раз с шагом $2h$. Рассчитывается величина R^h и сравнивается с заданной точностью ε. Если величина R^h меньше заданной точности, то можно продолжить вычисления с тем же шагом, в противном случае необходимо вернуться к решению в точке x, уменьшить шаг h и повторить вычисления.

Вычислительная стоимость такого контроля точности достаточно велика, особенно для многостадийных методов, поэтому можно использовать более грубый способ контроля правильности выбора шага h. В случае метода Рунге-Кутты четвертого порядка точности следует на каждом шаге h рассчитывать параметр:

$$\theta^k = |\frac{K_2^k - K_3^k}{K_1^k - K_2^k}|.$$

Если величина θ^k порядка нескольких сотых единицы, то расчет продолжается с тем же шагом, если θ^k больше одной десятой, то шаг следует уменьшить, если же θ^k меньше одной сотой, то шаг можно увеличить.
Таким образом, с помощью определения величин θ^k или R^h можно организовать алгоритм выбора шага h для явного метода Рунге-Кутты.

Решение задачи Коши для системы обыкновенных дифференциальных уравнений.

Рассматривается задача Коши для системы дифференциальных уравнений первого порядка разрешенных относительно производной

$$\begin{cases} y_1' = f_1(x, y_1, y_2, \ldots, y_n) \\ y_2' = f_2(x, y_1, y_2, \ldots, y_n) \\ \quad \ldots\ldots\ldots\ldots\ldots\ldots \\ y_n' = f_n(x, y_1, y_2, \ldots, y_n) \end{cases} \tag{3}$$

$$\begin{cases} y_1(x_0) = y_{01} \\ y_2(x_0) = y_{02} \\ \ldots\ldots\ldots \\ y_n(x_0) = y_{0n} \end{cases} \tag{4}$$

Система в более компактном виде записывается в векторной форме

$$\bar{y}' = \bar{F}(x, y),$$

$$\bar{y}(x_0) = \bar{y}_0.$$

Здесь $\bar{y}(x) = (y_1, y_2, \ldots, y_n)^T$ — вектор-столбец неизвестных функций, $\bar{F} = (f_1, f_2, \ldots, f_n)^T$ – вектор-функция правых частей.

К векторному дифференциальному уравнению можно применить все методы, рассмотренные выше в данном разделе (благодаря линейной структуре всех рассмотренных методов). При этом в формулах все величины векторные кроме переменной

x и шага h.
Рассмотрим задачу Коши для системы двух ОДУ первого порядка, где уравнения записаны в развернутом виде

$$\begin{cases} y' = f(x,y,z) \\ z' = g(x,y,z) \\ y(x_0) = y_0 \\ z(x_0) = x_0 \end{cases} \tag{5}$$

Формулы метода Рунге-Кутты 4-го порядка точности для решения системы следующие:

$$y_{k+1} = y_k + \Delta y_k,$$
$$z_{k+1} = z_k + \Delta y_z,$$
$$\Delta y_k = \frac{1}{6}(K_1^k + 2\cdot K_2^k + 2\cdot K_3^k + K_4^k),$$
$$\Delta z_k = \frac{1}{6}(L_1^k + 2\cdot L_2^k + 2\cdot L_3^k + L_4^k),$$
$$K_1^k = h\cdot f(x_k, y_k, z_k),$$
$$L_1^k = h\cdot g(x_k, y_k, z_k),$$
$$K_2^k = h\cdot f(x_k + \frac{1}{2}h, y_k + \frac{1}{2}K_1^k, z_k + \frac{1}{2}L_1^k),$$
$$L_2^k = h\cdot g(x_k + \frac{1}{2}h, y_k + \frac{1}{2}K_1^k, z_k + \frac{1}{2}L_1^k),$$
$$K_3^k = h\cdot f(x_k + \frac{1}{2}h, y_k + \frac{1}{2}K_2^k, z_k + \frac{1}{2}L_2^k),$$
$$L_3^k = h\cdot g(x_k + \frac{1}{2}h, y_k + \frac{1}{2}K_2^k, z_k + \frac{1}{2}L_2^k),$$
$$K_4^k = h\cdot f(x_k + h, y_k + K_3^k, z_k + L_3^k),$$
$$L_4^k = h\cdot g(x_k + h, y_k + K_3^k, z_k + L_3^k).$$

Контроль правильности выбора шага h в случае использования метода Рунге-Кутты четвертого порядка точности для системы ОДУ может быть организован с помощью вычисления на каждом шаге h параметров

$$\theta_1^k = |\frac{K_2^k - K_3^k}{K_1^k - K_2^k}|, \qquad \theta_2^k = |\frac{L_2^k - L_3^k}{L_1^k - L_2^k}|.$$

Если величины $\theta_i^k, \quad i = 1, 2, \ldots$ порядка нескольких сотых единицы, то расчет продолжается с тем же шагом, если больше одной десятой, то шаг следует уменьшить, если же меньше одной сотой, то шаг можно увеличить.

Решение задачи Коши для ОДУ второго и более высоких порядков.

Задача Коши для $n-$го порядка ставится следующим образом

$$y^{(n)} = f(x, y, y', y'', \dots, y^{(n-1)})$$

$$\begin{cases} y(x_0) = y_0 \\ y'(x_0) = y_{01} \\ y''(x_0) = y_{02} \\ \dots\dots\dots\dots\dots \\ y^{(n-1)}(x_0) = y_{0(n-1)} \end{cases} \tag{6}$$

Здесь $y^{(m)} = \frac{d^m y}{dx^m}$ производная m порядка от решения, $m = 1, 2, \dots, n$.
Основной прием используемый при решении задач такого типа заключается в введении новых переменных и сведении задачи для ОДУ высокого порядка к решению системы ОДУ первого порядка.
Введем новые переменные

$$\begin{cases} z_1 = y' \\ z_2 = y'' \\ \dots\dots\dots\dots \\ z_{n-1} = y^{(n-1)}, \end{cases} \tag{7}$$

тогда исходную задачу можно переписать в виде n ОДУ первого порядка

$$\begin{cases} y' = z_1 \\ z_1' = z_2 \\ \dots\dots\dots\dots \\ z_{n-2}' = z_{n-1} \\ z_{n-1}' = f(x, y, z_1, \dots, z_{n-1}) \end{cases} \tag{8}$$

$$\begin{cases} y(x_0) = y_0 \\ z_1(x_0) = y_{01} \\ z_2(x_0) = y_{02} \\ \dots\dots\dots\dots \\ z_{n-1}(x_0) = y_{0(n-1)} \end{cases} \tag{9}$$

Полученная система, состоящая из n ОДУ первого порядка с соответствующими начальными условиями решается любым из описанных методов.
Пусть необходимо решить задачу Коши для ОДУ второго порядка:

$$\begin{cases} y'' = f(x, y, y') \\ y(x_0) = y_0 \\ y'(x_0) = y_{01} \end{cases} \tag{10}$$

Путем введения замены $z = y'$, сведем ОДУ второго порядка к системе:

$$\begin{cases} y' = z \\ z' = f(x, y, z) \\ y(x_0) = y_0 \\ z(x_0) = y_{01}, \end{cases} \tag{11}$$

которую можно решить, например, с использованием метода Рунге-Кутты.

Многошаговые методы. Метод Адамса.

Многошаговые методы решения задачи Коши характеризуются тем, что решение в текущем узле зависит от данных не в одном предыдущем узле, как это имеет место в одношаговых методах, а от нескольких предыдущих узлах. Многие многошаговые методы различного порядка точности можно конструировать с помощью квадратурного способа (т.е. с использованием эквивалентного интегрального уравнения).
Решение дифференциального уравнения $y' = f(x, y)$ удовлетворяет интегральному соотношению:

$$y_{k+1} = y_k + \int\limits_{x_k}^{x_{k+1}} f(x, y(x))dx$$

Если решение задачи Коши получено в узлах вплоть до $k-$го, то можно аппроксимировать подынтегральную функцию, например: интерполяционным многочленом какой-либо степени. Вычислив интеграл от построенного многочлена на отрезке $[x_i, x_{i+k}]$ получим ту или иную формулу Адамса. В частности, если использовать многочлен нулевой степени (то есть заменить подынтегральную функцию ее значением на левом конце отрезка в точке x_k), то получим явный метод Эйлера. Если проделать то же самое, но подынтегральную функцию аппроксимировать значением на правом конце в точке x_{k+1}, то получим неявный метод Эйлера.

Метод Адамса.

При использовании интерполяционного многочлена $3-$й степени построенного по значениям подынтегральной функции в последних четырех узлах получим метод Адамса четвертого порядка точности:

$$y_{k+1} = y_k + \frac{h}{24} \cdot (55f_k - 59f_{k-1} + 37f_{k-2} - 9f_{k-3}),$$

где f_k значение подынтегральной функции в узле x_k.
Метод Адамса как и все многошаговые методы не является самостартующим, то есть для того, что бы использовать метод Адамса необходимо иметь решения в первых четырех узлах. В узле x_0 решение y_0 известно из начальных условий, а в других трех узлах $x_1, \; x_2, \; x_3$ решения $y_1, \; y_2, \; y_3$ можно получить с помощью подходящего одношагового метода, например: метода Рунге-Кутты четвертого порядка.

2 Моя задача

Задача Коши:

$$\begin{cases} (x^2+1)\cdot y'' - 2x\cdot y' + 2y = 0, \\ y(0) = 1, \\ y'(0) = 1, \\ x \in [0,1], h = 0.1. \end{cases} \tag{12}$$

Точное решение: $y = x - x^2 + 1$.
Сведем ОДУ второго порядка к решению системы ОДУ первого порядка. Для этого введем новые переменные.
Пусть $y = y_1$, тогда имеем:

$$\begin{cases} y_1' = y_2 \\ y_2' = \frac{2x\cdot y_2 - 2y_1}{x^2+1} \\ y_1(0) = 1 \\ y_2(0) = 1 \\ x \in [0,1], h = 0.1 \end{cases} \tag{13}$$

Решение системы ОДУ первого порядка методом Эйлера:
При $n = \frac{b-a}{h}, \quad i = 0, 1, \ldots, n:$

$$y_1[i+1] = y[i] + h\cdot y_2[i];$$

$$y_2[i+1] = y_2[i] + h\cdot f(x[i], y_1[i], y_2[i]),$$

где

$$f = \frac{2x\cdot y_2 - 2y_1}{x^2+1};$$

$$x[i+1] = x[i] + h.$$

Решение системы ОДУ первого порядка методом Рунге-Кутта:
При $n = \frac{b-a}{h}, \quad i = 0, 1, \ldots, n:$

$$y_1[i+1] = y_1[i] + \Delta y_1[i],$$

$$y_2[i+1] = y_2[i] + \Delta y_2[i],$$

$$\Delta y_1[i] = \frac{1}{6}(K_1^i + 2\cdot K_2^i + 2\cdot K_3^i + K_4^i),$$

$$\Delta y_2[i] = \frac{1}{6}(L_1^i + 2\cdot L_2^i + 2\cdot L_3^i + L_4^i),$$

$$K_1^i = h \cdot y_2[i],$$
$$L_1^i = h \cdot f(x[i], y_1[i], y_2[i]),$$
$$K_2^i = h \cdot (y_2[i] + \frac{1}{2}L_1^i),$$
$$L_2^i = h \cdot f(x[i] + \frac{1}{2}h, y_1[i] + \frac{1}{2}K_1^i, y_2[i] + \frac{1}{2}L_1^i),$$
$$K_3^i = h \cdot (y_2[i] + \frac{1}{2}L_2^i),$$
$$L_3^i = h \cdot f(x[i] + \frac{1}{2}h, y_1[i] + \frac{1}{2}K_2^i, y_2[i] + \frac{1}{2}L_2^i),$$
$$K_4^i = h \cdot (y_2[i] + L_3^i),$$
$$L_4^i = h \cdot f(x_i + h, y_1[i] + K_3^i, y_2[i] + L_3^i),$$

где

$$f = \frac{2x \cdot y_2 - 2y_1}{x^2 + 1},$$
$$x[i+1] = x[[i] + h.$$

Решение системы ОДУ первого порядка методом Адамса:
Первые три значения y_1, y_2 и y_3 находятся методом Рунге-Кутта по формулам, представленным выше.
Далее, согласно методу Адамса: При $n = \frac{b-a}{h}$, $i = 3, 4, \ldots, n$:

$$y_1[i+1] = y_1[i] + \frac{h}{24} \cdot (55y_2[i] - 59y_2[i-1] + 37y_2[i-2] - 9y_2[i-3]);$$

$$y_2[i+1] = y_2[i] + \frac{h}{24} \cdot (55 \cdot f(x[i], y_1[i], y_2[i]) - 59 \cdot f(x[i-1], y_1[i-1], y_2[i-1]) +$$
$$+37 \cdot f(x[i-2], y_1[i-2], y2[i-2]) - 9f(x[i-3], y_1[i-3], y2[i-3])),$$

где

$$f = \frac{2x \cdot y_2 - 2y_1}{x^2 + 1},$$
$$x[i+1] = x[i] + h.$$

3 Протокол

Входные данные я ввожу через консоль: значение шага h.
Выходные данные я вывожу на экран.
Скриншот консоли:

I. Метод Эйлера.

```
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab4$ g++ 4.1-1.cpp -o 4.1-1
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab4$ ./4.1-1
0.1
Решение задачи явным методом Эйлера задано таблично:
x = 0; y = 1
eps = 0;  Погрешность по Рунге-Ромбергу = 0

x = 0.1; y = 1.1
eps = 0.01;  Погрешность по Рунге-Ромбергу = 0.005

x = 0.2; y = 1.18
eps = 0.02;  Погрешность по Рунге-Ромбергу = 0.0100498

x = 0.3; y = 1.2398
eps = 0.029802;  Погрешность по Рунге-Ромбергу = 0.0150501

x = 0.4; y = 1.27921
eps = 0.0392117;  Погрешность по Рунге-Ромбергу = 0.0199024

x = 0.5; y = 1.29804
eps = 0.0480422;  Погрешность по Рунге-Ромбергу = 0.0245114

x = 0.6; y = 1.29612
eps = 0.0561159;  Погрешность по Рунге-Ромбергу = 0.0287856

x = 0.7; y = 1.27327
eps = 0.0632668;  Погрешность по Рунге-Ромбергу = 0.0326393

x = 0.8; y = 1.22934
eps = 0.0693411;  Погрешность по Рунге-Ромбергу = 0.0359922

x = 0.9; y = 1.1642
eps = 0.0741973;  Погрешность по Рунге-Ромбергу = 0.0387705

x = 1; y = 1.07771
eps = 0.0777061;  Погрешность по Рунге-Ромбергу = 0.0409064

(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab4$
```

Решение ОДУ методом Эйлера:

x	y	ε
0	1	0
0.1	1.1	0.01
0.2	1.18	0.02
0.3	1.2398	0.029802
0.4	1.27921	0.0392117
0.5	1.29804	0.0480422
0.6	1.29612	0.0561159
0.7	1.27327	0.0632668
0.8	1.22934	0.0693411
0.9	1.1642	0.0741973
1	1.07771	0.0777061

Погрешность достаточно большая, т.к. явный метод Эйлера имеет первый порядок точности — численное решение совпадает с релаьным решением до первого знака после запятой.

II. Метод Рунге-Кутты.
Решение ОДУ методом Рунге-Кутты:

x	y	ε
0	1	0
0.1	1.09	4.14591e-08
0.2	1.16	4.17591e-09
0.3	1.21	1.40984e-07
0.4	1.24	3.72462e-07
0.5	1.25	7.0078e-07
0.6	1.24	1.12655e-06
0.7	1.21	1.64904e-06
0.8	1.16	2.26652e-06
0.9	1.09	2.97671e-06
1	1	3.77712e-06

```
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab4$ g++ 4.1-2.cpp -o 4.1-2
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab4$ ./4.1-2
0.1
Решение задачи методом Рунге-Кутты задано таблично:
x_1 = 0; y_1 = 1
eps = 0;  Погрешность по Рунге-Ромбергу = 0

x_2 = 0.1; y_2 = 1.09
eps = 4.14591e-08;  Погрешность по Рунге-Ромбергу = 2.76507e-09

x_3 = 0.2; y_3 = 1.16
eps = 4.17591e-09;  Погрешность по Рунге-Ромбергу = 8.06593e-11

x_4 = 0.3; y_4 = 1.21
eps = 1.40984e-07;  Погрешность по Рунге-Ромбергу = 8.31241e-09

x_5 = 0.4; y_5 = 1.24
eps = 3.72462e-07;  Погрешность по Рунге-Ромбергу = 2.26347e-08

x_6 = 0.5; y_6 = 1.25
eps = 7.0078e-07;  Погрешность по Рунге-Ромбергу = 4.30214e-08

x_7 = 0.6; y_7 = 1.24
eps = 1.12655e-06;  Погрешность по Рунге-Ромбергу = 6.95096e-08

x_8 = 0.7; y_8 = 1.21
eps = 1.64904e-06;  Погрешность по Рунге-Ромбергу = 1.0205e-07

x_9 = 0.8; y_9 = 1.16
eps = 2.26652e-06;  Погрешность по Рунге-Ромбергу = 1.40534e-07

x_10 = 0.9; y_10 = 1.09
eps = 2.97671e-06;  Погрешность по Рунге-Ромбергу = 1.84816e-07

x_11 = 1; y_11 = 1
eps = 3.77712e-06;  Погрешность по Рунге-Ромбергу = 2.34739e-07

(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab4$
```

Этот метод имеет четвертый порядок точности и, соответственно, численное решение, полученное методом Рунге-Кутты, имеет очень малую погрешность.

III. Метод Адамса.

```
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab4$ g++ 4.1-3.cpp -o 4.1-3
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab4$ ./4.1-3
0.1
Решение задачи методом Адамса задано таблично:
x_1 = 0; y_1 = 1
eps = 0;  Погрешность по Рунге-Ромбергу = 0

x_2 = 0.1; y_2 = 1.09
eps = 4.14591e-08;  Погрешность по Рунге-Ромбергу = 2.76507e-09

x_3 = 0.2; y_3 = 1.16
eps = 4.17591e-09;  Погрешность по Рунге-Ромбергу = 4.42842e-10

x_4 = 0.3; y_4 = 1.21
eps = 1.40984e-07;  Погрешность по Рунге-Ромбергу = 7.70491e-09

x_5 = 0.4; y_5 = 1.24
eps = 6.52765e-07;  Погрешность по Рунге-Ромбергу = 4.07522e-08

x_6 = 0.5; y_6 = 1.25
eps = 1.06588e-06;  Погрешность по Рунге-Ромбергу = 6.7196e-08

x_7 = 0.6; y_7 = 1.24
eps = 1.59237e-06;  Погрешность по Рунге-Ромбергу = 1.01174e-07

x_8 = 0.7; y_8 = 1.21
eps = 2.14275e-06;  Погрешность по Рунге-Ромбергу = 1.36717e-07

x_9 = 0.8; y_9 = 1.16
eps = 2.68449e-06;  Погрешность по Рунге-Ромбергу = 1.71657e-07

x_10 = 0.9; y_10 = 1.09
eps = 3.25989e-06;  Погрешность по Рунге-Ромбергу = 2.08814e-07

x_11 = 1; y_11 = 1
eps = 3.85724e-06;  Погрешность по Рунге-Ромбергу = 2.47408e-07

(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ/Lab4$
```

Решение ОДУ методом Адамса:

x	y	ε
0	1	0
0.1	1.09	4.14591e-08
0.2	1.16	4.17591e-09
0.3	1.21	1.40984e-07
0.4	1.24	6.52765e-07
0.5	1.25	1.06588e-06
0.6	1.24	1.59237e-06
0.7	1.21	2.14275e-06
0.8	1.16	2.68449e-06
0.9	1.09	3.25989e-06
1	1	3.85724e-06

Этот метод имеет четвертый порядок точности и, соответственно, численное решение, полученное методом методом Адамса, имеет очень малую погрешность. Порядок погрешности совпадает с порядком погрешности метода Рунге-Кутты.

4 Исходный код

Листинг 1: Метод Эйлера

```
#include <stdio.h>
#include <iostream>
#include <vector>
#include <math.h>

double ExactVal(double x)
{
    double val = x - pow(x, 2) + 1;
    return val;
}

double F(double x, double y1, double y2)
{
    double value = 2*(x*y2 - y1)/(pow(x, 2) + 1);
    return value;
}

void ExEuler(std::vector<double>& y, std::vector<double>& eps, double h, int a, int b)
{
    double n = (b - a)/h + 1;
    double y_h = 1;
    y[0] = 1;
    eps[0] = fabs(ExactVal(a) - y[0]);
    for(int i = 0; i < n-1; i++)
    {
        y[i+1] = y[i] + h*y_h;
        y_h += h*F(a + h*i, y[i], y_h);
        eps[i+1] = fabs(ExactVal(a + h*(i+1)) - y[i+1]);
    }
}

int main()
{
    double k = 0.5;
    double h, n, m, x;
    double y_h;
    int b = 1;
    int a = 0;
    std::cin >> h;
```

```
    n = (b-a)/h + 1;
    m = (b-a)/(h*k) + 1;

    std::vector<double> y(n);
    std::vector<double> eps(n);
    std::vector<double> rung(m);
    std::vector<double> eps2(m);

    ExEuler(y, eps, h, a, b);
    ExEuler(rung, eps2, k*h, a, b);

    std::cout << "The solution of the problem using the explicit Euler method is given in a table:"
         << std::endl;
    for(int i = 0; i < n; i++)
    {
        std::cout << "x = " << a + i*h << "; y = " << y[i] << std::endl;
        std::cout << "eps = " << eps[i] << "; The error in the Runge-Romberg = " << fabs(y
            [i]-rung[i/k]) << std::endl << std::endl;
    }
    return 0;
}
```

Листинг 2: Метод Рунге-Кутты

```
#include <stdio.h>
#include <iostream>
#include <vector>
#include <math.h>

double ExactVal(double x)
{
    double val = x - pow(x, 2) + 1;
    return val;
}

double F(double x, double y1, double y2)
{
    double value = 2*(x*y2 - y1)/(pow(x, 2) + 1);
    return value;
}

void Runge_Kutta(std::vector<double>& y, std::vector<double>& eps, double h, int a, int b)
{
```

```
    int n = (b-a)/h + 1;
    double y_h = 1;
    int p = 4;
    std::vector<double> K(p);
    std::vector<double> L(p);
    y[0] = 1;
    eps[0] = fabs(ExactVal(a) - y[0]);

    for(int i = 0; i < n-1; i++)
    {
        K[0] = h*y_h;
        L[0] = h*F(a + h*i, y[i], y_h);
        K[1] = h*(y_h + L[0]/2);
        L[1] = h*F(a + h*i + h/2, y[i] + K[0]/2, y_h + L[0]/2);
        K[2] = h*(y_h + L[1]/2);
        L[2] = h*F(a + h*i + h/2, y[i] + K[1]/2, y_h + L[1]/2);
        K[3] = h*(y_h + L[2]);
        L[3] = h*F(a + h*i + h, y[i] + K[2], y_h + L[2]);
        y[i+1] = y[i] + (K[0] + 2*K[1] + 2*K[2] + K[3])/6;
        y_h += (L[0] + 2*L[1] + 2*L[2] + L[3])/6;
        eps[i+1] = fabs(ExactVal(a + h*(i+1)) - y[i+1]);
    }
}

int main()
{
    double k = 0.5;
    double h, h_r, x;
    int n, m;
    double y_h;
    int a = 0;
    int b = 1;

    std::cin >> h;
    h_r = h*k;
    n = (b-a)/h + 1;
    m = (b-a)/h_r + 1;

    std::vector<double> y(n);
    std::vector<double> eps(n);
    std::vector<double> rung(m);
    std::vector<double> eps2(m);

    Runge_Kutta(y, eps, h, a, b);
```

```
    Runge_Kutta(rung, eps2, h_r, a, b);

    std::cout << "The solution of the problem by the Runge-Kutta method is given in a table:" << std::endl;
    for(int i = 0; i < n; i++)
    {
        std::cout << "x_" << i + 1<< " = " << a + i*h << "; y_" << i + 1 << " = " << y[i] << std::endl;
        std::cout << "eps = " << eps[i] << "; The error in the Runge-Romberg = " << fabs(y[i]-rung[i/k])/15 << std::endl << std::endl;
    }
    return 0;
}
```

Листинг 3: Метод Адамса

```
#include <stdio.h>
#include <iostream>
#include <vector>
#include <math.h>

double ExactVal(double x)
{
    double val = x - pow(x, 2) + 1;
    return val;
}

double F(double x, double y1, double y2)
{
    double value = 2*(x*y2 - y1)/(pow(x, 2) + 1);
    return value;
}

void Runge_Kutta(std::vector<double>& y, std::vector<double>& y2, std::vector<double>& eps, double h, int a, int b)
{
    int n = 4;
    int p = 4;
    std::vector<double> K(p);
    std::vector<double> L(p);
    y[0] = y2[0] = 1;
    eps[0] = fabs(ExactVal(0) - y[0]);
    for(int i = 0; i < n-1; i++)
```

```
    {
        K[0] = h*y2[i];
        L[0] = h*F(a + h*i, y[i], y2[i]);
        K[1] = h*(y2[i] + L[0]/2);
        L[1] = h*F(a + h*i + h/2, y[i] + K[0]/2, y2[i] + L[0]/2);
        K[2] = h*(y2[i] + L[1]/2);
        L[2] = h*F(a + h*i + h/2, y[i] + K[1]/2, y2[i] + L[1]/2);
        K[3] = h*(y2[i] + L[2]);
        L[3] = h*F(a + h*i + h, y[i] + K[2], y2[i] + L[2]);

        y[i+1] = y[i] + (K[0] + 2*K[1] + 2*K[2] + K[3])/6;
        y2[i+1] = y2[i] + (L[0] + 2*L[1] + 2*L[2] + L[3])/6;
        eps[i+1] = fabs(ExactVal(a + h*(i+1)) - y[i+1]);
    }
}

void Adams(std::vector<double>& y, std::vector<double>& eps, double h, int a, int b)
{
    int n = (b-a)/h + 1;
    std::vector<double> y2(n);
    Runge_Kutta(y, y2, eps, h, a, b);
    for(int i = 3; i < n-1; i++)
    {
        y[i+1] = y[i] + h/24*(55*y2[i] - 59*y2[i-1] + 37*y2[i-2] - 9*y2[i-3]);
        y2[i+1] = y2[i] + h/24*(55*F(a + h*i, y[i], y2[i]) - 59*F(a + h*(i-1), y[i-1], y2[i-1])
            + 37*F(a + h*(i-2), y[i-2], y2[i-2]) - 9*F(a + h*(i-3), y[i-3], y2[i-3]));
        eps[i+1] = fabs(ExactVal(a + h*(i+1)) - y[i+1]);
    }
}

int main()
{
    double k = 0.5;
    double h, h_r, x;
    int n, m;
    double y_h;
    int a = 0;
    int b = 1;

    std::cin >> h;
    h_r = h*k;
    n = (b-a)/h + 1;
    m = (b-a)/h_r + 1;

```

```
    std::vector<double> y(n);
    std::vector<double> eps(n);
    std::vector<double> rung(m);
    std::vector<double> eps2(m);

    Adams(y, eps, h, a, b);
    Adams(rung, eps2, h_r, a, b);

    std::cout << "The solution of the problem using the Adams method is set in a table:" << std
        ::endl;
    for(int i = 0; i < n; i++)
    {
        std::cout << "x_" << i + 1 << " = " << a + i*h << "; y_" << i + 1 << " = "
            << y[i] << std::endl;
        std::cout << "eps = " << eps[i] << "; The error in the Runge-Romberg = " << fabs(y
            [i]-rung[i/k])/15 << std::endl << std::endl;
    }
    return 0;
}
```

5 Выводы

Выполнив первое задание четвертой лабораторной работы, я познакомилась с тремя методами численного решения обыкновенных дифференциальных уравнений и систем дифференциальных уравнений 1 и 2 порядков. На некотором интервале вводится разностная сетка — совокупность узлов, в которых будет аппроксимироваться функция. Такая функция называется сеточной и представляется в табличном виде. Метод Эйлера имеет первый порядок точности — приращение функции в следующем узле зависит от тангенса угла в данной точке и от значения шага. Так, при малом шаге считают, что значение неизвестного точного решения совпадает со значением касательной в точке. Несколько модификаций метода Эйлера имеют такой же порядок точности, первый улучшенный метод имеет порядок точности, равный двум. Так же я познакомилась с двумя методами Рунге-Кутты — третьего и четвертого порядков точности. В решении задачи я использовала метод с большим порядком. Метод Рунге-Кутты вычисляет несколько параметров, которые подбираются так, чтобы значение y совпадало со значением разложения в точке x точного решения в ряд Тейлора. Метод Адамса является многошаговым методом — в процессе поиска решения в точке x_k он обращается к уже вычисленным ранее значениям в точках $x_{k-1},\quad x_{k-2}, x_{k-3}$. Порядок точности метода Адамса равен четырем.