МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М. В. Ломоносова

Факультет вычислительной математики и кибернетики

Компьютерный практикум по учебному курсу «Введение в численные методы» Задание №2

Отчет о выполненном задании

студентки 219 учебной группы факультета ВМК МГУ

Чернобай Анны Александровны

Москва
2020

Оглавление

Глава 1

Решение задачи Коши для дифференциального уравнения первого порядка или системы дифференциальных уравнений первого порядка

Постановка задачи

Рассматривается обыкновенное дифференциальное уравнение первого порядка, разрешенное относительно производной и имеющее вид:

$$\frac{dy}{dx} = f(x, y), x < x_0 \tag{1.1}$$

с дополнительным начальным условием, заданным в точке $x = x_0$:

$$y(x_0) = y_0 \tag{1.2}$$

Предполагается, что правая часть уравнения такова, что гарантирует существование и единственность решения задачи Коши.

В случае, если рассматривается не одно дифференциальное уравнение вида 1.1, а система обыкновенных дифференциальных уравнений первого порядка, разрешенных относительно производных неизвестных функций, соответствующая задача Коши имеет вид (на примере двух дифференциальных уравнений):

$$\begin{cases} \frac{dy_1}{dx} = f_1(x, y_1, y_2) \\ \frac{dy_2}{dx} = f_2(x, y_1, y_2) \\ x > x_0 \end{cases} \tag{1.3}$$

Дополнительные начальные условия задаются в точке $x = x_0$:

$$y_1(x_0) = y_1^0{}_2(x_0) = y_2^0 \tag{1.4}$$

Также предполагается, что правые части уравнений 1.3 заданы так, что гарантируетсz существование и единственность решения задачи Коши 1.3 - 1.4, но уже для системы обыкновенных дифференциальных уравнений первого порядка в форме, разрешенной относительно производных неизвестных функций.

Задачи:

1. Решить задачу Коши методами Рунге-Кутта второго и четвертого порядка точности, аппроксимировав дифференциальную задачу соответствующей разностной схемой (на равномерной сетке); полученное конечно-разностное уравнение (или уравнения в случае системы), просчитать численно

2. Найти численное решение задачи и построить его график

3. Найденное численное решение сравнить с точным решением дифференциального уравнения.

Метод и алгоритм решения

Пусть на отрезке $[x_0, x_0 + l]$ задана равномерная сетка $x_{i+1} - x_i = h = \frac{l}{n}, 0 \leq i \leq n-1$.

Из метода Эйлера численного решения дифференциальных уравнений при обрыве разложения по формуле Тейлора на члене порядка h^2 получим

$$\frac{y_{i+1}}{h} - \frac{y_i}{h} = f(x_i, y_i) + \frac{1}{2}(\frac{\partial f}{\partial x}(x_i, y_i) + \frac{\partial f}{\partial y}(x_i, y_i) f(x_i, y_i))h \tag{1.5}$$

разностное уравнение:

Приближенно заменим первые производные от функции $f(x, y)$ на сумму значений функции f в двух разных точках с точностью до членов порядка h^2. Положим:

$$\frac{y_{i+1}}{h} - \frac{y_i}{h} = f(x_i, y_i) + \frac{1}{2}(\frac{\partial f}{\partial x}(x_i, y_i) + \frac{\partial f}{\partial y}(x_i, y_i) f(x_i, y_i))h = \beta f(x_i, y_i) + \alpha f(x_i + \delta h, y_i + \gamma h) + O(h^2) \tag{1.6}$$

где α, β, δ, γ — четыре свободных параметра, которые нужно подобрать так, чтобы правая часть равнялась левой с нужной степенью точности.

После разложения функции $f(x_i + \delta h, y_i + \gamma h)$ по степени h получим:

$$\beta = 1 - \alpha, \gamma = \frac{1}{2\alpha}, \delta = \frac{1}{2\alpha} f(x_i, y_i) \tag{1.7}$$

Приходим к рекуррентному соотношению $y_{i+1} = y_i + ((1-\alpha) f(x_i, y_i) + \alpha f(x_i + \frac{h}{2\alpha}, y_i + \frac{h}{2\alpha} f(x_i, y_i)))h$, которое зависит от параметра α. Чаще всего используется метод Рунге-Кутта с параметром $\alpha = \frac{1}{2}$ или $\alpha = 1$.

Ниже приведем формулы метода Рунге-Кутта II порядка точности:

$$y_{i+1} = y_i + \frac{h}{2}(f(x_i, y_i) + f(x_i + h, y_i + hf(x_i, y_i))) \tag{1.8}$$

Для IV порядка точности:

$$y_{i+1} - i = \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \text{где} \tag{1.9}$$

$$k_1 = f(x_i, y_i),\ k_2 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2} k_1),\ k_3 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2} k_2),\ k_4 = f(x_i + h, y_i + hk_3) \tag{1.10}$$

Формулы метода Рунге-Кутта для системы из 2-х обыкновенных дифференциальных уравнений II порядка точности :

$$\begin{cases} u_{i+1} = u_i + \frac{h}{2}(k_1 + k_2) \\ v_{i+1} = v_i + \frac{h}{2}(m_1 + m_2) \end{cases} \quad \text{, где} \tag{1.11}$$

$$k_1 = f_1(x_i, u_i, v_i)$$

$$m_1 = f_2(x_i, u_i, v_i)$$

$$k_2 = f_1(x_i + h, u_i + hk_1, v_i + hm_1)$$

$$m_2 = f_2(x_i + h, u_i + hk_1, v_i + hm_1)$$

Формулы метода Рунге-Кутта для системы из 2-х обыкновенных дифференциальных уравнений IV порядка точности :

$$\begin{cases} u_{i+1} = u_i + \frac{h}{2}(k_1 + 2k_2 + 2k_3 + k_4) \\ v_{i+1} = v_i + \frac{h}{2}(m_1 + 2m_2 + 2m_3 + m_4) \end{cases} \quad \text{, где} \tag{1.12}$$

$$k_1 = f_1(x_i, u_i, v_i)$$

$$k_2 = f_1(x_i + \frac{h}{2}, u_i + \frac{hk_1}{2}, v_i + \frac{hm_1}{2})$$

$$k_3 = f_1(x_i + \frac{h}{2}, u_i + \frac{hk_2}{2}, v_i + \frac{hm_2}{2})$$

$$k_4 = f_1(x_i + h, u_i + hk_3, v_i + hm_3)$$

$$m_1 = f_2(x_i, u_i, v_i)$$

$$m_2 = f_2(x_i + \frac{h}{2}, u_i + \frac{hk_1}{2}, v_i + \frac{hm_1}{2})$$

$$m_3 = f_2(x_i + \frac{h}{2}, u_i + \frac{hk_2}{2}, v_i + \frac{hm_2}{2})$$

$$m_4 = f_2(x_i + h, u_i + hk_3, v_i + hm_3)$$

Структура программы и спецификация функций

Программа состоит из одного модуля RUNGEKUTT.c. В данном модуле реализованы функции, описанные ниже:

Спецификация функций:

- ```
  double f(double x, double y);
  double f_ans(double x);
  ```

  Функции, задающие ОДУ первого порядка и решение данного ОДУ соответственно. ОДУ и его решение взяты из таблицы 1, пункта 3.

- ```
  double test1(double x, double y);
  double test1_ans(double x);
  double test2(double x, double y);
  double test2_ans(double x);
  ```

  Функции, задающие ОДУ первого порядка и решение данного ОДУ соответственно. Данные функции реализованы для тестирования программы. ОДУ и его решение взяты из таблицы 1, пункты 1 и 2.

- ```
  double f1(double x, double u, double v);
  double f2(double x, double u, double v);
  ```

  Функции, задающие систему из двух ОДУ первого порядка. Система ОДУ взята из таблицы 2, пункта 19.

- ```
  double test_sys11(double x, double u, double v);
  double test_sys12(double x, double u, double v);
  double test_sys21(double x, double u, double v);
  double test_sys22(double x, double u, double v);
  ```

  Функции, задающие системы из двух ОДУ первого порядка. Данные функции реализованы для тестирования программы. Система ОДУ взята из таблицы 2, пункты 2 и 3.

- ```
  void runge_kutt_2(double x0, double y0, double h, int n, double(*f)(),
  double(*f_ans)());
  void runge_kutt_4(double x0, double y0, double h, int n, double(*f)(),
  double(*f_ans)());
  ```

  Функции, реализующие соответственно методы Рунге-Кутта II и IV порядка для ОДУ первого порядка.

- ```
  void runge_kutt_2_sys(double x0, double u0, double v0, double h, int n,
  double(*f1)(), double(*f2)());
  void runge_kutt_4_sys(double x0, double u0, double v0, double h, int n,
  double(*f1)(), double(*f2)());
  ```

Функции, реализующие соответственно методы Рунге-Кутта II и IV порядка для системы из двух ОДУ первого порядка.

- `int main(int argc, char *argv[])`

  Основная функция программы. При запуске программы в командной строке необходимо задать один из трех ключей: test(для запуска тестирования реализованных методов Рунге-Кутта на 4 примерах: 2 для ОДУ и 2 для системы ОДУ), equation (для решения ОДУ первого порядка моего варианта практической работы), system (для решения системы из двух ОДУ первого порядка моего варианта практической работы). При задании ключей equation, system необходимо вторым параметром командной строки необходимо задать порядок метода Рунге-Кутта: 2 или 4 соответcnвенно.
  Независимо от введенного ключа далее необходимо ввести в консоль число n - количество и задать сегмент $[a, b]$.

Программа на СИ

```
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <inttypes.h>
#include <limits.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double f(double x, double y) {
    return -y - x * x;
}

double f_ans(double x) {
    return  2 * x - 2 - x * x + 12 * exp(-x);
}

double f1(double x, double u, double v) {
    return sin(x + u) - v * 1.1;
}

double f2(double x, double u, double v) {
    return u * 2.5 - (x + v) * (x + v);
}

double test1(double x, double y) {
    return 3 - y - x;
}

double test1_ans (double x) {
    return 4 - x - 4 * exp(-x);
}

double test2(double x, double y) {
    return sin(x) - y;
}

double test2_ans (double x) {
    return -0.5*cos(x) + 0.5* sin(x) + exp(-x) * 21/2;
}

double test_sys11(double x, double u, double v) {
```

```
    return x * u + v;
}

double test_sys12(double x, double u, double v) {
    return u - v;
}

double test_sys21(double x, double u, double v) {
    return x + v * v;
}

double test_sys22(double x, double u, double v) {
    return x * u;
}

void runge_kutt_2(double x0, double y0, double h, int n, double(*f)(), double(*f_ans)()) {
    double y_cur, x_cur;
    double x_prev = x0, y_prev = y0;
    double f_prev = f(x_prev, y_prev);
    for (int i = 1; i <= n; i++) {
        x_cur = x0 + i * h;
        y_cur = y_prev + (f_prev + f(x_cur, y_prev + f_prev * h)) * h / 2;
        printf("x = %9.5lf y = %9.5lf | y = %9.5lf\n", x_cur, y_cur, f_ans(x_cur));
        y_prev = y_cur;
        x_prev = x_cur;
        f_prev = f(x_prev, y_prev);
    }
}

void runge_kutt_2_sys(double x0, double u0, double v0, double h, int n,
double(*f1)(), double(*f2)()) {
    double u_cur, v_cur, x_cur;
    double x_prev = x0, u_prev = u0, v_prev = v0;
    double fu_prev = u_prev + f1(x_prev, u_prev, v_prev) * h;
    double fv_prev = v_prev + f2(x_prev, u_prev, v_prev) * h;
    for (int i = 1; i <= n; i++) {
        x_cur = x0 + i * h;
        fu_prev = u_prev + f1(x_prev, u_prev, v_prev) * h;
        fv_prev = v_prev + f2(x_prev, u_prev, v_prev) * h;
        u_cur = u_prev + (f1(x_prev, u_prev, v_prev) + f1(x_cur, fu_prev, fv_prev)) * h / 2;
        v_cur = v_prev + (f2(x_prev, u_prev, v_prev) + f2(x_cur, fu_prev, fv_prev)) * h / 2;
        printf("x = %9.5lf u = %9.5lf v = %9.5lf\n", x_cur, u_cur, v_cur);
        x_prev = x_cur;
        u_prev = u_cur;
        v_prev = v_cur;
        fu_prev = u_prev + f1(x_prev, u_prev, v_prev) * h;
```

```
        fv_prev = v_prev + f2(x_prev, u_prev, v_prev) * h;
    }
}

void runge_kutt_4(double x0, double y0, double h, int n, double(*f)(), double(*f_ans)()) {
    double y_cur, x_cur;
    double x_prev = x0, y_prev = y0;
    double k1 = f(x_prev, y_prev) * h;
    double k2 = f(x_prev + h/2, y_prev + k1/2) * h;
    double k3 = f(x_prev + h/2, y_prev + k2/2) * h;
    double k4 = f(x_prev + h, y_prev + k3) * h;
    for (int i = 1; i <= n; i++) {
        x_cur = x0 + i * h;
        y_cur = y_prev + (k1 + 2 * k2 + 2 * k3 + k4)/6;
        printf("x = %9.5lf y = %9.5lf | y = %9.5lf\n", x_cur, y_cur, f_ans(x_cur));
        x_prev = x_cur;
        y_prev = y_cur;
        k1 = f(x_prev, y_prev) * h;
        k2 = f(x_prev + h/2, y_prev + k1/2) * h;
        k3 = f(x_prev + h/2, y_prev + k2/2) * h;
        k4 = f(x_prev + h, y_prev + k3) * h;
    }
}

void runge_kutt_4_sys(double x0, double u0, double v0, double h, int n,
double (*f1)(), double (*f2)()) {
    double x_cur, u_cur, v_cur;
    double x_prev = x0, u_prev = u0, v_prev = v0;
    double k1 = h * f1(x_prev, u_prev, v_prev);
    double l1 = h * f2(x_prev, u_prev, v_prev);
    double k2 = h * f1(x_prev + h / 2, u_prev + k1 / 2, v_prev + l1 / 2);
    double l2 = h * f2(x_prev + h / 2, u_prev + k1 / 2, v_prev + l1 / 2);
    double k3 = h * f1(x_prev + h / 2, u_prev + k2 / 2, v_prev + l2 / 2);
    double l3 = h * f2(x_prev + h / 2, u_prev + k2 / 2, v_prev + l2 / 2);
    double k4 = h * f1(x_prev + h, u_prev + k3, v_prev + l3);
    double l4 = h * f2(x_prev + h, u_prev + k3, v_prev + l3);
    for (int i = 1; i <= n; i++) {
        x_cur = x0 + i * h;
        u_cur = u_prev + (k1 + 2 * k2 + 2 * k3 + k4) / 6;
        v_cur = v_prev + (l1 + 2 * l2 + 2 * l3 + l4) / 6;
        printf("x = %9.5lf u = %9.5lf v = %9.5lf\n", x_cur, u_cur, v_cur);
        x_prev = x_cur;
        u_prev = u_cur;
        v_prev = v_cur;
        k1 = h * f1(x_prev, u_prev, v_prev);
        l1 = h * f2(x_prev, u_prev, v_prev);
```

```
        k2 = h * f1(x_prev + h / 2, u_prev + k1 / 2, v_prev + l1 / 2);
        l2 = h * f2(x_prev + h / 2, u_prev + k1 / 2, v_prev + l1 / 2);
        k3 = h * f1(x_prev + h / 2, u_prev + k2 / 2, v_prev + l2 / 2);
        l3 = h * f2(x_prev + h / 2, u_prev + k2 / 2, v_prev + l2 / 2);
        k4 = h * f1(x_prev + h, u_prev + k3, v_prev + l3);
        l4 = h * f2(x_prev + h, u_prev + k3, v_prev + l3);
    }
}

int main(int argc, char *argv[]) {
    int n;
    double a, b;
    printf("Enter n: ");
    scanf("%d", &n);
    printf("Enter a and b: ");
    scanf("%lf %lf", &a, &b);
    double l = a - b;
    double h = l / n;
    if (argv[1][0] == 't') {
        double x0, y0, y10, y20;
        printf("TEST1: f(x,y) = 3 - y - x, x0 = 0, y0 = 0, n = %d\n", n);
        x0 = 0;
        y0 = 0;
        printf("runge_kutt_2:\n");
        runge_kutt_2(x0, y0, h, n, test1, test1_ans);
        printf("runge_kutt_4:\n");
        runge_kutt_4(x0, y0, h, n, test1, test1_ans);
        printf("TEST2: f(x,y) = sin(x) - y, x0 = 0, y0 = 10, n = %d\n", n);
        x0 = 0;
        y0 = 10;
        printf("runge_kutt_2:\n");
        runge_kutt_2(x0, y0, h, n, test2, test2_ans);
        printf("runge_kutt_4:\n");
        runge_kutt_4(x0, y0, h, n, test2, test2_ans);
        printf("TEST3: f1(x,u,v) = x*u + v, f2(x,u,v) = u - v, x0 = 0, y10 = 0, y20 = 1, n =
        x0 = 0;
        y10 = 0;
        y20 = 1;
        printf("runge_kutt_sys2:\n");
        runge_kutt_2_sys(x0, y10, y20, h, n, test_sys11, test_sys12);
        printf("runge_kutt_sys4:\n");
        runge_kutt_4_sys(x0, y10, y20, h, n, test_sys11, test_sys12);
        printf("TEST4: f1(x,u,v) = x + v*v, f2(x,u,v) = x*u, x0 = 0, y10 = 1, y20 = -1, n = %
        x0 = 0;
        y10 = 1;
        y20 = -1;
```

```
        printf("runge_kutt_sys2:\n");
        runge_kutt_2_sys(x0, y10, y20, h, n, test_sys21, test_sys22);
        printf("runge_kutt_sys4:\n");
        runge_kutt_4_sys(x0, y10, y20, h, n, test_sys21, test_sys22);
    }
    if (argv[1][0] == 'e') {
        double x0 = 0, y0 = 10;
        if (argv[2][0] == '2') {
            runge_kutt_2(x0, y0, h, n, f, f_ans);
        }
        if (argv[2][0] == '4') {
            runge_kutt_4(x0, y0, h, n, f, f_ans);
        }
    }
    if (argv[1][0] == 's') {
        double x0 = 0, y10 = 0.5, y20 = 1;
        if (argv[2][0] == '2') {
            runge_kutt_2_sys(x0, y10, y20, h, n, f1, f2);
        }
        if (argv[2][0] == '4') {
            runge_kutt_4_sys(x0, y10, y20, h, n, f1, f2);
        }
    }
    return 0;
}
```

Тестирование программы

Тестирование программы выполнялось для одного ОДУ на тестах:

Тест	Ожидаемые результаты	Результаты работы программы
$f(x, y) = 3 - y - x$ $(0, 0)$ Segment: $[0, 1]$ $n = 10$	$y = 4 - x - 4e^{-x}$ y = 0.28065 y = 0.52508 y = 0.73673 y = 0.91872 y = 1.07388 y = 1.20475 y = 1.31366 y = 1.40268 y = 1.47372 y = 1.52848	Runge-Kutt II: x = 0.10000 y = 0.28000 x = 0.20000 y = 0.52390 x = 0.30000 y = 0.73513 x = 0.40000 y = 0.91679 x = 0.50000 y = 1.07170 x = 0.60000 y = 1.20239 x = 0.70000 y = 1.31116 x = 0.80000 y = 1.40010 x = 0.90000 y = 1.47109 x = 1.00000 y = 1.52584 Runge-Kutt IV x = 0.10000 y = 0.28065 x = 0.20000 y = 0.52508 x = 0.30000 y = 0.73673 x = 0.40000 y = 0.91872 x = 0.50000 y = 1.07388 x = 0.60000 y = 1.20475 x = 0.70000 y = 1.31366 x = 0.80000 y = 1.40268 x = 0.90000 y = 1.47372 x = 1.00000 y = 1.52848

Тест	Ожидаемые результаты	Результаты работы программы
$f(x, y) = sinx - y$ $(0, 10)$ Segment: $[0, 1]$ $n = 10$	$y = -0.5cos(x) + 0.5sin(x) + \frac{21}{2}e^{-x}$ y = 9.05321 y = 8.20597 y = 7.44868 y = 6.77254 y = 6.16949 y = 5.63218 y = 5.15383 y = 4.72828 y = 4.34984 y = 4.01332	Runge-Kutt II: x = 0.10000 y = 9.05499 x = 0.20000 y = 8.20919 x = 0.30000 y = 7.45304 x = 0.40000 y = 6.77777 x = 0.50000 y = 6.17537 x = 0.60000 y = 5.63852 x = 0.70000 y = 5.16048 x = 0.80000 y = 4.73509 x = 0.90000 y = 4.35671 x = 1.00000 y = 4.02014 Runge-Kutt IV x = 0.10000 y = 9.05321 x = 0.20000 y = 8.20598 x = 0.30000 y = 7.44869 x = 0.40000 y = 6.77254 x = 0.50000 y = 6.16950 x = 0.60000 y = 5.63218 x = 0.70000 y = 5.15384 x = 0.80000 y = 4.72828 x = 0.90000 y = 4.34984 x = 1.00000 y = 4.01332

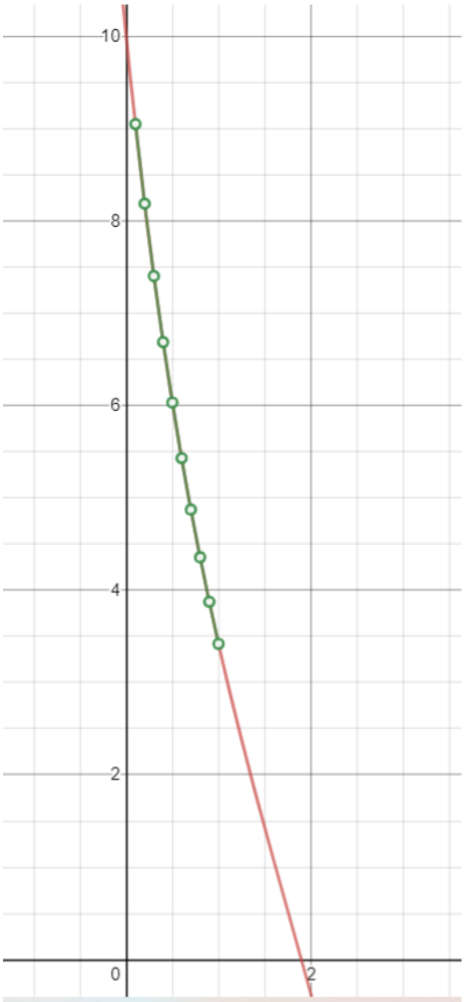
Для системы из двух ОДУ на тестах:

Тест	Результаты работы программы
$f_1(x,u,v) = xu + v$ $f_2(x,u,v) = u - v$ $(0,0,1)$ Segment: $[0,1]$ $n = 10$	Runge-Kutt II x = 0.10000 u = 0.09550 v = 0.91000 x = 0.20000 u = 0.18478 v = 0.83722 x = 0.30000 u = 0.27117 v = 0.77961 x = 0.40000 u = 0.35780 v = 0.73561 x = 0.50000 u = 0.44777 v = 0.70411 x = 0.60000 u = 0.54431 v = 0.68440 x = 0.70000 u = 0.65097 v = 0.67615 x = 0.80000 u = 0.77181 v = 0.67942 x = 0.90000 u = 0.91165 v = 0.69468 x = 1.00000 u = 1.07639 v = 0.72287 Runge-Kutt IV x = 0.10000 u = 0.09564 v = 0.90953 x = 0.20000 u = 0.18499 v = 0.83644 x = 0.30000 u = 0.27140 v = 0.77863 x = 0.40000 u = 0.35804 v = 0.73454 x = 0.50000 u = 0.44801 v = 0.70303 x = 0.60000 u = 0.54457 v = 0.68336 x = 0.70000 u = 0.65129 v = 0.67522 x = 0.80000 u = 0.77227 v = 0.67866 x = 0.90000 u = 0.91236 v = 0.69418 x = 1.00000 u = 1.07752 v = 0.72270

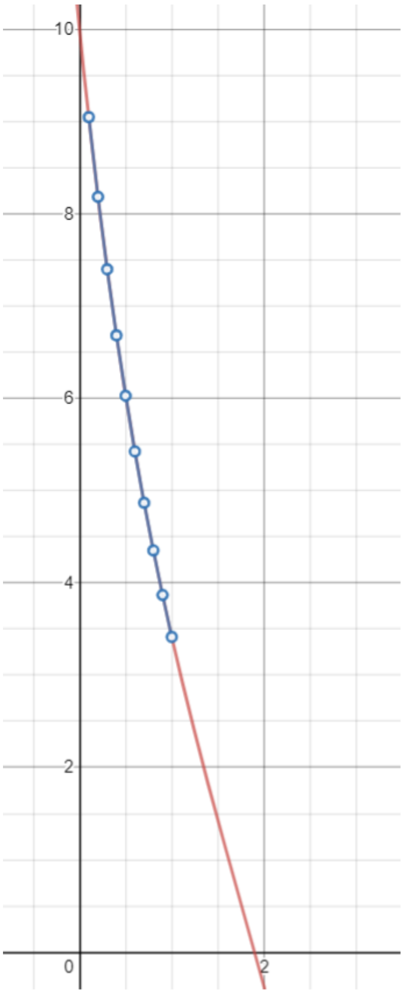
Тест	Результаты работы программы
$f_1(x, u, v) = x + v^2$ $f_2(x, u, v) = xu$ $(0, 1, -1)$ Segment: $[0, 1]$ $n = 10$	Runge-Kutt II x = 0.10000 u = 1.10500 v = -0.99450 x = 0.20000 u = 1.21781 v = -0.97684 x = 0.30000 u = 1.33588 v = -0.94466 x = 0.40000 u = 1.45641 v = -0.89552 x = 0.40000 u = 1.45641 v = -0.89552 x = 0.50000 u = 1.57656 v = -0.82698 x = 0.60000 u = 1.69374 v = -0.73671 x = 0.70000 u = 1.80605 v = -0.62262 x = 0.80000 u = 1.91274 v = -0.48282 x = 0.90000 u = 2.01483 v = -0.31558 x = 1.00000 u = 2.11571 v = -0.11918 Runge-Kutt IV x = 0.10000 u = 1.10465 v = -0.99466 x = 0.20000 u = 1.21707 v = -0.97716 x = 0.30000 u = 1.33472 v = -0.94517 x = 0.40000 u = 1.45479 v = -0.89626 x = 0.50000 u = 1.57445 v = -0.82799 x = 0.60000 u = 1.69113 v = -0.73807 x = 0.70000 u = 1.80292 v = -0.62439 x = 0.80000 u = 1.90910 v = -0.48507 x = 0.90000 u = 2.01066 v = -0.31837 x = 1.00000 u = 2.11096 v = -0.12252

Результаты работы программы

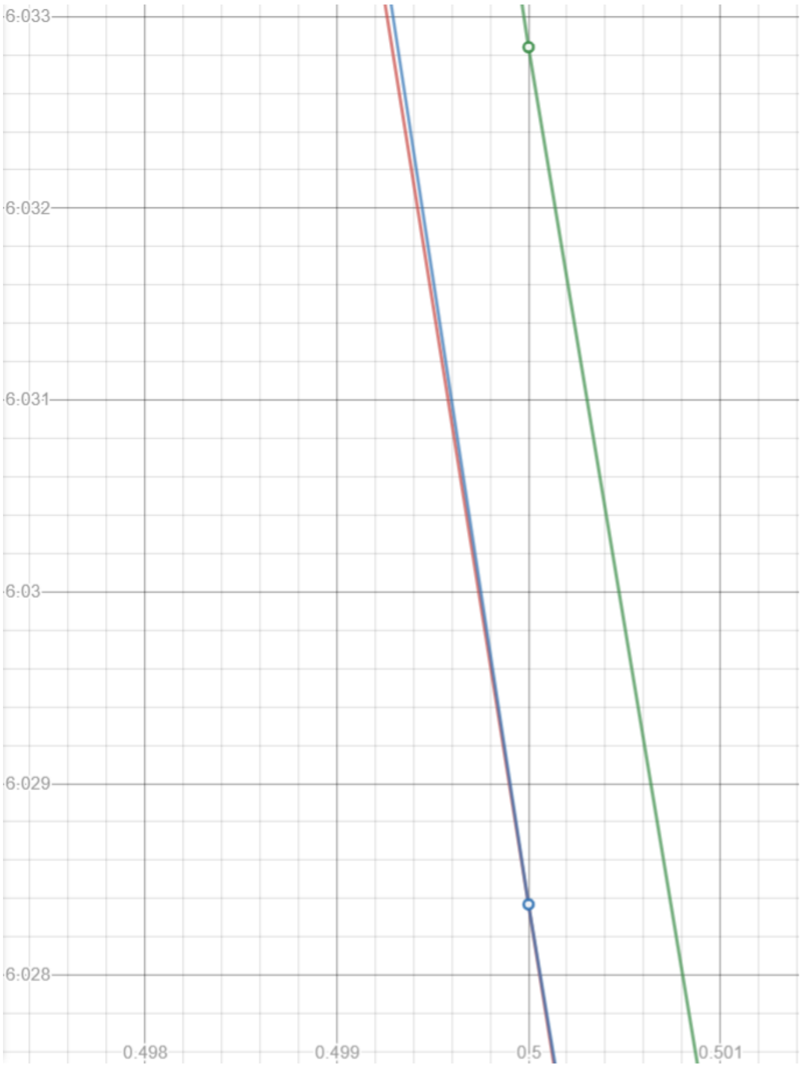
ОДУ	Точное решение	Результаты работы программы
$f(x, y) = -y - x^2$ $(0, 10)$ Segment: $[0, 1]$ $n = 10$	$y = -x^2 + 2x - 2 + 12e^{-x}$ y = 9.04805 y = 8.18477 y = 7.39982 y = 6.68384 y = 6.02837 y = 5.42574 y = 4.86902 y = 4.35195 y = 3.86884 y = 3.41455	Runge-Kutt II: x = 0.10000 y = 9.04950 x = 0.20000 y = 8.18735 x = 0.30000 y = 7.40325 x = 0.40000 y = 6.68789 x = 0.50000 y = 6.03284 x = 0.60000 y = 5.43047 x = 0.70000 y = 4.87388 x = 0.80000 y = 4.35681 x = 0.90000 y = 3.87361 x = 1.00000 y = 3.41917 Runge-Kutt IV x = 0.10000 y = 9.04805 x = 0.20000 y = 8.18477 x = 0.30000 y = 7.39982 x = 0.40000 y = 6.68384 x = 0.50000 y = 6.02837 x = 0.60000 y = 5.42574 x = 0.70000 y = 4.86903 x = 0.80000 y = 4.35195 x = 0.90000 y = 3.86884 x = 1.00000 y = 3.41456



● - решение методом Рунге-Кутта II порядка
● - точное решение $y = -x^2 + 2x - 2 + 12e^{-x}$

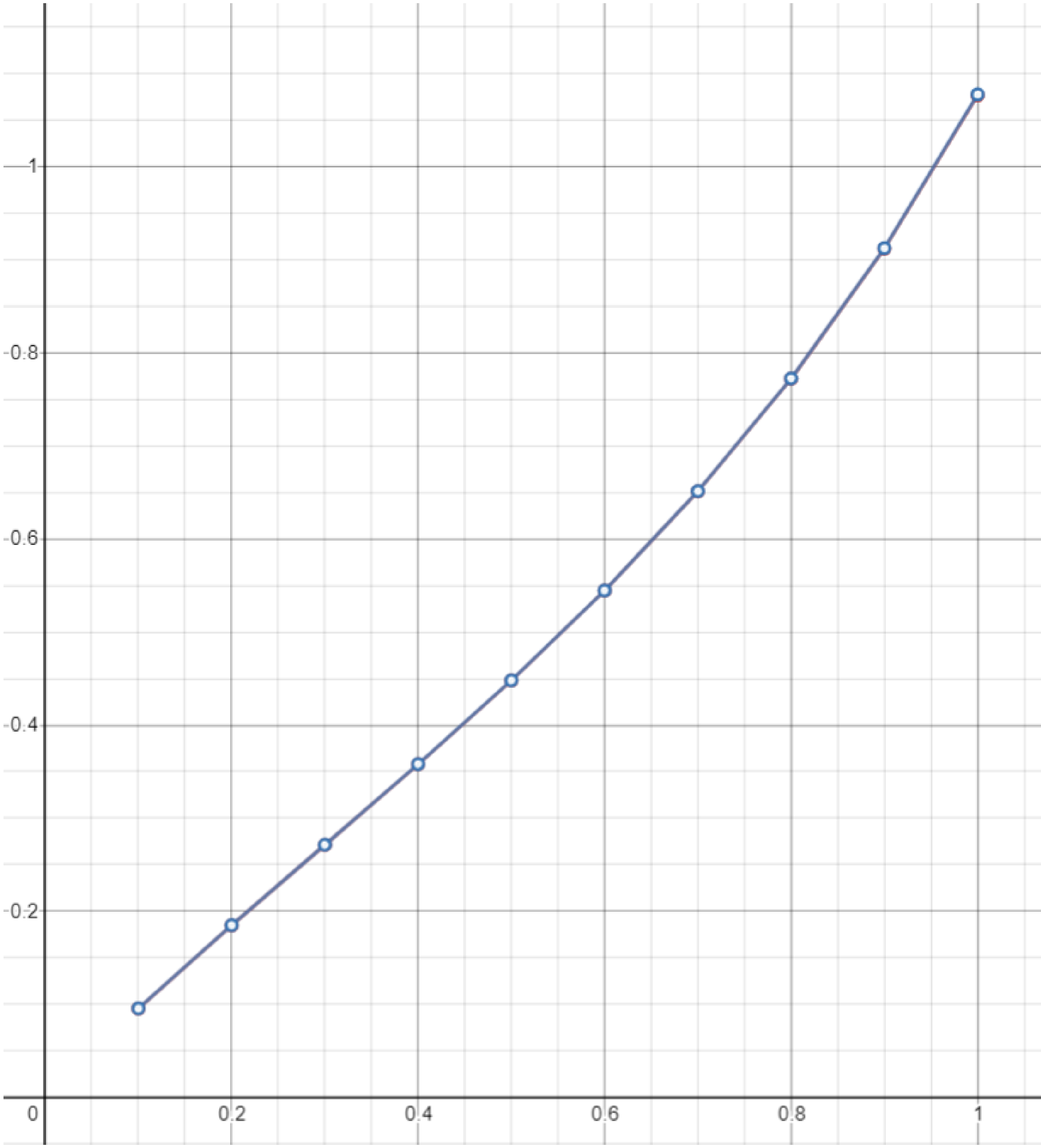


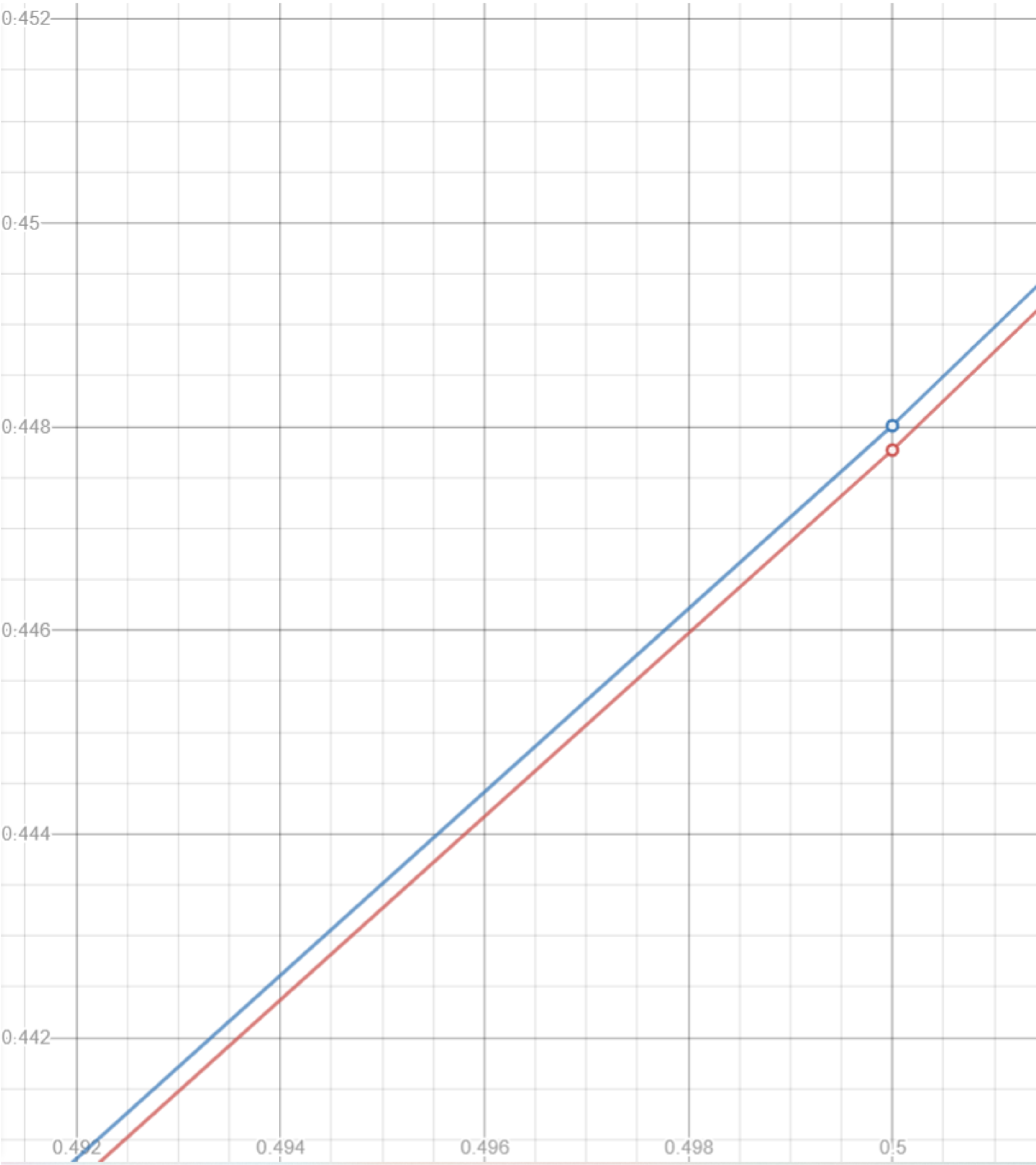
● - решение методом Рунге-Кутта IV порядка
● - точное решение $y = -x^2 + 2x - 2 + 12e^{-x}$


6.033
6.032
6.031
6.03
6.029
6.028
0.498
0.499
0.5
0.501

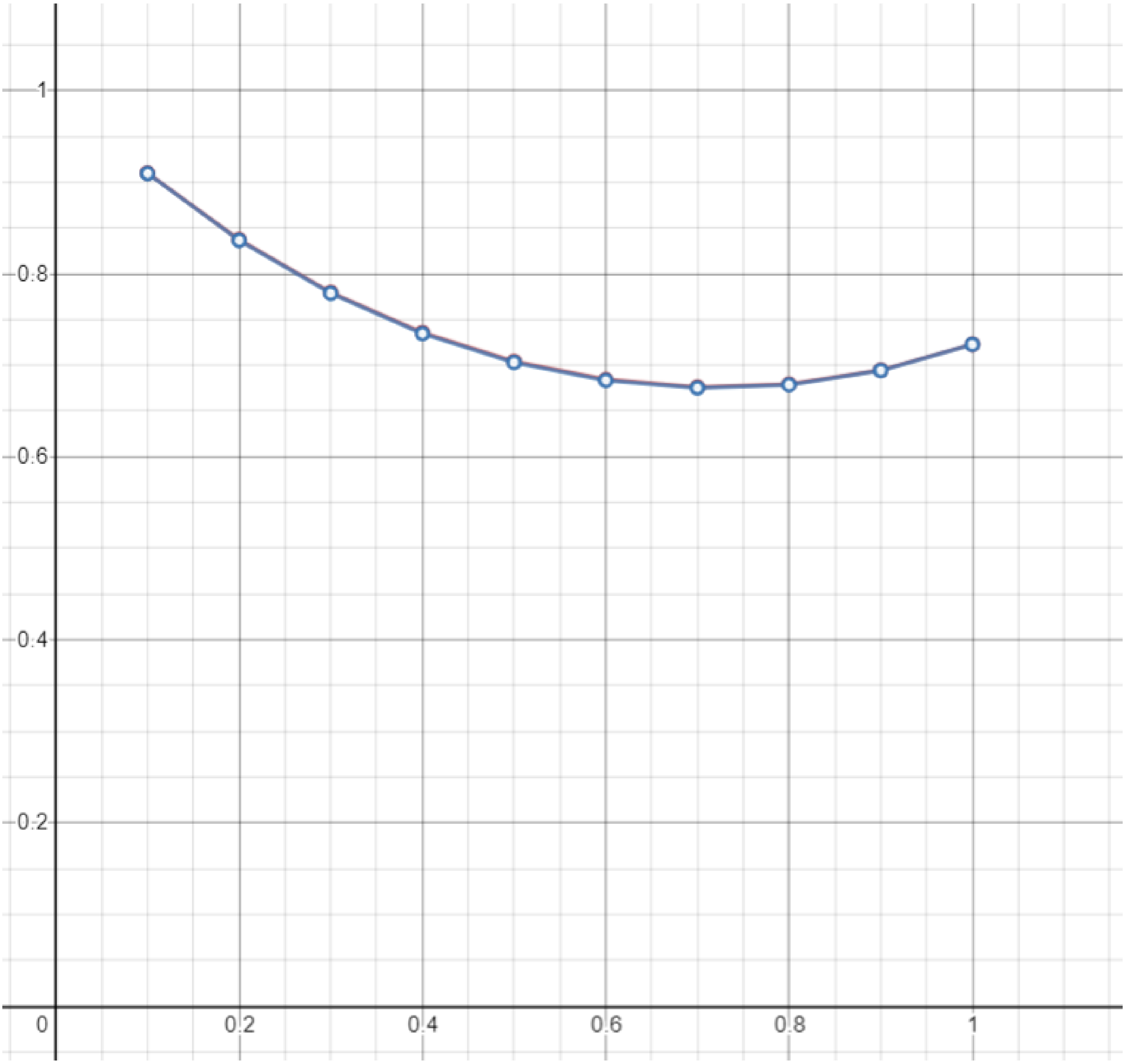
● - решение методом Рунге-Кутта II порядка
● - решение методом Рунге-Кутта IV порядка
● - точное решение $y = -x^2 + 2x - 2 + 12e^{-x}$

Система ОДУ	Результаты работы программы
$f_1(x, u, v) = sin(x+u)-1.1v$ $f_2(x, u, v) = 2.5u - (x+v)^2$ $(0, 0.5, 1)$ Segment: $[0, 1]$ $n = 10$	Runge-Kutt II x = 0.10000 u = 0.43821 v = 1.00396 x = 0.20000 u = 0.38145 v = 0.97418 x = 0.30000 u = 0.33351 v = 0.91823 x = 0.40000 u = 0.29758 v = 0.84365 x = 0.50000 u = 0.27608 v = 0.75733 x = 0.60000 u = 0.27068 v = 0.66523 x = 0.70000 u = 0.28218 v = 0.57227 x = 0.80000 u = 0.31046 v = 0.48237 x = 0.90000 u = 0.35437 v = 0.39840 x = 1.00000 u = 0.41178 v = 0.32219 Runge-Kutt IV x = 0.10000 u = 0.43896 v = 1.00571 x = 0.20000 u = 0.38275 v = 0.97735 x = 0.30000 u = 0.33518 v = 0.92243 x = 0.40000 u = 0.29943 v = 0.84851 x = 0.50000 u = 0.27796 v = 0.76255 x = 0.60000 u = 0.27246 v = 0.67052 x = 0.70000 u = 0.28375 v = 0.57742 x = 0.80000 u = 0.31174 v = 0.48718 x = 0.90000 u = 0.35533 v = 0.40271 x = 1.00000 u = 0.41242 v = 0.32587


1
0.8
0.6
0.4
0.2
0
0.2
0.4
0.6
0.8
1



• - $u(x)$, посчитанная с помощью метода Рунге-Кутта II порядка
• - $u(x)$, посчитанная с помощью метода Рунге-Кутта IV порядка



1
0.8
0.6
0.4
0.2
0
0.2
0.4
0.6
0.8
1

● - $v(x)$, посчитанная с помощью метода Рунге-Кутта II порядка
● - $v(x)$, посчитанная с помощью метода Рунге-Кутта IV порядка

Выводы

В данной работе был реализован метод Рунге-Кутта II и IV порядка точности для решения ОДУ первого порядка и системы из двух ОДУ первого порядка.

По результатам работы программы, которая представлена в данной работе, продемонстрировано построение графика на основании результатов численных просчетов конечно-разностных уравнений метода Рунге-Кутта. Решения, полученные с помощью метода Рунге-Кутта II и Рунге-Кутта IV в данном случае оказались достаточно близки к точному решению. По результатам построения заметим, что график, полученный с помощью метода Рунге-Кутта IV порядка, имеет меньшее отклонение от графика точного решения уравнения, чем график, полученный при использовании метода Рунге-Кутта II порядка.

Таким образом, метод Рунге-Кутта IV порядка дает более точные результаты для решения ОДУ первого порядка. Однако также необходимо отметить, что данный метод реализуется сложнее чем метод Рунге-Кутта II порядка.

Глава 2

Решение краевой задачи для обыкновенного дифференциального уравнения второго порядка, разрешенного относительно старшей производной

Постановка задачи

Рассматривается линейное дифференциальное уравнение II порядка вида:

$$y'' + p(x)y' + q(x)y = f(x) \tag{2.1}$$

с дополнительными условиями в граничных точках:

$$\begin{cases} \alpha_0 y(0) + \alpha_1 y'(0) = A \\ \beta_0 y(0) + \beta_1 y'(0) = B \end{cases} \tag{2.2}$$

Задачи

1. Решить краевую задачу методом конечных разностей, аппроксимировав ее разностной схемой второго порядка точности (на равномерной сетке), полученную систему конечно-разностных уравнений решить методом прогонки.

2. Найти разностное решение и построить его график.

Метод и алгоритм решения

Рассмотрим линейное дифференциальное уравнение II порядка вида:

$$y'' + p(x)y' + q(x)y = f(x) \tag{2.3}$$

с дополнительными условиями в граничных точках:

$$\begin{cases} \alpha_0 y(a) + \alpha_1 y'(a) = A \\ \beta_0 y(b) + \beta_1 y'(b) = B \end{cases} \tag{2.4}$$

Пусть на отрезке $[a,b]$ задана равномерная сетка: $x_{i+1}-x_i=h=\frac{l}{n}, 0\le i\le n-1$.

В исходном уравнении заменим y'' и y' конечно-разностными отношениями:

$$y''=\frac{y_{i+1}-2y_i+y_{i-1}}{h^2} \qquad y'=\frac{y_{i+1}-y_{i-1}}{2h},\ 0\le i\le n-1 \tag{2.5}$$

В соответствующих точках получим уравнение:

$$\frac{y_{i+1}-2y_i+y_{i-1}}{h^2}+p(x_i)\frac{y_{i+1}-y_{i-1}}{2h}+q(x_i)y_i=f(x_i),\text{ где } 0\le i\le n-1 \tag{2.6}$$

Запишем его в другом виде:

$$k_iy_{i-1}+l_iy_i+m_iy_{i+1}=f(x_i),\ 0\le i\le n-1 \tag{2.7}$$

где $k_i=\frac{1}{h^2}-\frac{p(x_i)}{2h}$, $m_i=\frac{1}{h^2}+\frac{p(x_i)}{2h}$, $l_i=q(x_i)-\frac{2}{hh^2}$.

Аппроксимируем производные в дополнительных условиях:

$$\begin{cases}\alpha_0y_0+\alpha_1\frac{y_1-y_0}{h}=A\\ \beta_0y_n+\beta_1\frac{y_n-y_{n-1}}{h}=B\end{cases} \tag{2.8}$$

Заметим, что СЛАУ (2.7) имеет трехдиагональную матрицу, поэтому ее можно решить методом прогонки. Решения будем искать в виде $y_i=\xi_{i+1}y_{i+1}+\eta_{i+1}$. Прогоночные коэффиценты:

$$\begin{cases}\xi_{i+1}=\frac{-m_i}{k_i\xi_i+li}\\ \eta_{i+1}=\frac{f(x_i)-k_i\eta_i}{k_i\xi_i+li}\end{cases} \tag{2.9}$$

Из левого граничного условия (2.4) получим: $\xi_1=\frac{\alpha_1}{\alpha_1-\alpha_0h}$ и $\eta_1=\frac{Ah}{\alpha_0h-\alpha_1}$.

Структура программы и спецификация функций

Программа состоит из одного модуля RUNGEKUTT.c. В данном модуле реализованы функции, описанные ниже:

Спецификация функций:

- ```
  double p(double x);
  double q(double x);
  double f(double x);
  double y(double x);
  ```

  Функции вычисляют значения функций $p(x)$, $q(x)$, $f(x)$ и $y(x)$ (решение дифференциального уравнения) соответственно в точке x для заданного в варианте 2 дифференциального уравнения.

- ```
  double test1_p(double x);
  double test1_q(double x);
  double test1_f(double x);
  double test1_y(double x);
  double test2_p(double x);
  double test2_q(double x);
  double test2_f(double x);
  double test2_y(double x);
  double test3_p(double x);
  double test3_q(double x);
  double test3_f(double x);
  double test3_y(double x);
  ```

  Функции вычисляют значения функций $p(x)$, $q(x)$, $f(x)$ и $y(x)$ (решение дифференциального уравнения) соответственно в точке x для дифференциальных уравнений в тестах 1-3.

- ```
  void solve_diff(int n, double h, double x0, double alpha0, double alpha1, double A,
   double beta0, double beta1, double B, double (*p)(), double (*q)(), double (*f)(), doubl
  ```

  Функция, вычисляющая прогоночные коэффиценты, осуществляющая метод конечных разностей и выводящая полученные значения функции в точках равномерной сетки.

- ```
  int main(int argc, char *argv[])
  ```

  Основная функция программы. При вводе в командную строку ключа test программа переходит в режим тестировки и далее работает с тестами 1-3 (см раздел Тестирование программы). Иначе программа работает с дифференциальным уравнением из варианта 2 задания.

Программа на СИ

```
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <inttypes.h>
#include <limits.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double test1_p(double x) {
    return 0.0;
}

double test1_q(double x) {
    return -1.0;
}

double test1_f(double x) {
    return 0.0;
}

double test1_y(double x) {
    return -exp(1 - x);
}

double test2_p(double x) {
    return 2.0;
}

double test2_q(double x) {
    return 1.0;
}

double test2_f(double x) {
    return 1/(x * exp(x));
}

double test2_y(double x) {
    return exp(-x) * (0.994824 -x + x * log(x));
}

double test3_p(double x) {
```

```
    return 1.0;
}

double test3_q(double x) {
    return 0.0;
}

double test3_f(double x) {
    return 1.0;
}

double test3_y(double x) {
    return x + exp(-x) - exp(-1);
}

double p(double x) {
    return -x;
}

double q(double x) {
    return 2.0;
}

double f(double x) {
    return x - 1;
}

double y(double x) {
    return (0.661683 - 0.661683 * x * x) * erf(x/sqrt(2)) - 1.45132* x * x + (0.527947 * exp(
}

void solve_diff(int n, double h, double x0, double alpha0, double alpha1, double A,
 double beta0, double beta1, double B, double (*p)(), double (*q)(), double (*f)(), double (*
    double x = x0 + h;
    double ksi[n], eta[n];
    ksi[0] = alpha1 / (alpha1 - alpha0 * h);
    eta[0] = A * h / (alpha0 * h - alpha1);
    double k, l, m;
    for (int i = 1; i < n; i++) {
        k = 1 / (h * h) - p(x) / (2 * h);
        l = q(x) - (2 / (h * h));
        m = 1 / (h * h) + p(x) / (2 * h);
        ksi[i] = -m / (k * ksi[i - 1] + l);
        eta[i] = (f(x) - k * eta[i - 1]) / (k * ksi[i - 1] + l);
        x += h;
    }
```

```
    double y = (beta1 * eta[n - 1] + B * h) / (beta0 * h + beta1 * (1 - ksi[n - 1]));
    printf("x = %.5lf y = %.5lf | y = %.5lf\n", x, y, ans(x));
    for (int i = n - 1; i >= 0; i--) {
        x -= h;
        y = ksi[i] * y + eta[i];
        printf("x = %.5lf y = %.5lf | y = %.5lf\n", x, y, ans(x));
    }
}

int main(int argc, char *argv[]){
    if (argc > 1 && argv[1][0] == 't') {
        printf("Enter n:");
        int n;
        scanf("%d", &n);
        printf("Test1:\n");
        printf("Equation: y'' - y = 0\n");
        printf("y(0)+y'(0) = 0\n");
        printf("y'(1) = 1\n");
        printf("Segment: [0;1]\n");
        double h = 1;
        h = h / n;
        solve_diff(n, h, 0, 1, 1, 0, 0, 1, 1, test1_p, test1_q, test1_f, test1_y);
        printf("Test2:\n");
        printf("Equation: y'' + 2y' + y = exp(-x)/x\n");
        printf("y(0) = 0\n");
        printf("y(1)+y'(1) = 0\n");
        printf("Segment: [0,5;1]\n");
        h = 0.1;
        h = h / n;
        solve_diff(n, h, 0.9, 1, 0, 0, 1, 1, 0, test2_p, test2_q, test2_f, test2_y);
        printf("Test3:\n");
        printf("Equation: y'' + y' = 1\n");
        printf("y'(0) = 0\n");
        printf("y'(1) = 1\n");
        printf("Segment: [0;1]\n");
        h = 1;
        h = h / n;
        solve_diff(n, h, 0, 0, 1, 0, 1, 0, 1, test3_p, test3_q, test3_f, test3_y);
        return 0;
    }
    int n;
    printf("Enter n:");
    scanf("%d", &n);
    double a_seg, b_seg, alpha0, alpha1, A, beta0, beta1, B;
    printf("Enter segment [a,b]: ");
    scanf("%lf%lf", &a_seg, &b_seg);
```

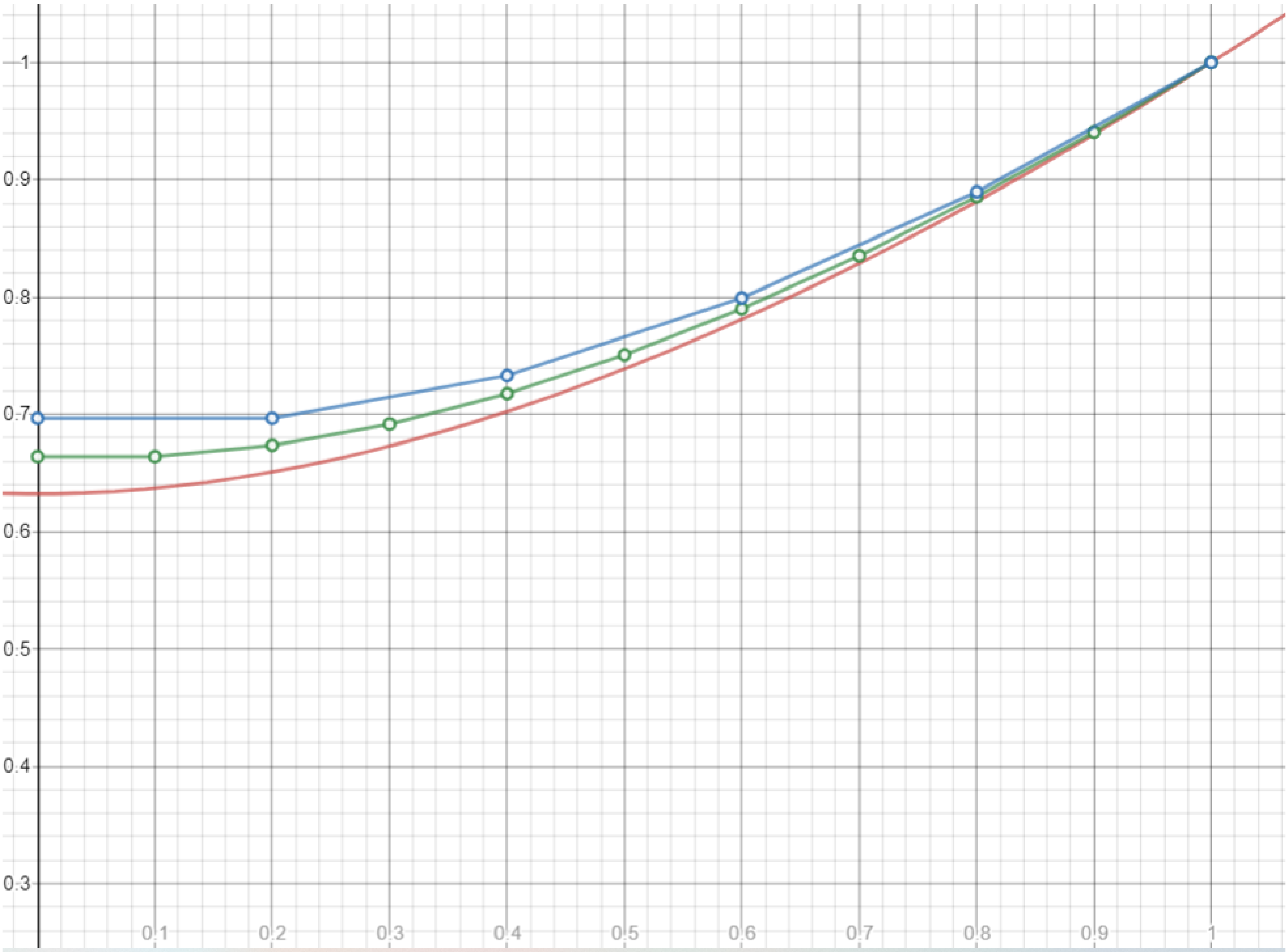
```
    double h = (b_seg - a_seg) / n;
    printf("Enter coefficents a_0, a_1, A, b_0, b_1, B for boundary conditions:\n");
    printf("a_0y(a) + a_1y'(a) = A\n");
    printf("b_0y(b) + b_1y'(b) = B\n");
    scanf("%lf %lf %lf %lf %lf %lf", &alpha0, &alpha1, &A, &beta0, &beta1, &B);
    solve_diff(n, h, a_seg, alpha0, alpha1, A, beta0, beta1, B, p, q, f, y);
    return 0;
}
```

Тестирование программы

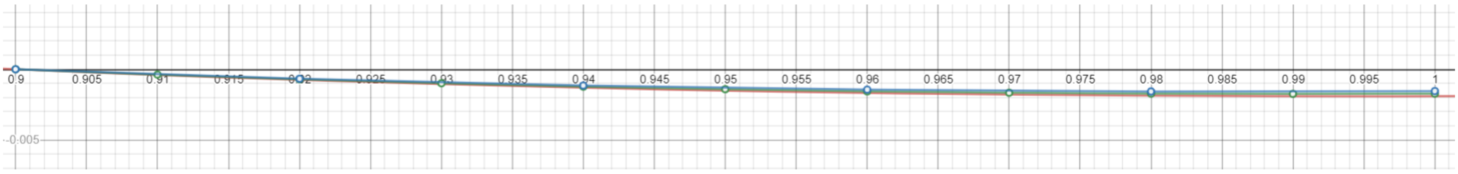
Тестирование программы осуществлялось на тестах 1-3 в режиме test.

Тест	Ожидаемые результаты	Результаты работы программы
$y'' - y = 0$ $y(0) + y'(0) = 0$ $y'(1) = 1$ n = 10	$y(x) = -e^{1-x}$ x = 1.00000 y = -1.00000 x = 0.90000 y = -1.10517 x = 0.80000 y = -1.22140 x = 0.70000 y = -1.34986 x = 0.60000 y = -1.49182 x = 0.50000 y = -1.64872 x = 0.40000 y = -1.82212 x = 0.30000 y = -2.01375 x = 0.20000 y = -2.22554 x = 0.10000 y = -2.45960 x = 0.00000 y = -2.71828	 x = 1.00000 y = -0.67687 x = 0.90000 y = -0.77687 x = 0.80000 y = -0.88464 x = 0.70000 y = -1.00125 x = 0.60000 y = -1.12788 x = 0.50000 y = -1.26579 x = 0.40000 y = -1.41635 x = 0.30000 y = -1.58108 x = 0.20000 y = -1.76162 x = 0.10000 y = -1.95977 x = 0.00000 y = -2.17752
$y'' + 2y' + y = \frac{e^{-x}}{x}$ $y(0) = 0$ $y(1) + y'(1) = 0$ n = 10	$y(x) = e^{-x}(0.994824 - x + xlnx)$ x = 1.00000 y = -0.00190 x = 0.99000 y = -0.00190 x = 0.98000 y = -0.00187 x = 0.97000 y = -0.00179 x = 0.96000 y = -0.00167 x = 0.95000 y = -0.00151 x = 0.94000 y = -0.00130 x = 0.93000 y = -0.00105 x = 0.92000 y = -0.00075 x = 0.91000 y = -0.00040 x = 0.90000 y = 0.00000	 x = 1.00000 y = -0.00172 x = 0.99000 y = -0.00174 x = 0.98000 y = -0.00172 x = 0.97000 y = -0.00166 x = 0.96000 y = -0.00156 x = 0.95000 y = -0.00141 x = 0.94000 y = -0.00123 x = 0.93000 y = -0.00099 x = 0.92000 y = -0.00071 x = 0.91000 y = -0.00038 x = 0.90000 y = 0.00000
$y'' + y' = 1$ $y'(0) = 0$ $y'(1) = 1$ n = 10	$y(x) = x + e^{-x} - e^{-1}$ x = 1.00000 y = 1.00000 x = 0.90000 y = 0.93869 x = 0.80000 y = 0.88145 x = 0.70000 y = 0.82871 x = 0.60000 y = 0.78093 x = 0.50000 y = 0.73865 x = 0.40000 y = 0.70244 x = 0.30000 y = 0.67294 x = 0.20000 y = 0.65085 x = 0.10000 y = 0.63696 x = 0.00000 y = 0.63212	 x = 1.00000 y = 1.00000 x = 0.90000 y = 0.94063 x = 0.80000 y = 0.88553 x = 0.70000 y = 0.83516 x = 0.60000 y = 0.79001 x = 0.50000 y = 0.75064 x = 0.40000 y = 0.71765 x = 0.30000 y = 0.69171 x = 0.20000 y = 0.67357 x = 0.10000 y = 0.66405 x = 0.00000 y = 0.66405
$y'' - y = 0$ $y(0) + y'(0) = 0$ $y'(1) = 1$ n = 5	$y(x) = -e^{1-x}$ x = 1.00000 y = -1.00000 x = 0.80000 y = -1.22140 x = 0.60000 y = -1.49182 x = 0.40000 y = -1.82212 x = 0.20000 y = -2.22554 x = 0.00000y = -2.71828	 x = 1.00000 y = -0.47487 x = 0.80000 y = -0.67487 x = 0.60000 y = -0.90187 x = 0.40000 y = -1.16494 x = 0.20000 y = -1.47461 x = 0.00000 y = -1.84326

Тест	Ожидаемые результаты	Результаты работы программы
$y'' + 2y' + y = \frac{e^{-x}}{x}$	$y(x) = e^{-x}(0.994824 - x + xlnx)$	
$y(0) = 0$	x = 1.00000 x = 1.00000	x = 1.00000 y = -0.00154
$y(1) + y'(1) = 0$	x = 0.98000 y = -0.00187	x = 0.98000 y = -0.00157
n = 5	x = 0.96000 y = -0.00167	x = 0.96000 y = -0.00144
	x = 0.94000 y = -0.00130	x = 0.94000 y = -0.00115
	x = 0.92000 y = -0.00075	x = 0.92000 y = -0.00067
	x = 0.90000 y = -0.00000	x = 0.90000 y = 0.00000
$y'' + y' = 1$	$y(x) = x + e^{-x} - e^{-1}$	
$y'(0) = 0$	x = 1.00000 y = 1.00000	x = 1.00000 y = 1.00000
$y'(1) = 1$	x = 0.80000 y = 0.88145	x = 0.80000 y = 0.88963
n = 5	x = 0.60000 y = 0.78093	x = 0.60000 y = 0.79917
	x = 0.40000 y = 0.70244	x = 0.40000 y = 0.73305
	x = 0.20000 y = 0.65085	x = 0.20000 y = 0.69669
	x = 0.00000 y = 0.63212	x = 0.00000 y = 0.69669



● - решение при разбиении n = 10
● - решение при разбиении n = 5
● - точное решение для уравнения $y'' + y' = 1$

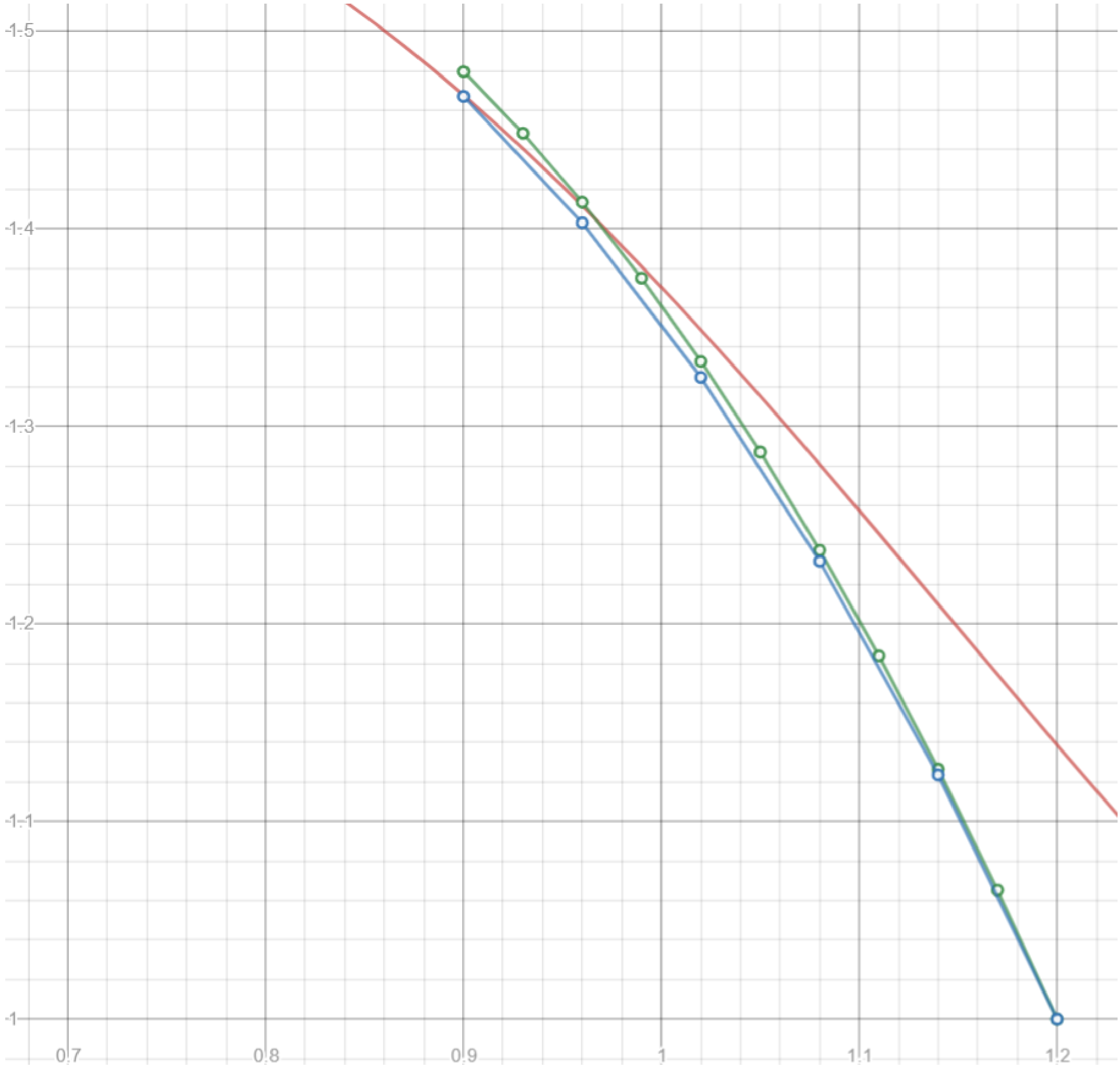


- - решение при разбиении n = 10
- - решение при разбиении n = 5
- - точное решение для уравнения $y'' + 2y' + y = \frac{e^{-x}}{x}$

Результаты работы программы

Краевая задача:
$y'' - xy' + 2y = x - 1$
$y(0.9) - 0.5y'(0.9) = 2$
$y(1.2) = 1$
Решение:
$y(x) = (0.661683 - 0.661683x^2)erfi(\frac{x}{\sqrt{2}}) - 1.45132x^2 + (0.527947e^{\frac{x^2}{2}} + 1)x + 0.951319$

Уравнение	Ожидаемые результаты	Результаты работы программы
$y'' - xy' + 2y = x - 1$	x = 1.20000 y = 1.13884	x = 1.20000 y = 1.00000
$y(0.9) - 0.5y'(0.9) = 2$	x = 1.17000 y = 1.17428	x = 1.17000 y = 1.06526
$y(1.2) = 1$	x = 1.14000 y = 1.21001	x = 1.14000 y = 1.12653
n = 10	x = 1.11000 y = 1.24565	x = 1.11000 y = 1.18388
	x = 1.08000 y = 1.28087	x = 1.08000 y = 1.23734
	x = 1.05000 y = 1.31536	x = 1.05000 y = 1.28698
	x = 1.02000 y = 1.34882	x = 1.02000 y = 1.33285
	x = 0.99000 y = 1.38101	x = 0.99000 y = 1.37499
	x = 0.96000 y = 1.41167	x = 0.96000 y = 1.41344
	x = 0.93000 y = 1.44060	x = 0.93000 y = 1.44827
	x = 0.90000 y = 1.46759	x = 0.90000 y = 1.47950
n = 5	x = 1.20000 y = 1.13884	x = 1.20000 y = 1.00000
	x = 1.14000 y = 1.21001	x = 1.14000 y = 1.12362
	x = 1.08000 y = 1.28087	x = 1.08000 y = 1.23173
	x = 1.02000 y = 1.34882	x = 1.02000 y = 1.32474
	x = 0.96000 y = 1.41167	x = 0.96000 y = 1.40304
	x = 0.90000 y = 1.46759	x = 0.90000 y = 1.46700



- - решение при разбиении n = 10
- - решение при разбиении n = 5
- - точное решение

Вывод

В данной работе был реализован способ решения краевой задачи методом конечных разностей, и полученная система конечно-разностных уравнений была решена методом прогонки.

На тестах было показано, что график построенный на основе решений, полученных с помощью данного метода отклоняется от графика точного решения, однако не очень значительно.

Также на тестах было показано, что точность метода растет с увеличением числа разбиений.

Литература

[1] Костомаров Д.П., Фаворский А.П. Вводные лекции по численным методам: Учеб. Пособие. – М.: Университетская книга, Логос, 2006

[2] Баркалов К.А. Методы параллельных вычислений. Н. Новгород: Изд-во Нижегородского госуниверситета им. Н.И. Лобачевского, 2011.