МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В.Ломоносова

Факультет вычислительной математики и кибернетики

«ЧИСЛЕННЫЕ МЕТОДЫ»

ЗАДАНИЕ № 2.

Численные методы решения дифференциальных уравнений

ОТЧЕТ

о выполненном задании

студента 202 учебной группы факультета ВМК МГУ

Струкова Павла Вячеславовича

гор. Москва

2022 г.

Оглавление

Подвариант 1

Постановка задачи

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

$$\frac{dy}{dx} = f(x, y),$$

с дополнительным начальным условием, заданном в точке $x = x_0$ $(x_0 < x)$:

$$y(x_0) = y_0.$$

Предполагается, что правая часть первого уравнения – функция $f = f(x, y)$ такова, что гарантирует существование и единственность решения задачи Коши.

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

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

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

$$y_1(x_0) = y_1, y_2(x_0) = y_2.$$

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

Цели и задачи практической работы

1. Решить задачу Коши наиболее известными и широко используемыми на практике методами Рунге-Кутта второго и четвертого порядка точности, аппроксимировав дифференциальную задачу соответствующей разностной схемой (на равномерной сетке).
2. Найти численное решение задачи и построить его график.
3. Найденное численное решение сравнить с точным решением дифференциального уравнения.

Описание метода решения

Метод Рунге-Кутте второго порядка точности

Метод Рунге-Кутте для численного решения задачи Коши на отрезке $[x_0; x_0 + l]$:

$$u'(x) = f\big(x, u(x)\big)$$
$$u(x_0) = u_0$$

Реализуем метод Рунге-Кутта второго порядка точности. Результатом работы алгоритма будет являться сеточная функция $y(x_i)$, определенная на сетке $x_i = x_0 + ih$, где $i \in 0, \dots, n$, h - фиксированный шаг. В нашем случае сетка равномерная и равна $h = \frac{l}{n}$, где n – параметр программы. Метод Рунге-Кутта второго порядка точности представляет нам рекуррентные формулы для вычисления сеточной функции y_i:

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

Сначала делается шаг h и по схеме Эйлера вычисляется значение:

$$\tilde{y}_{i+1} = y_i + f(x_i, y_i) \cdot h$$

Затем находится значение функции f в точке $(x_{i+1}, \tilde{y}_{i+1})$, составляется полусумма

$$\frac{f(x_i, y_i) + f(x_{i+1}, \tilde{y}_{i+1})}{2}$$

и окончательно:

$$y_{i+1} = y_i + \frac{f(x_i, y_i) + f(x_i, \tilde{y}_{i+1})}{2} h.$$

Метод Рунге-Кутте четвертого порядка точности

Второго порядка точности, как показывает практика, может быть недостаточно. Наиболее часто при проведении реальных расчетов используется схема Рунге-Кутта четвертого порядка точности. Метод определяется формулами:

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

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

Откуда получаем рекуррентную формулу:

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Описание программы

Данная программа позволяет найти решение задачи методами Рунге-Кутта второго и четвертого порядка точности.

Описание функций

1. Функция $\boldsymbol{void\ RK2(double\ (*function)(double, double), double\ a, double\ b, double\ n, double\ x\_0, double\ y\_0)}$ находит решение дифференциального уравнения методом Рунге-Кутте 2 порядка.

```
void RK2(double (*function)(double, double), double a, double b, double n, double x_0, double y_0)
{
    double x, y;
    double h = (b - a) / n;
    double f_prev, x_prev, y_prev = y_0;

    for (int i = 0; i < n; i++) {
        x = x_0 + h * (i + 1);
        x_prev = x_0 + h * i;
        f_prev = function(x_prev, y_prev);
        y = y_prev + (f_prev + function(x, y_prev + f_prev * h)) * h / 2;
        y_prev = y;

        printf("%lf %lf\n", x, y);
    }
}
```

2. Функция $\boldsymbol{void\ RK4(double\ (*function)(double, double), double\ a, double\ b, double\ n, double\ x\_0, double\ y\_0)}$ находит решение дифференциального уравнения методом Рунге-Кутте 4 порядка.

```
void RK4(double (*function)(double, double), double a, double b, double n, double x_0, double y_0)
{
    double x, y;
    double h = (b - a) / n;
    double x_prev, y_prev = y_0;
    double k_1, k_2, k_3, k_4;

    for (int i = 0; i < n; i++) {
        x = x_0 + h * (i + 1);
        x_prev = x_0 + h * i;

        k_1 = h * function(x_prev, y_prev);
        k_2 = h * function(x_prev + h / 2, y_prev + k_1 / 2);
        k_3 = h * function(x_prev + h / 2, y_prev + k_2 / 2);
        k_4 = h * function(x_prev + h, y_prev + k_3);

        y = y_prev + (k_1 + 2 * k_2 + 2 * k_3 + k_4) / 6;
        y_prev = y;

        printf("%lf %lf\n", x, y);

    }
}
```

3. Функция $\boldsymbol{void\ RK2\_s(double\ (*function\_1)(double, double, double), double\ (*function\_2)(double, double, double)}$,

$\boldsymbol{double\ a, double\ b, double\ n, double\ x\_0, double\ y1\_0, double\ y2\_0}$) находит решение системы уравнений методом Рунге-Кутте 2 порядка.

```
void RK2_s(double (*function_1)(double, double, double), double (*function_2)(double, double, double),
    double a, double b, double n, double x_0, double y1_0, double y2_0)
{
    double h = (b - a) / n;
    double x, u, v, f_u, f_v;
    double x_prev, u_prev = y1_0, v_prev = y2_0;

    for (int i = 0; i < n; i++) {
        x = x_0 + h * (i + 1);
        x_prev = x_0 + h * i;
        f_u = u_prev + h * function_1(x_prev, u_prev, v_prev);
        f_v = v_prev + h * function_2(x_prev, u_prev, v_prev);
        u = u_prev + h / 2 * (function_1(x_prev, u_prev, v_prev) + function_1(x, f_u, f_v));
        v = v_prev + h / 2 * (function_2(x_prev, u_prev, v_prev) + function_2(x, f_u, f_v));
        u_prev = u;
        v_prev = v;

        printf("%lf %lf %lf\n", x, v);
    }
}
```

4. Функция $\boldsymbol{void\ RK4\_s(double\ (*function\_1)(double, double, double), double\ (*function\_2)(double, double, double), double\ a, double\ b, double\ n, double\ x\_0, double\ y1\_0, double\ y2\_0}$) находит решение системы уравнений методом Рунге-Кутте 4 порядка.

```
void
RK4_s(double (*function_1)(double, double, double),
    double (*function_2)(double, double, double),
    double a, double b, double n, double x_0, double y1_0, double y2_0)
{
    double h = (b - a) / n;
    double x, u, v, f_u, f_v;
    double x_prev, u_prev = y1_0, v_prev = y2_0;
    double k_1, k_2, k_3, k_4, m_1, m_2, m_3, m_4;

    for (int i = 0; i < n; i++) {
        x = x_0 + h * (i + 1);
        x_prev = x_0 + h * i;

        k_1 = h * function_1(x_prev, u_prev, v_prev),
            m_1 = h * function_2(x_prev, u_prev, v_prev);

        k_2 = h * function_1(x_prev + h / 2, u_prev + k_1 / 2, v_prev + m_1 / 2);
        m_2 = h * function_2(x_prev + h / 2, u_prev + k_1 / 2, v_prev + m_1 / 2);

        k_3 = h * function_1(x_prev + h / 2, u_prev + k_2 / 2, v_prev + m_2 / 2);
        m_3 = h * function_2(x_prev + h / 2, u_prev + k_2 / 2, v_prev + m_2 / 2);

        k_4 = h * function_1(x_prev + h, u_prev + k_3, v_prev + m_3);
        m_4 = h * function_2(x_prev + h, u_prev + k_3, v_prev + m_3);

        u = u_prev + (k_1 + 2 * k_2 + 2 * k_3 + k_4) / 6;
        v = v_prev + (m_1 + 2 * m_2 + 2 * m_3 + m_4) / 6;

        u_prev = u;
        v_prev = v;

        printf("%lf %lf %lf\n", x, u, v);
    }
}
```

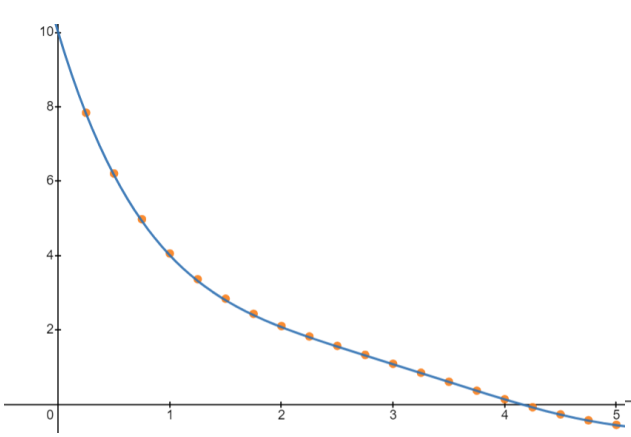
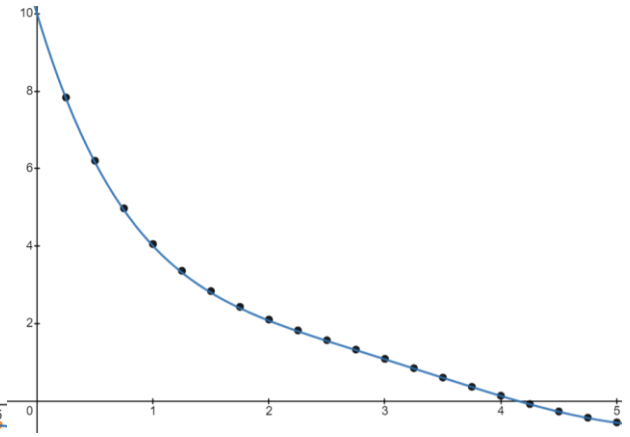
Тестирование программы

Проверка проводилась и для метода Рунге-Кутте 2 порядка, и 4 порядка. Слева для 2 порядка, справа для 4.

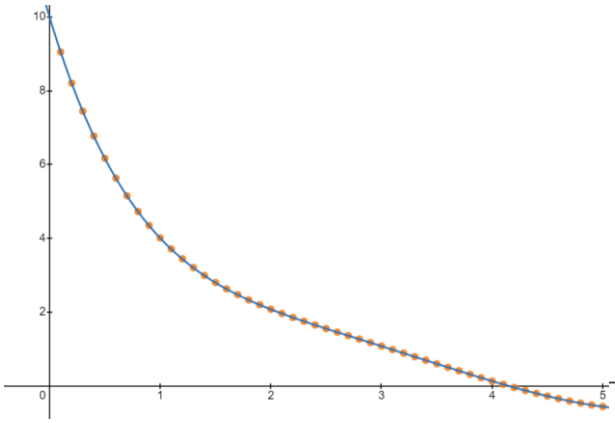
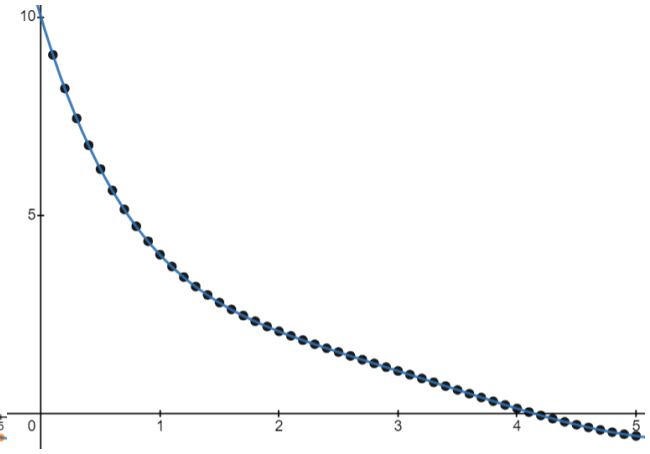
Тест 1. Таблица 1–2.

$\frac{dy}{dx} = sin(x) - y, y(0) = 10.$

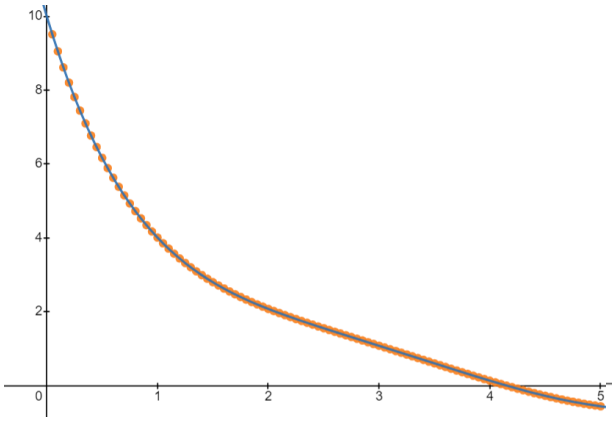
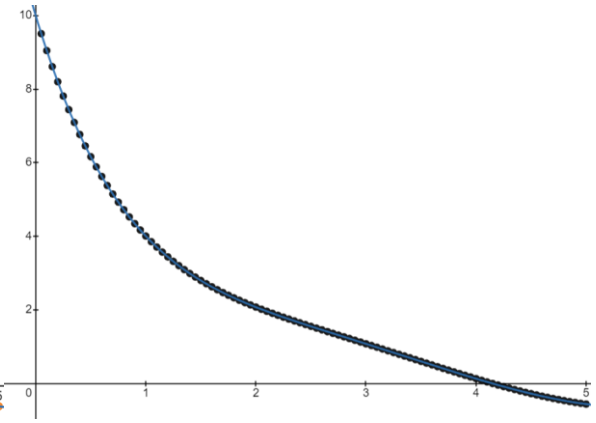
$n = 20$

$n = 50$

$n = 100$

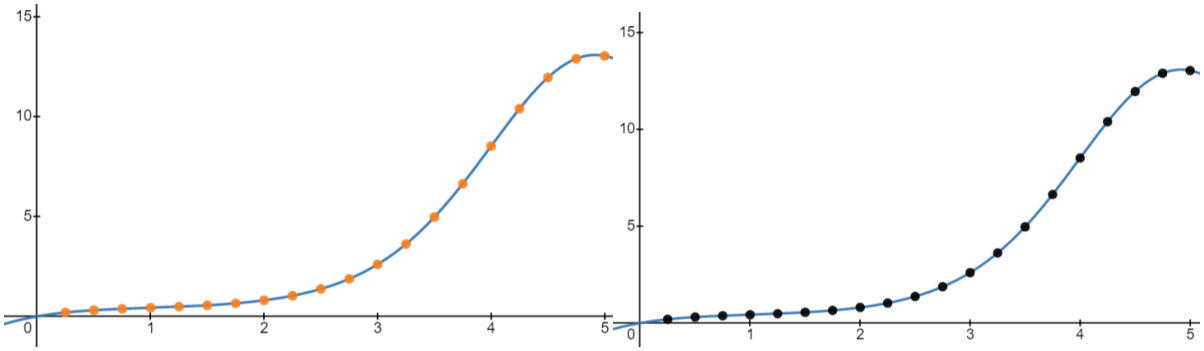



Тест 2.

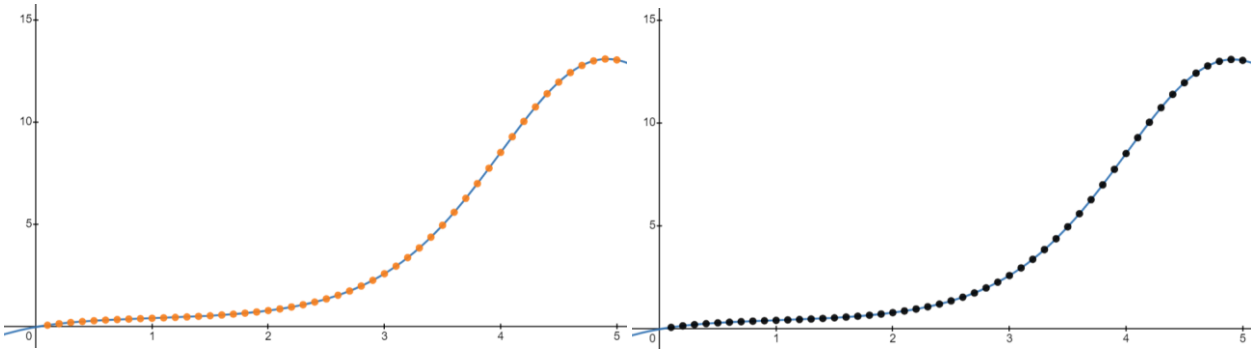
$\frac{dy}{dx} = e^{-sin(x)} - ycos(x), y(0) = 0.$

Точное решение $xe^{-sin(x)}$.

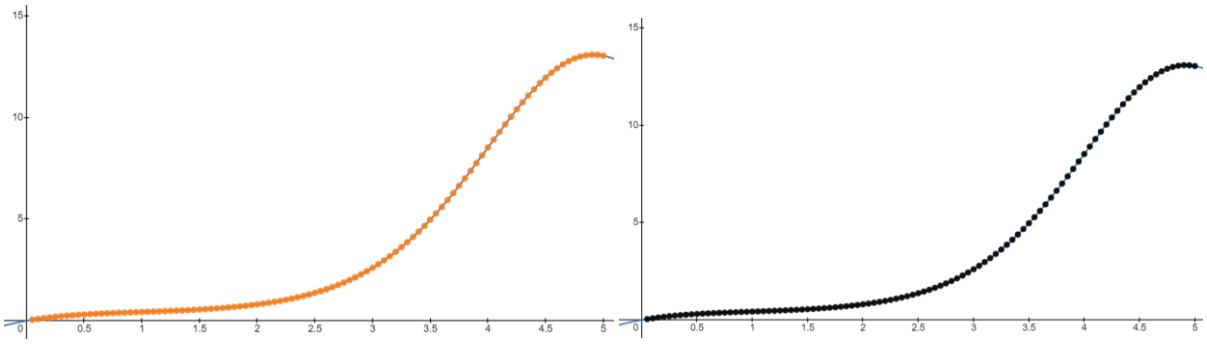
$n = 20$



$n = 50$

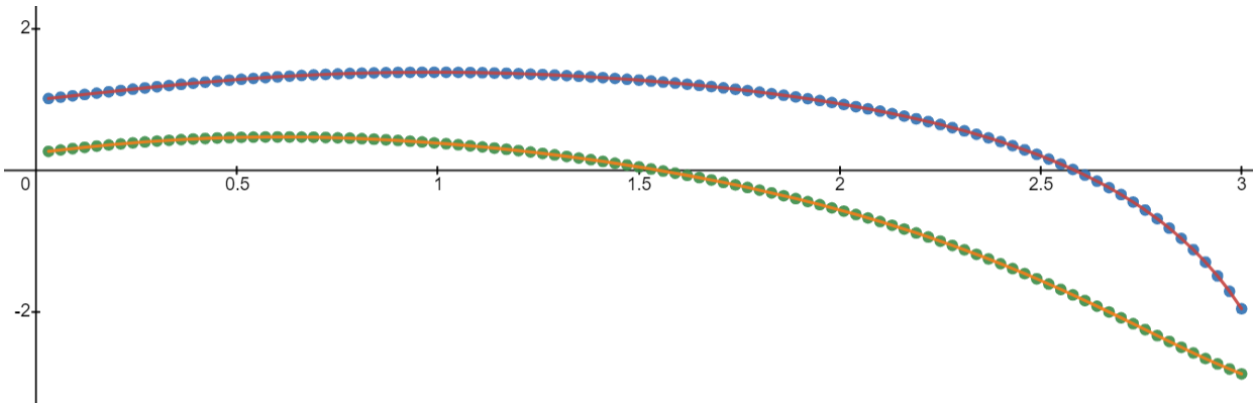


$n = 100$



Тест 3. Таблица 2–8.

$$\begin{cases} \frac{du}{dx} = \cos(x + 1.1 \cdot v) + u \\ \frac{dv}{dx} = -v^2 + 2.1 \cdot u + 1.1 \end{cases} \qquad u(0) = 0.25, v(0) = 1$$


2
0
0.5
1
1.5
2
2.5
3
-2

Вывод

В ходе работы освоены методы Рунге-Кутта второго и четвертого порядка точности, применяемые для численного решения задачи Коши для дифференциальных уравнений первого порядка и системы дифференциальных уравнений первого порядка. Полученные решения сопоставлены с точными решениями соответствующих задач.

Подвариант 2

Постановка задачи

Рассматривается дифференциальное уравнение второго порядка вида:

$$y'' + p(x) \cdot y' + q(x) \cdot y = -f(x), 0 < x < 1,$$

с дополнительными условиями в граничных точках

$$\begin{cases} \sigma_1 y(0) + \gamma_1 y'(0) = \delta_1, \\ \sigma_2 y(1) + \gamma_2 y'(1) = \delta_2. \end{cases}$$

Цели и задачи практической работы

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 задачи и построить его график;
3. Найденное разностное решение сравнить с точным решением дифференциального уравнения.

Описание метода решения

Построим равномерную метку на отрезке $[a, b]$ на n равноотстоящих узлов с некоторым шагом $h = \frac{b-a}{n}$. Разбиение абсциссы: $x_i = x_0 + ih$. Введем обозначения для функции для расчёта приближенных значений искомой функции $y(x)$ ее производных $y'(x), y''(x)$ в узлах сетки x_i: y_i, y'_i, y''_i. Также введем обозначение для значений функций p, q, f: p_i, q_i, f_i.

Заменим значения производных на их конечно-разностные отношения:

$$y'_i = \frac{y_{i+1} - y_i}{h}, y''_i = \frac{y_{i+2} + 2y_{i+1} - y_i}{h^2}$$

На концах отрезка пусть: $y'_0 = \frac{y_1 - y_0}{h}, y'_n = \frac{y_n - y_{n-1}}{h}$

Подставим в систему:

$$\begin{cases} \frac{y_{i+2} - 2y_{i+1} + y_i}{h^2} + p_i \cdot \frac{y_{i+1} - y_i}{h} + q_i \cdot y_i = -f_i \\ \sigma_1 y_0 + \gamma_1 \frac{y_1 - y_0}{h} = \delta_1 \\ \sigma_2 y_n + \gamma_2 \frac{y_n - y_{n-1}}{h} = \delta_2 \end{cases}$$

Получаем систему линейных алгебраических уравнений, матрица коэффициентов которой будет трехдиагональной. Первые n – 1 уравнений системы будут выглядеть так:

$$(\sigma_1 h - \gamma_1) y_0 + \gamma_1 y_1 = \delta_1 h$$
$$(1 - \frac{p_i h}{2}) y_{i-1} + (q_i h^2 - 2) y_i + (1 + \frac{p_i h}{2}) y_{i+1} = f_i h^2$$
$$-\gamma_2 y_{n-1} + (\sigma_2 h + \gamma_2) y_n = \delta_2 h$$

Систему можно решить с помощью метода прогонки.

Метод прогонки:

Рассматривается система вида

$$A_i y_{i-1} + C_i y_i + B_i y_{i+1} = F_i$$

Выражаем неизвестные из первого уравнения:

$$y_0 = \alpha_0 y_1 + \beta_0, \alpha_o = -\frac{c_0}{b_0}, \beta_0 = \frac{d_0}{b_0}$$

Подставляя выражение в систему на i-том шаге, получим:

$$y_i = \alpha_i y_i + \beta_i, \quad \alpha_i = -\frac{c_i}{a_i \alpha_i + b_i}, \quad \beta_i = \frac{d_i - a_i \beta_{i-1}}{a_i \alpha_i + b_i}$$

Для метода прогонки есть два хода: прямой ход и обратный ход.

- Прямой ход:

- Используя систему, выражаем коэффициенты a и b по формулам, приведенным выше.

- Обратный ход:
  - Полагаем, что $y_n = b_n$. Далее подстановками в формулу $y_i = \alpha_i y_{i+1} + \beta_i$. Находим $y_{n-1}, \ldots, y_0$.

Описание программы

Данная программа позволяет найти решение краевой задачи для дифференциального уравнения второго порядка.

Описание функций

1. Функции $\boldsymbol{double\ A(int\ i)}, \boldsymbol{double\ B(int\ i)}, \boldsymbol{double\ C(int\ i)}, \boldsymbol{double\ D(int\ i)}$ вычисляют коэффициенты для трехдиагональной матрицы.

```
double A(int i)
{
    if (i == n) {
        return -gamma2;
    }
    return 2 - h * p(a + h * i);
}

double B(int i)
{
    if (i == 0) {
        return h * sigma1 - gamma1;
    }

    if (i == n) {
        return h * sigma2 + gamma2;
    }
    return -4 + 2 * h * h * q(a + h * i);
}

double C(int i)
{
    if (i == 0) {
        return gamma1;
    }
    return 2 + h * p(a + h * i);
}

double D(int i)
{
    if (i == 0) {
        return h * delta1;
    }

    if (i == n) {
        return h * delta2;
    }

    return -2 * h * h * f(a + h * i);
}
```

2. Функция $void\ count\_ab(double * a, double * b, int\ n)$ вычисляет прогоночные коэффициенты.

```
void count_ab(double *a, double *b, int n)
{
    a[0] = -C(0) / B(0);
    b[0] = D(0) / B(0);
    double den;

    for (int i = 1; i < n; i++) {
        den = A(i) * a[i - 1] + B(i);

        a[i] = -C(i) / den;
        b[i] = (D(i) - A(i) * b[i - 1]) / den;
    }
}
```

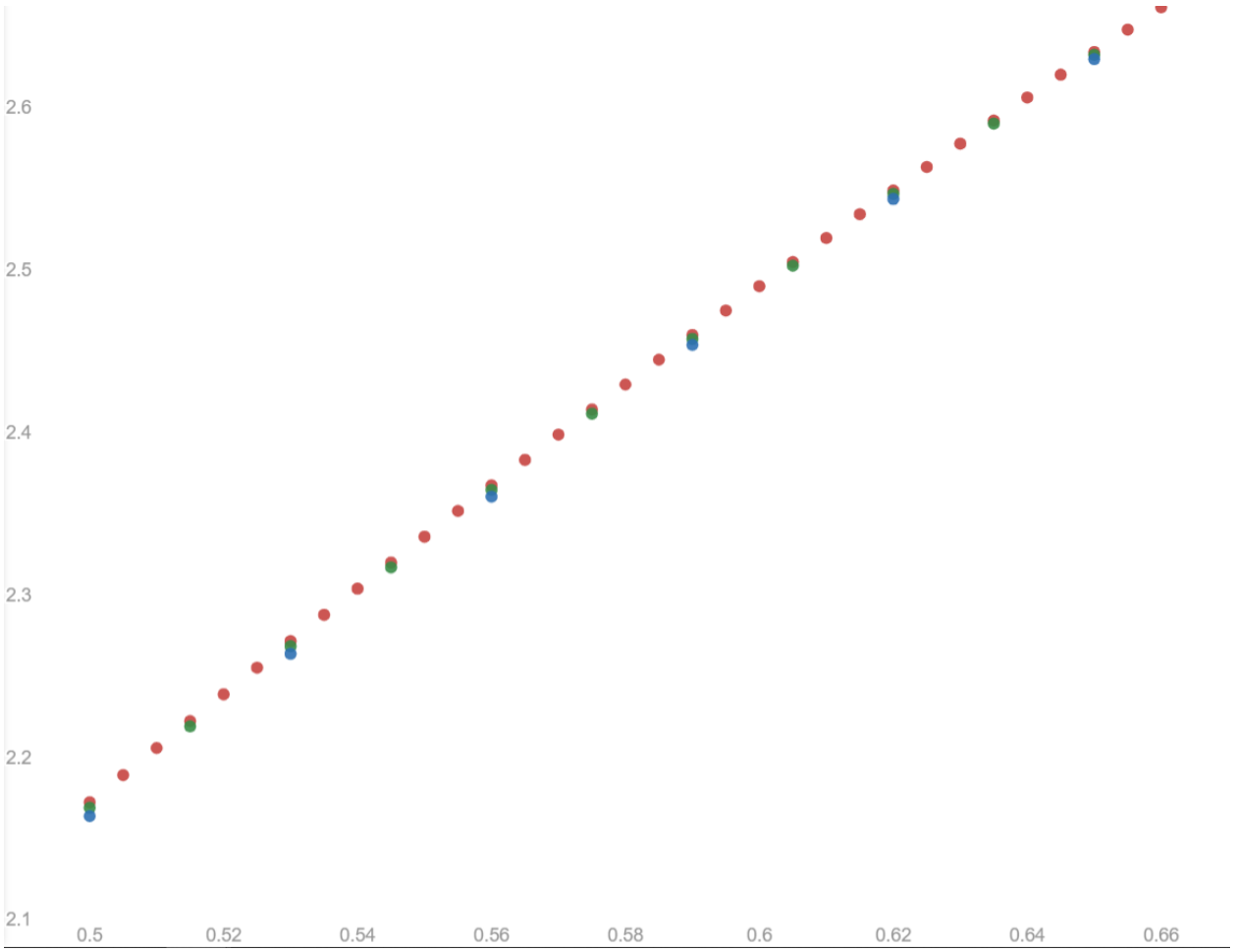
3. Функция $double * solution(double * a, double * b)$ возвращает массив полученных решений.

```
double *solution(double *a, double *b)
{
    double *y = (double *) malloc(sizeof(double) * (n + 1));

    y[n] = (D(n) - A(n) * b[n - 1]) / (A(n) * a[n - 1] + B(n));
    double den;

    for (int i = n - 1; i > 0; i--) {
        den = A(i) * a[i - 1] + B(i);

        y[i] = -C(i) / den * y[i + 1] + (D(i) - A(i) * b[i - 1]) / den;
    }

    y[0] = -C(0) / B(0) * y[1] + D(0) / B(0);

    return y;
}
```

Тестирование программы

Тест 1. Таблица 2-14.

$$y'' + 2x^2 y' + y = x\,,\ 2y(0.5) - y'(0.5) = 1\,,\ y(0.8) = 3$$



Для данного теста не удалось найти решение аналитически.

Синий: $n = 10$

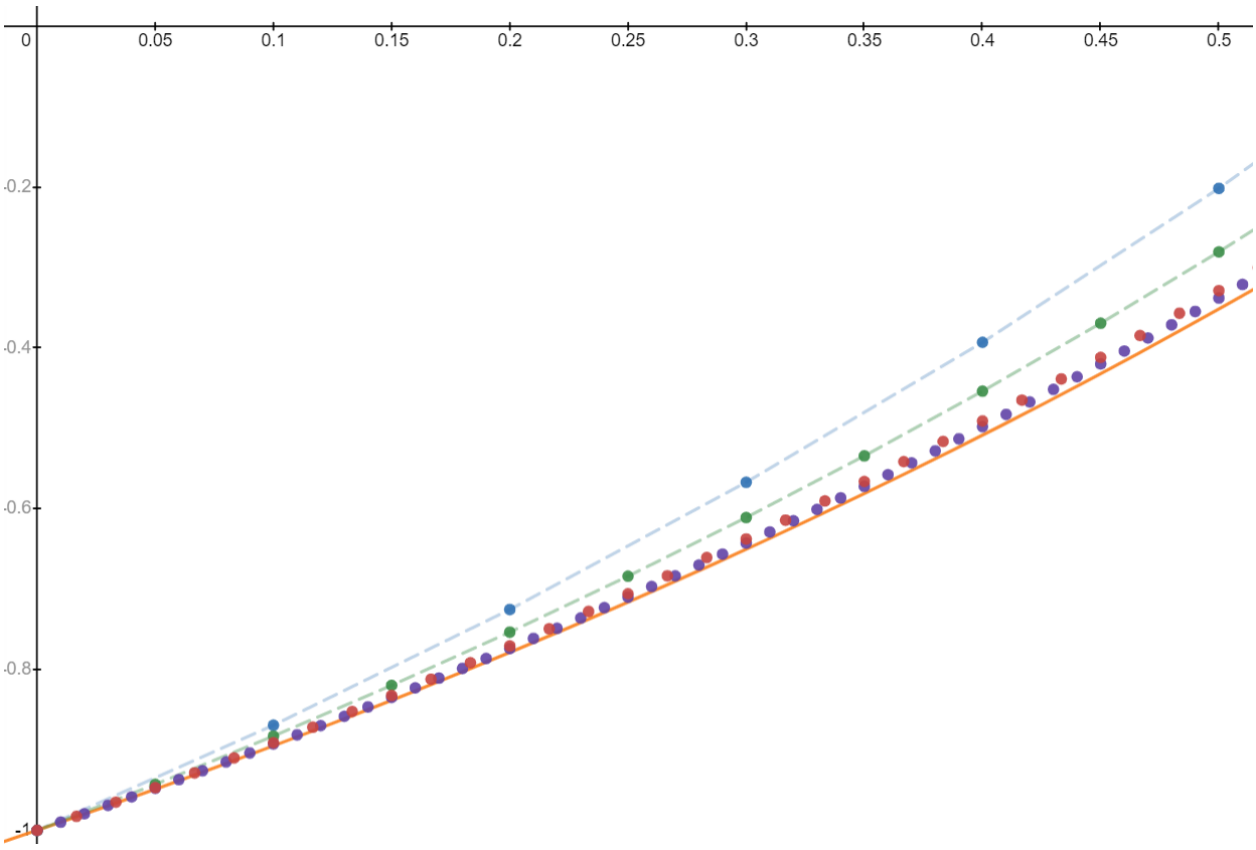
Зеленый: $n = 20$

Красный: $n = 60$

Тест 2.

$$\boldsymbol{y'' - y' = 0, y(0) = -1, y'(1) - y(1) = 2.}$$

Аналитическое решение: $y(x) = e^x - 2$.



Оранжневый: $y = e^x - 2$

Голубой: $n = 10$

Зеленый: $n = 20$

Красный: $n = 50$

Фиолетовый: $n = 100$

Вывод

В данной работе был реализован способ решения краевой задачи методом конечных разностей, и полученная система конечно-разностных уравнений была решена методом прогонки. Также на тестах было показано, что точность метода растет с увеличением числа разбиений.