Содержание

Подвариант 1

Цель работы

Освоить методы Рунге-Кутта второго и четвертого порядка точности, применяемые для численного решения задачи Коши для дифференциального уравнения (или системы дифференциальных уравнений) первого порядка.

Постановка задачи

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

$$\frac{dy}{dx} = f(x, y), \; x_0 < x, \tag{1}$$

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

$$y(x_0) = y_0. \tag{2}$$

Предполагается, что правая часть уравнения (1) функция $f = f(x, y)$ такова, что гарантирует существование и единственность решения задачи Коши (1)-(2).

В том случае, если рассматривается не одно дифференциальное уравнение вида (1), а система обыкновенных дифференциальных уравнений первого порядка, разрешенных относительно производных неизвестных функций, то соответствующая задача Коши имеет вид (на примере двух дифференциальных уравнений):

$$\begin{cases} \dfrac{dy_1}{dx} = f_1(x, y_1, y_2), \\ \dfrac{dy_2}{dx} = f_2(x, y_1, y_2), \; x > x_0. \end{cases} \tag{3}$$

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

$$y_1(x_0) = y_1^{(0)}, \; y_2(x_0) = y_2^{(0)}. \tag{4}$$

Также предполагается, что правые части уравнений из (3) заданы так, что это гарантирует существование и единственность решения задачи Коши (3)-(4), но уже для системы обыкновенных дифференциальных уравнений первого порядка в форме, разрешенной относительно производных неизвестных функций.

Цели и задачи практической работы

- Решить задачу Коши (1)-(2) (или (3)-(4)) методами Рунге-Кутта второго и четвертого порядка точности, аппроксимировав дифференциальную задачу соответствующей разностной схемой, полученное конечно-разностное уравнение (или уравнения в случае системы), представляющее фактически некоторую рекуррентную формулу, просчитать численно.
- Найти численное решение задачи и построить его график.
- Найденное численное решение сравнить с точным решением дифференциального уравнения

Описание методов решения

Рассмотрим задачу Коши для одного дифференциального уравнения $y' = f(x, y)$ с начальным условием $y(x_0) = y_0$ на отрезке $[x_0, x_0 + l]$ и его равномерную сетку из n точек:

$$x_{i+1} - x_i = h = \frac{l}{n}, 0 \leq i \leq n - 1$$

Поскольку решение может иметь производные высокого порядка, разложим его по формуле Тейлора и получим разностное уравнение:

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

Метод Рунге-Кутта второго порядка заключается в замене правой части уравнения на сумму значений функции f в двух разных точках с точностью до членов порядка h^2:

$$f(x_i, y_i) + \frac{h}{2} \cdot (\frac{\partial f}{\partial x}(x_i, y_i) + \frac{\partial f}{\partial x}(x_i, y_i) \cdot f(x_i, y_i)) = \alpha \cdot f(x_i + \gamma h, y_i + \delta h) + \beta \cdot f(x_i, y_i) + O(h^2)$$

где α, β, γ, $\delta -$ свободные параметры. Раскладывая функцию $f(x_i + \gamma h, y_i + \delta h)$ по Тейлору и подставляя разложение в исходное уравнение, получим следующие соотношения:

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

Подставив эти коэффициенты в правую часть и отбросив члены порядка $O(h^2)$, получим разностные схемы Рунге-Кутта:

$$\frac{y_{i+1} - y_i}{h} = (1 - \alpha) \cdot f(x_i, y_i) + \alpha \cdot f(x_i + \frac{h}{2\alpha}, y_i + \frac{h}{2\alpha} f(x_i, y_i))$$

Откуда получаем рекуррентное соотношение:

$$y_{i+1} = y_i + h((1 - \alpha) \cdot f(x_i, y_i) + \alpha \cdot f(x_i + \frac{h}{2\alpha}, y_i + \frac{h}{2\alpha} f(x_i, y_i)))$$

Одной из самых удобных схем для вычисления является подстановка $\alpha = 0.5$:

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

В случае решения задачи Коши для **системы дифференциальных уравнений**, эта формула также справедлива для любого y из вектора решений. Но гораздо более точный результат дает схема **Рунге-Кутта четвертого порядка** точности следующего вида:

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

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

В случае **системы дифференциальных уравнений** сопоставление схем проводится аналогичным образом. Используя эти рекуррентные формулы, можно последовательно рассчитать сеточную функцию и построить ее график.

Описание программ

1. Функции для тестов №1 и №2

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

double
f_answer_1(double x, double y)
{
    return (exp((-1) * x * x * (2 * x - 3) / 6));
}

double
f1(double x, double u, double v)
{
    return (cos(x + 1.5 * v) - u);
}

double
f2(double x, double u, double v)
{
    return ((-1) * v * v + 2.3 * u - 1.2);
}
```

2. Функция RK2 реализует метод Рунге-Кутта второго порядка

```c
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("x = %lf\ty = %lf\n", x, y);
    }
}
```

3. Функция RK4 реализует метод Рунге-Кутта четвертого 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("x = %lf\ty = %lf\n", x, y);
    }
}
```

4. Функция RK2_for_system реализует метод Рунге-Кутта второго порядка для решения систем

```
void
RK2_for_system(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("x = %lf\tu = %lf\tv = %lf\n", x, u, v);
    }
}
```

5. Функция RK4_for_system реализует метод Рунге-Кутта четвертого порядка для решения систем

```c
void
RK4_for_system(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("x = %lf\tu = %lf\tv = %lf\n", x, u, v);
    }
}
```

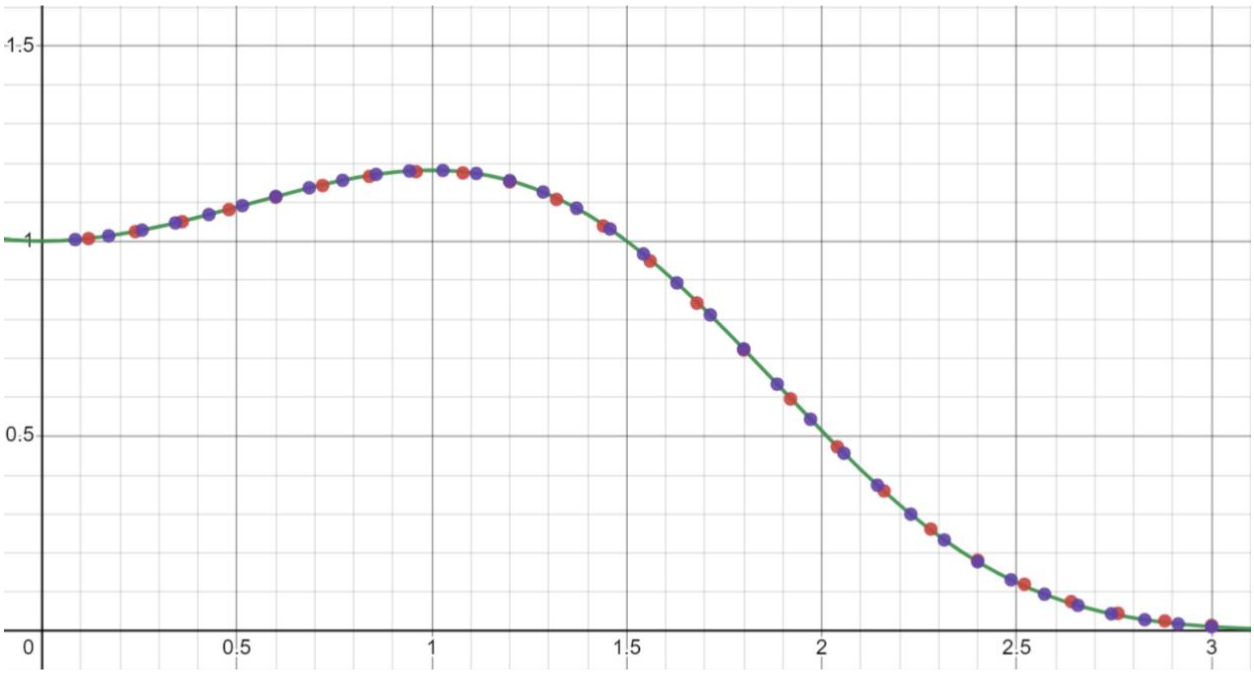
Тесты и проверка корректности

Тест №1 (таблица 1-6)

$$\begin{cases} y' = (x - x^2)y \\ y(0) = 1 \end{cases}$$

Точное решение: $y = e^{-\frac{1}{6}x^2(-3+2x)}$

По результатам работы программы построен график, который демонстрирует решение задачи Коши на отрезке [0, 3]: на нем отмечены точное решение зеленым цветом, решение методом Рунге-Кутта второго порядка красным цветом и решение методом Рунге-Кутта четвертого порядка фиолетовым цветом (по 30 точек). Заметно, что метод Рунге-Кутта четвертого порядка более точный, как и должно быть, и программа работает корректно.

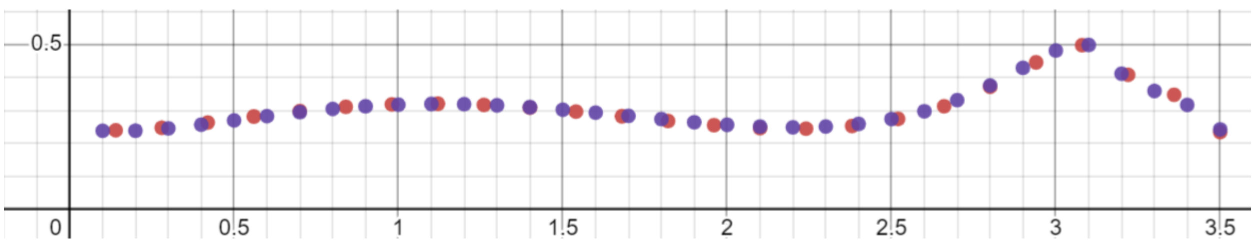


Тест №2 (таблица 2-18)

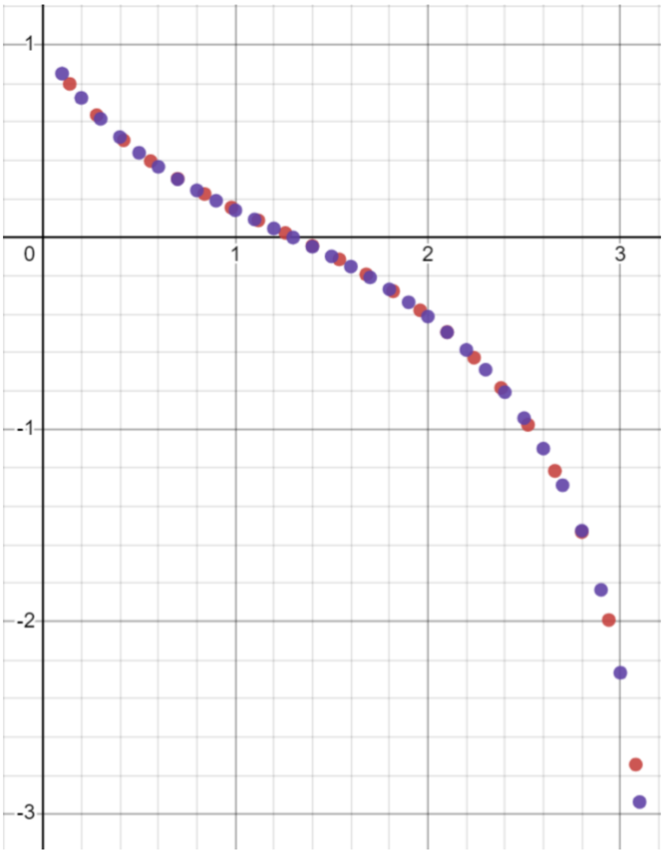
$$\begin{cases} \dfrac{dy_1}{dx} = cos(x + 1.5y_2) - y_1 \\ \dfrac{dy_2}{dx} = -y_2^2 + 2.3y_1 - 1.2 \\ y_1(0) = 0.25 \\ y_2(0) = 1 \end{cases}$$

По результатам работы программы построены 2 графика, которые демонстрируют решение задачи Коши на отрезке [0, 3.5] по 30 точек:

1. На первом графике красным цветом отмечены решения $y_1(x)$, полученные методом Рунге-Кутта второго порядка и фиолетовым цветом - методом Рунге-Кутта четвертого порядка.



2. На втором графике красным цветом отмечены решения $y_2(x)$, полученные методом Рунге-Кутта второго порядка и фиолетовым цветом - методом Рунге-Кутта четвертого порядка.



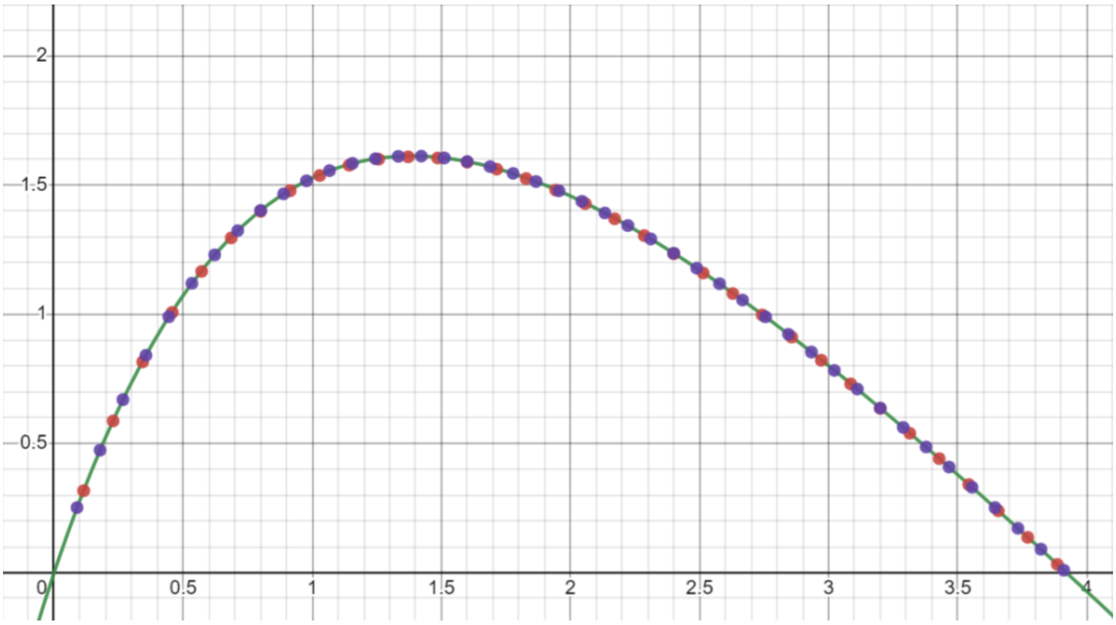
Также заметно, что метод Рунге-Кутта четвертого порядка более точный и программа работает корректно.

Тест №3 (дополнительный - таблица 1-1)

$$\begin{cases} y' = 3 - x - y \\ y(0) = 0 \end{cases}$$

Точное решение: $y = 4 - x - 4e^{-x}$

По результатам работы программы построен график, который демонстрирует решение задачи Коши на отрезке [0, 4]: на нем отмечены точное решение зеленым цветом, решение методом Рунге-Кутта второго порядка красным цветом и решение методом Рунге-Кутта четвертого порядка фиолетовым цветом (по 40 точек). Также заметно, что метод Рунге-Кутта четвертого порядка более точный и программа работает корректно.

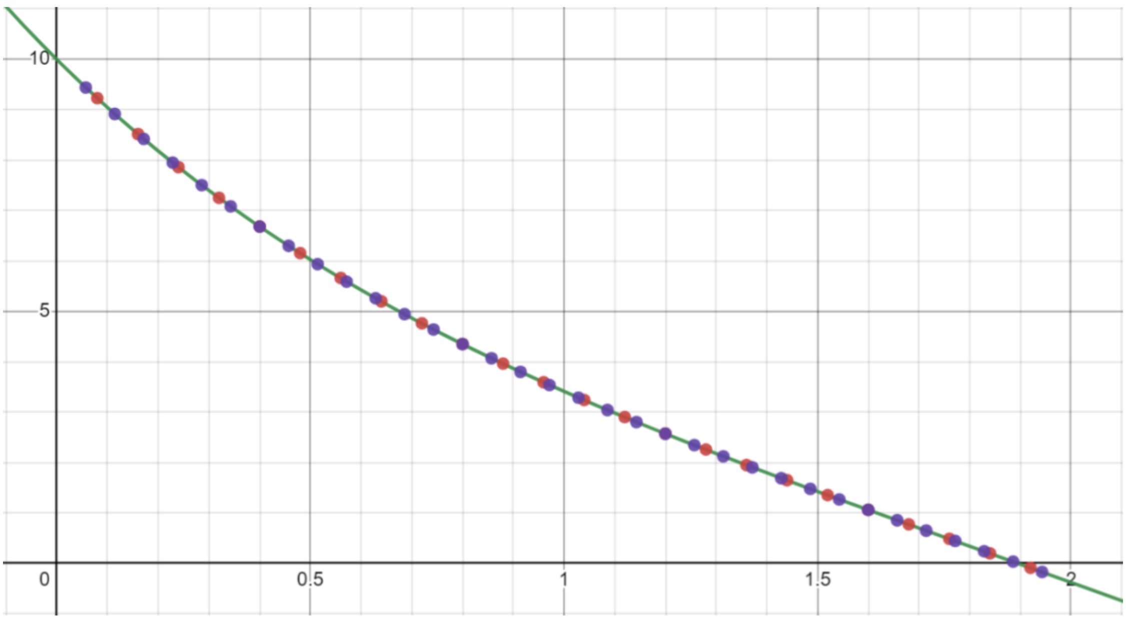


Тест №4 (дополнительный - таблица 1-3)

$$\begin{cases} y' = -y - x^2 \\ y(0) = 10 \end{cases}$$

Точное решение: $y = -x^2 + 2x - 2 + 12e^{-x}$

По результатам работы программы построен график, который демонстрирует решение задачи Коши на отрезке [0, 2]: на нем отмечены точное решение зеленым цветом, решение методом Рунге-Кутта второго порядка красным цветом и решение методом Рунге-Кутта четвертого порядка фиолетовым цветом (по 30 точек). Также заметно, что метод Рунге-Кутта четвертого порядка более точный и программа работает корректно.

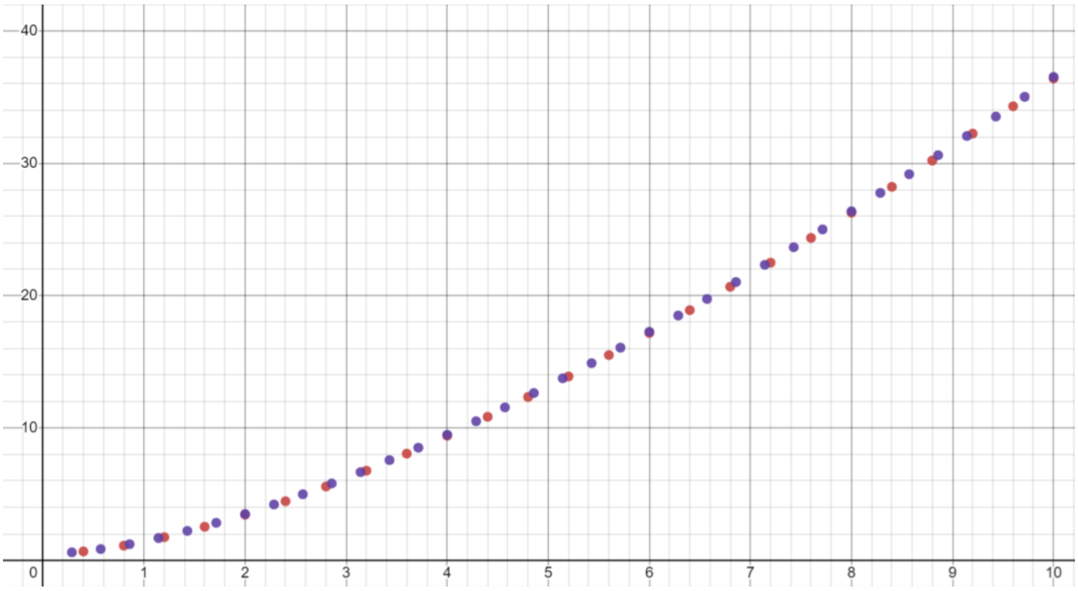


Тест №5 (дополнительный - таблица 2-13)

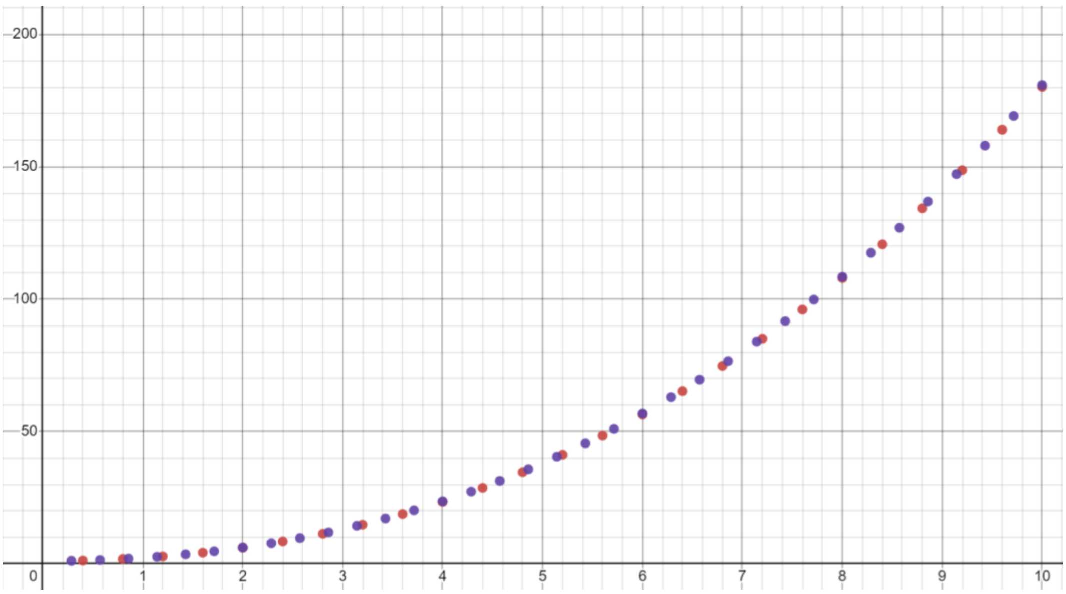
$$\begin{cases} \dfrac{dy_1}{dx} = ln(2x + \sqrt{4x^2 + y_2^2} \\ \dfrac{dy_2}{dx} = \sqrt{4x^2 + y_1^2} \\ y_1(0) = 0.5 \\ y_2(0) = 1 \end{cases}$$

По результатам работы программы построены 2 графика, которые демонстрируют решение задачи Коши на отрезке [0, 10] по 30 точек:

1. На первом графике красным цветом отмечены решения $y_1(x)$, полученные методом Рунге-Кутта второго порядка и фиолетовым цветом - методом Рунге-Кутта четвертого порядка.



2. На втором графике красным цветом отмечены решения $\boldsymbol{y_2(x)}$, полученные методом Рунге-Кутта второго порядка и фиолетовым цветом - методом Рунге-Кутта четвертого порядка.



Также заметно, что метод Рунге-Кутта четвертого порядка более точный и программа работает корректно.

Выводы

При выполнении практической работы были изучены и реализованы 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. Метод четвертого порядка точности может работать медленнее, но данное усложнение схемы окупается высокой точностью, что было продемонстрированно экспериментально: найдены численные решения задач Коши и построены их соответствующие графики. Полученные решения сопоставлены с точными решениями соответствующих задач.

Подвариант 2

Цель работы

Освоить метод прогонки решения краевой задачи для дифференциального уравнения второго порядка.

Постановка задачи

Рассматривается линейное дифференциальное уравнение второго порядка вида (1) с дополнительными условиями в граничных точках (2):

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

$$\begin{cases} \sigma_1 y(0) + \gamma_1 y'(0) = \delta_1, \\ \sigma_2 y(1) + \gamma_2 y'(1) = \delta_2. \end{cases} \qquad (2)$$

Цели и задачи практической работы

- Решить краевую задачу (1)-(2) методом конечных разностей, аппроксимировав ее разностной схемой второго порядка точности (на равномерной сетке); полученную систему конечно-разностных уравнений решить методом прогонки;
- Найти разностное решение задачи и построить его график;
- Найденное разностное решение сравнить с точным решением дифференциального уравнения

Описание методов решения

Осуществим поиск численного решения краевой задачи для одногодифференциального уравнения с дополнительными условиями в граничных точках:

$$\begin{cases} y'' + p(x)y' + q(x)y = -f(x) \\ \sigma_1 y(a) + \gamma_1 y'(a) = \delta_1 \\ \sigma_2 y(b) + \gamma_2 y'(b) = \delta_2 \end{cases}$$

Рассмотрим отрезок [a, b] и разобьем его на n частей: $x_i = a + ih$, где $h = \frac{b-a}{n}$ и $0 \le i \le n$,
Обозначив $y_i = y(x_i), p_i = p(x_i), q_i = q(x_i), f_i = f(x_i)$.

Используя конечно-разностными отношениями, представим производные в исходном дифференциальном уравнении следующим способом:

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + p_i \frac{y_{i+1} - y_{i-1}}{2h} + q_i y_i = -f_i, 1 \le i \le n - 1$$

Аппроксимируем производные в дополнительных условиях:

$$\begin{cases} \sigma_1 y_0 + \gamma_1 \dfrac{y_1 - y_0}{h} = \delta_1 \\ \sigma_2 y_n + \gamma_2 \dfrac{y_{n+1} - y_n}{h} = \delta_2 \end{cases}$$

Собираем коэффициенты при $y_i, y_{i+1}, y_{i-1}, y_0, y_1, y_n, y_{n+1}$ и получаем СЛАУ с неизвестными $y_0, y_1, \ \dots, y_n$:

$$\begin{cases} y_0(\sigma_1 - \frac{\gamma_1}{h}) + y_1 \frac{\gamma_1}{h} = \delta_1 \\ y_n(\sigma_2 - \frac{\gamma_2}{h}) + y_{n+1} \frac{\gamma_2}{h} = \delta_2 \\ A_i y_{i-1} - C_i y_i + B_i y_{i+1} = F_i \end{cases} \quad 1 \le i \le n-1 \qquad A_i = \frac{1}{h^2} - \frac{p_i}{2h} \quad B_i = \frac{1}{h^2} + \frac{p_i}{2h} \quad C_i = \frac{2}{h^2} + q_i \quad F_i = f_i$$

Данная система имеет трехдиагональную матрицу коэффициентов. Значит можем использовать для ее решения метод прогонки. Решение будет иметь вид $y_i = \alpha_i y_{i+1} + \beta_i$. Поскольку $y_0 = \alpha_0 y_1 + \beta_0$, то из первого уравнения дополнительных условий получим: $\alpha_0 = \frac{-\gamma_1}{\sigma_1 h - \gamma_1}$ и $\beta_0 = \frac{\delta_1 h}{\sigma_1 h - \gamma_1}$.

Затем из исходного уравнения получим рекуррентные формулы для α_i и β_i:

$$\alpha_{i+1} = \frac{B_i}{C_i - \alpha_i A_i} \qquad \beta_{i+1} = \frac{A_i \beta_i - F_i}{C_i - \alpha_i A_i} \qquad 1 \le i \le n-1$$

Из второго уравнения дополнительных условий получим: $y_n = \frac{\delta_2 h + \gamma_2 \beta_{n-1}}{\sigma_2 h + \gamma_2 (1 - \alpha_{n-1})}$

Используя данные формулы, мы можем найти численное решение в правой точке сетки. И затем по формуле $y_i = \alpha_i y_{i+1} + \beta_i$ можем определить значение в предыдущей точке, и так далее. Данный алгоритм позволит нам найти численное решение краевой задачи.

Описание программ

1. Функции из тестов задания

```
double p(double x) {
    return 2;
}

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

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

double
find_a(double x_0, double h, int i)
{
    return (h * h * q(x_0 + h * i) - h * p(x_0 + h * i) + 1);
}

double
find_b(double x_0, double h, int i)
{
    return (h * p(x_0 + h * i) - 2);
}

double
find_c(double x_0, double h, int i)
{
    return 1;
}

double
find_d(double x_0, double h, int i)
{
    return (-(h * h * f(x_0 + i * h)));
}
```

2. Функция create_vector создает вектор длины size

```c
double*
create_vector(int size)
{
    double *vector = (double*)malloc(size * sizeof(double));
    return vector;
}
```

3. Функция destroy_vector освобождает память, занятую вектором длины size

```c
void
destroy_vector(double *vector)
{
    free(vector);
}
```

4. Функция fin_dif реализует метод прогонки решения краевой задачи для дифференциального уравнения второго порядка

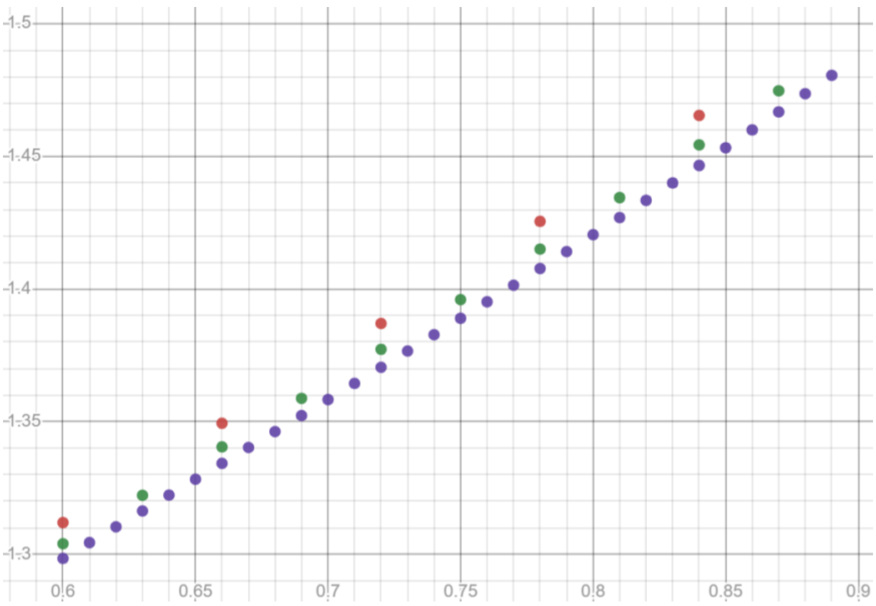
```c
void
fin_dif(double (*p)(double), double (*q)(double), double (*f)(double),
        double sigma_1, double gamma_1, double delta_1,
        double sigma_2, double gamma_2, double delta_2,
        double x_0, double x_n, int n)
{
    double h = (x_n - x_0) / n;

    double *alpha = create_vector(n + 1);
    double *beta = create_vector(n + 1);

    alpha[0] = -gamma_1 / (h * sigma_1 - gamma_1);
    beta[0] = (h * delta_1) /(h * sigma_1 - gamma_1);

    for (int i = 1; i < n; i++) {
        alpha[i] = -find_c(x_0, h, i) / (find_a(x_0, h, i) * alpha[i - 1] + find_b(x_0, h, i));
        beta[i] = (find_d(x_0, h, i) - find_a(x_0, h, i) * beta[i - 1]) / (find_a(x_0, h, i) * alpha[i - 1] + find_b(x_0, h, i));
    }

    double a_n = -gamma_2;
    double b_n = h * sigma_2 + gamma_2;
    double d_n = h *delta_2;
    beta[n] = (d_n - a_n * beta[n - 1]) / (a_n * alpha[n - 1] + b_n);

    double *y = create_vector(n + 1);
    y[n] = beta[n];

    for (int i = n - 1; i >= 0; i--) {
        y[i] = alpha[i] * y[i + 1] + beta[i];
        printf("x = %lf\ty = %lf\n", x_0 + h * i, y[i]);
    }

    destroy_vector(alpha);
    destroy_vector(beta);
    destroy_vector(y);
}
```

Тесты и проверка корректности

Тест №1 (задача №5)

$$\begin{cases} y'' + 2y' - xy = x^2 \\ y'(0.6) = 0.7 \\ y(0.9) - 0.5y'(0.9) = 1 \end{cases}$$

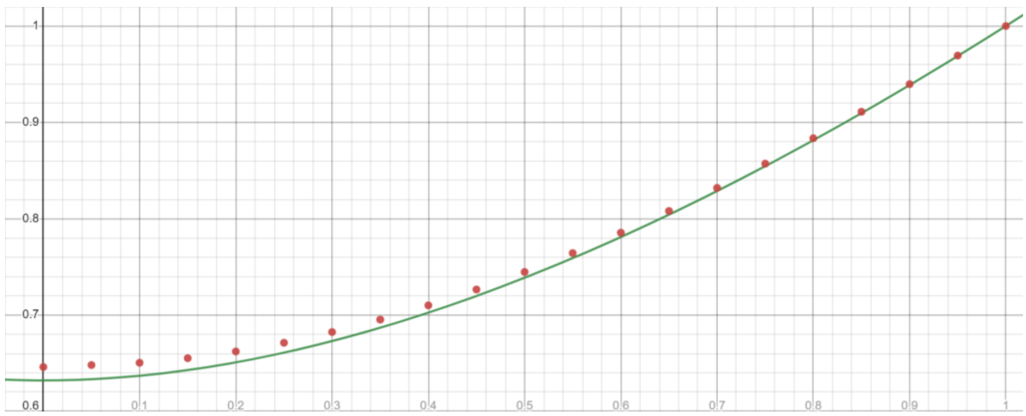
Для данного уравнения не существует решения, представимого в виде элементарных функций. Покажем, как изменяется решение при изменении количества итераций: на графике красным цветом отмечены решения с 5 итерациями, зеленым – с 10 итерациями и фиолетовым – с 30 итерациями. Решения рассматриваются на отрезке [0.6; 0.9].



Тест №2 (дополнительный)

$$\begin{cases} y'' + y' = 1 \\ y'(0) = 0 \\ y(1) = 1 \end{cases} \qquad \text{Точное решение: } y = x + e^{-x} - \frac{1}{e}$$

Для данного уравнения удалось найти точное решение. Покажем на графике зеленым цветом точное решение, красным – решение, полученное программой на отрезке [0; 1].

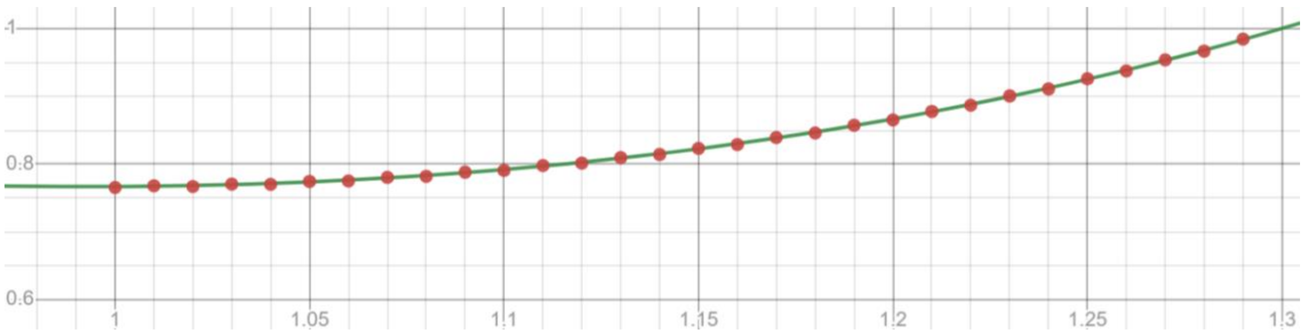


Тест №3 (дополнительный – задача №9)

$$\begin{cases} y'' - \dfrac{y'}{2} \cdot 3y = 2x^2 \\ y(1) - 2y'(1) = 0.6 \\ y(1.3) = 1 \end{cases}$$

$$\text{Аналитическое решение: } y = -0.48 + 2.77e^{-\frac{3x}{2}} + 0.14e^{2x} + \frac{2}{9}x - \frac{2}{3}x^2$$

Для данного уравнения удалось найти точное решение. Покажем на графике зеленым цветом точное решение, красным – решение, полученное программой на отрезке [1; 1.3].



Выводы

При выполнении практической работы были изучен и реализован метод прогонки решения краевой задачи для дифференциального уравнения второго порядка. Было показано, что метод достаточно точно вычисляет решения и что при увеличении числа итераций, точность решения задачи значительно увеличивается. Также были построены графики решений уравнений, которые сопоставлены с графиками точных решений.