МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В.Ломоносова

Факультет вычислительной математики и кибернетики

Компьютерный практикум по учебному курсу

«ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ»

ЗАДАНИЕ № 2.

Численные методы решения дифференциальных уравнений

ОТЧЕТ

о выполненном задании

студента 201 учебной группы факультета ВМК МГУ

Гореленковой Анастасии Петровны

гор. Москва

2022 г.

Содержание

1 Подвариант 1

1.1 Постановка задачи

В данном задании необходимо реализовать несколько методов Рунге-Кутта, позволяющих решать задачи Коши, а именно:

- Метод Рунге-Кутта 2-го порядка точности, применяемый при численном решении задачи Коши для дифференциального уравнения (далее - ДУ) 1-го порядка, разрешённого отностельно производной;
- Метод Рунге-Кутта 4-го порядка точности, применяемый при численном решении задачи Коши для ДУ 1-го порядка, разрешённого отностельно производной;
- Метод Рунге-Кутта 2-го порядка точности, применяемый при численном решении задачи Коши для системы дифференциального уравнения (далее - СДУ) 1-го порядка, разрешённых отностельно производной;
- Метод Рунге-Кутта 4-го порядка точности, применяемый при численном решении задачи Коши для СДУ 1-го порядка, разрешённых отностельно производной.

Существование и единственность решения вышеперечисленных задач Коши гарантируется.

Также требуется:

- Найти численные решения задач Коши и построить для каждого из них график;
- Сравнить полученные результаты с точными решениями.

1.2 Описание алгоритма

Задача Коши для ДУ 1-го порядка, разрешённого относительно производной [1]

Пусть функция $f(x, y)$ определена и непрерывна в прямоугольнике

$$\Pi = \{(x, y) : 0 \leq x - x_0 \leq L, |y - y_0| \leq Y\}.$$

Рассмотрим на отрезке $[x_0; x_0 + L]$ ДУ

$$y'(x) = f(x, y(x)) \tag{1}$$

с начальным условием

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

Задача поиска функции, удовлетворяющей уравнению (1) и начальному условию (2), называется задачей Коши (1)(2).

Функция $\overline{y}(x)$ называется решением задачи Коши (1)(2) на отрезке $[x_0; x_0 + l] \subseteq [x_0; x_0 + L]$, если:

1. $\overline{y}(x) \in C^1[x_0; x_0 + l]$;
2. $|\overline{y}(x) - y_0| \leq Y \ \forall x \in [x_0; x_0 + l]$;
3. $\overline{y}(t)$ удовлетворяет уравнению (1) и начальному условию (2) $\forall x \in [x_0; x_0 + l]$.

Равномерная сетка и разностная схема [2] [3] [4]

Рассмотрим отрезок $[x_0; x_0 + l]$. Возьмём $n \in \mathbb{N}$. Пусть $h = \frac{l}{n}$ и

$$x_i = x_0 + ih, 0 \leq i \leq n. \tag{3}$$

Множество точек $\{x\} = \{x_1; x_2; ...; x_n\}$ называется равномерной сеткой.

Заменив производную на правую разностную производную, сопоставим задаче Коши (1)(2) на отрезке $[x_0; x_0 + l]$ разностную задачу:

$$\frac{y(x_{i+1}) - y(x_i)}{h} = f(x_i, y(x_i)) \iff \frac{y_{i+1} - y_i}{h} = f(x_i, y_i), 0 \leq i \leq n - 1; \tag{4}$$

$$y(x_0) = y_0. \tag{5}$$

Уравнение (4) можно переписать в виде реккурентного соотношения:

$$y_{i+1} = y_i + hf(x_i, y_i), 0 \leq i \leq n - 1; \tag{6}$$

Отсюда видно, что таким способом можно последовательно рассчитать все значения $y_i, 0 \leq i \leq n - 1$ и тем самым решить разностную задачу (4)(5). Поэтому такую разностную схему называют явной.

Разложение по формуле Тейлора и аппроксимация производными [2]

Пусть решение ДУ $y(x)$ имеет производные достаточно высокого порядка. Запишем его разложение по формуле Тейлора:

$$\begin{aligned} y_{i+1} = y(x_{i+1}) = \frac{1}{0!}y(x_i)h^0 + \frac{1}{1!}y'(x_i)h + \frac{1}{2!}y''(x_i)h^2 + \frac{1}{3!}y'''(x_i)h^3 + ... = \\ = y_i + y'(x_i)h + \frac{1}{2}y''(x_i)h^2 + \frac{1}{6}y'''(x_i)h^3 + ... \end{aligned} \tag{7}$$

Нетрудно заметить, что показанная выше разностная схема (4)(5) получится, если оборвать разложение Тейлора на члене порядка h и положить $y'(x_i) = f(x_i, y_i)$. Чтобы повысить точность аппроксимации задачи Коши (1)(2) нужно получить разностные схемы более высоких порядков точности. Этого можно добиться либо если обрывать разложение на членах порядка h^2, h^3 и т.д., либо если заменять вычисление производных функции нахождением самой функции в нескольких точках (аналогично теореме Лагража о конечных приращениях). Первый способ имеет существенный недостаток: при возрастании порядка значительно увеличивается количество производных, которые надо вычислить $(\frac{\partial f(x)}{\partial x}, \frac{\partial f(x)}{\partial y}; \frac{\partial^2 f(x)}{\partial x^2}, \frac{\partial^2 f(x)}{\partial x \partial y}, \frac{\partial^2 f(x)}{\partial y^2}; \frac{\partial^3 f(x)}{\partial x^3}, \frac{\partial^3 f(x)}{\partial x^2 \partial y}, \frac{\partial^3 f(x)}{\partial x \partial y^2}, \frac{\partial^3 f(x)}{\partial y^3}; ...)$. Метод Рунге-Кунга, напротив, основывается на 2 подходе.

Метод Рунге-Кутта 2-го порядка точности [2] [3] [4]

Рассмотрим реккурентную формулу разностной схемы 2-го порядка точности:

$$\frac{y_{i+1} - y_i}{h} = f(x_i, y_i) + \frac{1}{2}\Big(\frac{\partial f}{\partial x}(x_i, y_i) + \frac{\partial f}{\partial y}(x_i, y_i) f(x_i, y_i)\Big)h. \tag{8}$$

Приближённо заменим правую часть формулы на сумму значений функции $f(x, y)$ в 2-ух разных точках с точностью до членов порядка h^2. Для этого положим

$$\begin{aligned} f(x_i, y_i) + \frac{1}{2}\Big(\frac{\partial f}{\partial x}(x_i, y_i) + \frac{\partial f}{\partial y}(x_i, y_i) f(x_i, y_i)\Big)h = \\ = \beta f(x_i, y_i) + \alpha f(x_i + \gamma h, y_i + \delta h) + O(h^2), \end{aligned} \tag{9}$$

где $\alpha, \beta, \gamma, \delta$ - параметры, которые нужно подобрать.

Разложив функцию $f(x_i + \gamma h, y_i + \delta h)$ по степеням h:

$$f(x_i + \gamma h, y_i + \delta h) = f(x_i, y_i) + \Big(\gamma \frac{\partial f}{\partial x}(x_i, y_i) + \delta \frac{\partial f}{\partial y}(x_i, y_i)\Big)h + O(h^2) \tag{10}$$

и подставив (10) в (9), получим:

$$\begin{aligned} &\beta = 1 - \alpha, \\ &\gamma = \frac{1}{2\alpha}, \\ &\delta = \frac{1}{2\alpha} f(x_i, y_i). \end{aligned} \tag{11}$$

Таким образом получили семейство разностных схем Рунге-Кутта 2-го порядка точности, которое зависит от параметра α. Реккурентное соотношение для него:

$$y_{i+1} = y_i + \Bigg((1-\alpha)f(x_i, y_i) + \alpha f\Big(x_i + \frac{h}{2\alpha}, y_i + \frac{h}{2\alpha}f(x_i, y_i)\Big)\Bigg)h. \tag{12}$$

Возьмём $\alpha = \frac{1}{2}$:

$$y_{i+1} = y_i + \frac{h}{2}\Bigg(f(x_i, y_i) + f\Big(x_i + h, y_i + hf(x_i, y_i)\Big)\Bigg). \tag{13}$$

Метод Рунге-Кутта 4-го порядка точности [2] [3] [4]

Аналогично реккурентному соотношению для метода 2-го порядка выводится реккурентное соотношения для метода 4-го порядка:

$$\begin{aligned} y_{i+1} &= y_i + \frac{h}{6}\Big(k_{i_1} + 2k_{i_2} + 2k_{i_3} + k_{i_4}\Big),\\ \text{где } k_{i_1} &= f(x_i, y_i),\\ k_{i_2} &= f(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_{i_1}),\\ k_{i_3} &= f(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_{i_2}),\\ k_{i_4} &= f(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_{i_3}). \end{aligned} \tag{14}$$

Задача Коши для СДУ 1-го порядка, разрешённых относительно производных [1]

Пусть функции $f_1(x, y_1, y_2)$ и $f_2(x, y_1, y_2)$ определены и непрерывны

$$\forall x \in [x_0; x_0 + L], (y_1, y_2) \in \mathbb{R}^2.$$

Рассмотрим на отрезке $[x_0; x_0 + L]$ СДУ

$$\begin{cases} \frac{dy^{(1)}(x)}{dx} = f_1(x, y^{(1)}(x), y^{(2)}(x)) \\ \frac{dy^{(2)}(x)}{dx} = f_2(x, y^{(1)}(x), y^{(2)}(x)) \end{cases} \tag{15}$$

с начальными условиями

$$\begin{cases} y^{(1)}(x_0) = y_0^{(1)} \\ y^{(2)}(x_0) = y_0^{(2)} \end{cases} \tag{16}$$

Задача поиска функций, удовлетворяющих системе (15) и начальным условиям (16), называется задачей Коши (15)(16).

Функции $\overline{y}^{(1)}(x)$ и $\overline{y}^{(2)}(x)$ называются решением задачи Коши (15)(16) на отрезке $[x_0; x_0 + L]$, если:

1. $\overline{y}^{(1)}(x), \overline{y}^{(2)}(x) \in C^1[x_0; x_0 + L]$;

2. $\overline{y}^{(1)}(x)$ и $\overline{y}^{(2)}(x)$ удовлетворяют системе (15) и начальным условиям (16) $\forall x \in [x_0; x_0 + L]$.

Методы Рунге-Кутта 2-го и 4-го порядка точности для СДУ [2] [3] [4]

Чтобы обобщить метод Рунге-Кутта на СДУ нужно заменить $y, k_{i_1}, k_{i_2}, k_{i_3}$ и k_{i_4} векторами, а функцию f - функциями f_1 и f_2.

Реккурентное соотношения для метода Рунге-Кутта 2-го порядка точности:

$$\begin{cases} y_{i+1}^{(1)} = y_i^{(1)} + \frac{h}{2}\Bigg(f_1(x_i, y_i^{(1)}) + f_1\Big(x_i + h, y_i^{(1)} + hf_1(x_i, y_i^{(1)})\Big)\Bigg) \\ y_{i+1}^{(2)} = y_i^{(2)} + \frac{h}{2}\Bigg(f_2(x_i, y_i^{(2)}) + f_2\Big(x_i + h, y_i^{(2)} + hf_2(x_i, y_i^{(2)})\Big)\Bigg) \end{cases} \tag{17}$$

Реккурентное соотношения для метода Рунге-Кутта 4-го порядка точности:

$$\begin{cases} y_{i+1}^{(1)} = y_i^{(1)} + \dfrac{h}{6}\Big(k_{i_1}^{(1)} + 2k_{i_2}^{(1)} + 2k_{i_3}^{(1)} + k_{i_4}^{(1)}\Big), \\ \quad \text{где } k_{i_1}^{(1)} = f(x_i, y_i^{(1)}), \\ \qquad k_{i_2}^{(1)} = f_1(x_i + \dfrac{h}{2}, y_i^{(1)} + \dfrac{h}{2}k_{i_1}^{(1)}), \\ \qquad k_{i_3}^{(1)} = f_1(x_i + \dfrac{h}{2}, y_i^{(1)} + \dfrac{h}{2}k_{i_2}^{(1)}), \\ \qquad k_{i_4}^{(1)} = f_1(x_i + \dfrac{h}{2}, y_i^{(1)} + \dfrac{h}{2}k_{i_3}^{(1)}). \\ \\ y_{i+1}^{(2)} = y_i^{(2)} + \dfrac{h}{6}\Big(k_{i_1}^{(2)} + 2k_{i_2}^{(2)} + 2k_{i_3}^{(2)} + k_{i_4}^{(2)}\Big), \\ \quad \text{где } k_{i_1}^{(2)} = f_2(x_i, y_i^{(2)}), \\ \qquad k_{i_2}^{(2)} = f_2(x_i + \dfrac{h}{2}, y_i^{(2)} + \dfrac{h}{2}k_{i_1}^{(2)}), \\ \qquad k_{i_3}^{(2)} = f_2(x_i + \dfrac{h}{2}, y_i^{(2)} + \dfrac{h}{2}k_{i_2}^{(2)}), \\ \qquad k_{i_4}^{(2)} = f_2(x_i + \dfrac{h}{2}, y_i^{(2)} + \dfrac{h}{2}k_{i_3}^{(2)}). \end{cases} \tag{18}$$

1.3 Тестирование программы

Текст программы находится в Приложении А.

1. **Пример решения задачи Коши для ДУ (см. таблицу 1-6 задания)**

   Задача Коши:

$$y' = (x - x^2)y, \qquad x \in [0; 1]; \tag{19}$$

$$y(0) = 1. \tag{20}$$

   Точное решение:

$$exp(-\frac{1}{6}x^2(-3 + 2x)). \tag{21}$$

   Рассмотрим работу методов Рунге-Кутта 2 и 4 порядка точности для ДУ при разном количестве шагов n.

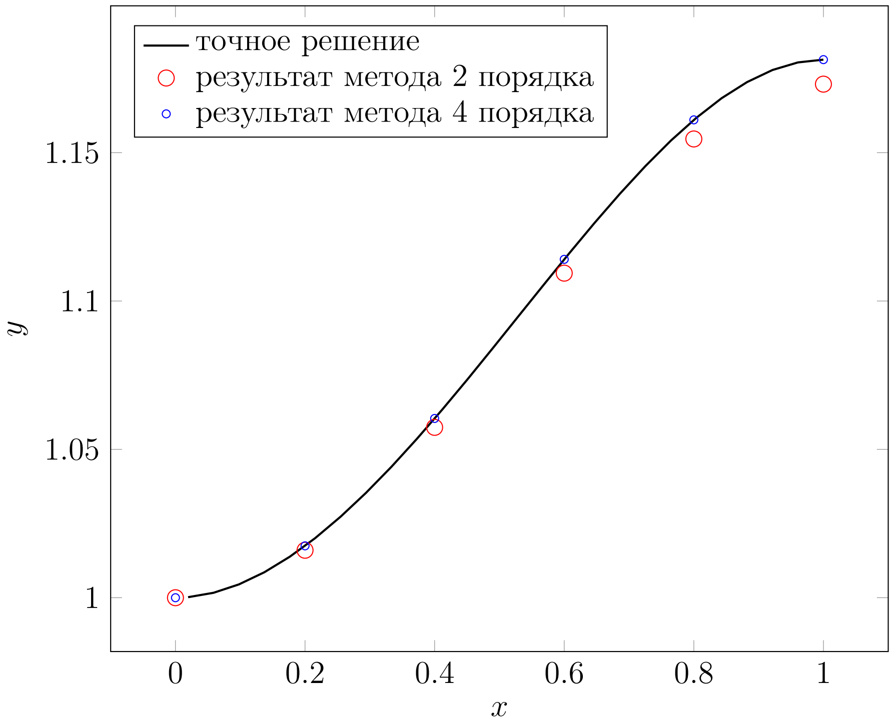

Рис. 1: $n = 5$

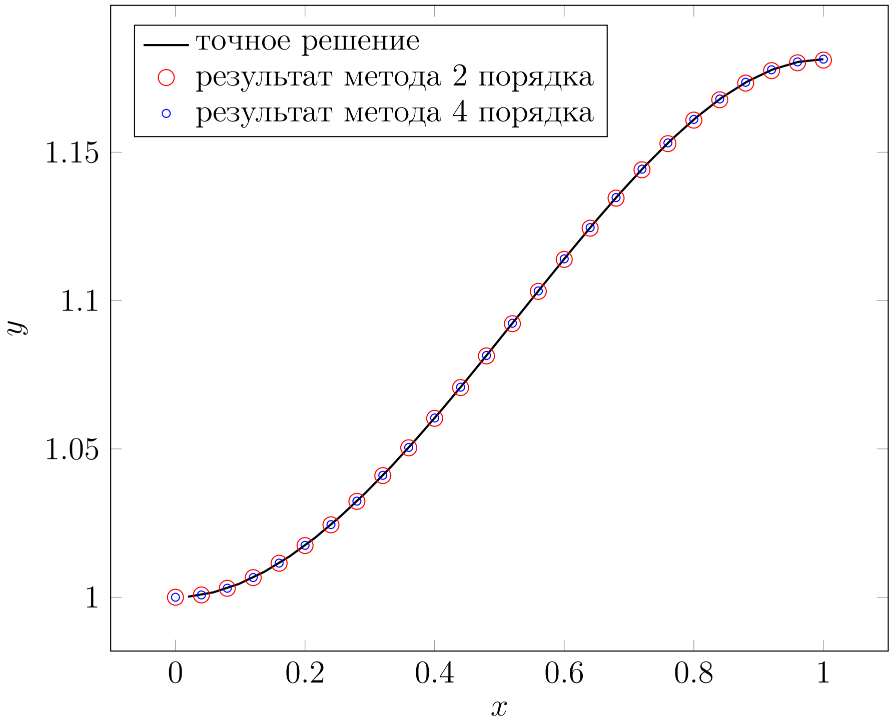

Рис. 2: $n = 25$

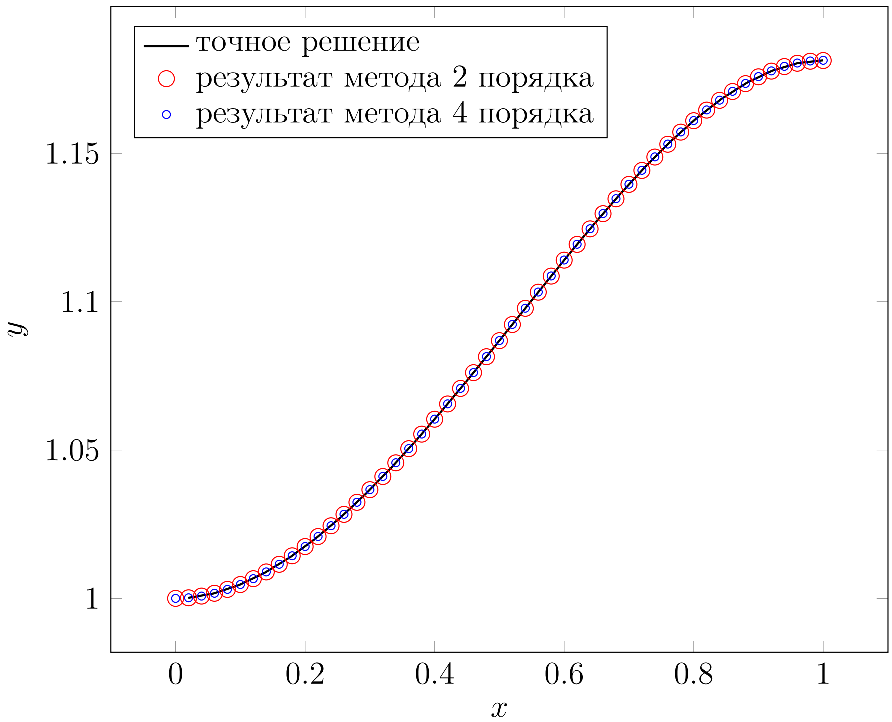

Рис. 3: $n = 50$

2. **Пример решения задачи Коши для СДУ (см. таблицу 2-5 задания)**

   Задача Коши:

$$\begin{cases} u' = cos(u + 1.1v) + 2.1, \\ v' = \frac{1.1}{x+2.1u^2} + x + 1, \qquad x \in [0; 10]; \end{cases} \tag{22}$$

$$\begin{cases} u(0) = 1, \\ v(0) = 0.05. \end{cases} \tag{23}$$

Рассмотрим работу методов Рунге-Кутта 2 и 4 порядка точности для ДУ при разном количестве шагов n.

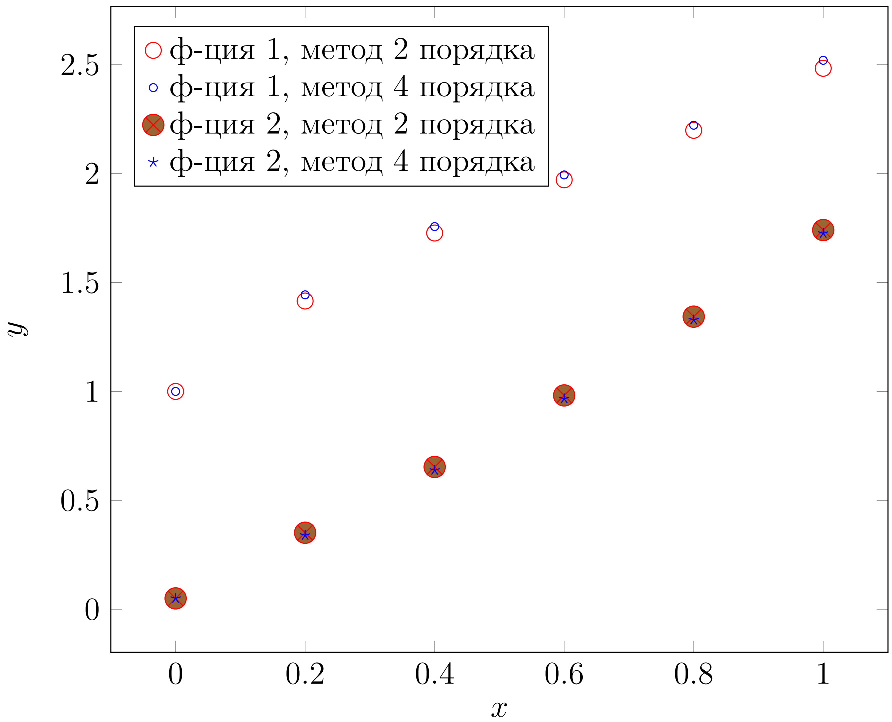

Рис. 4: $n = 5$

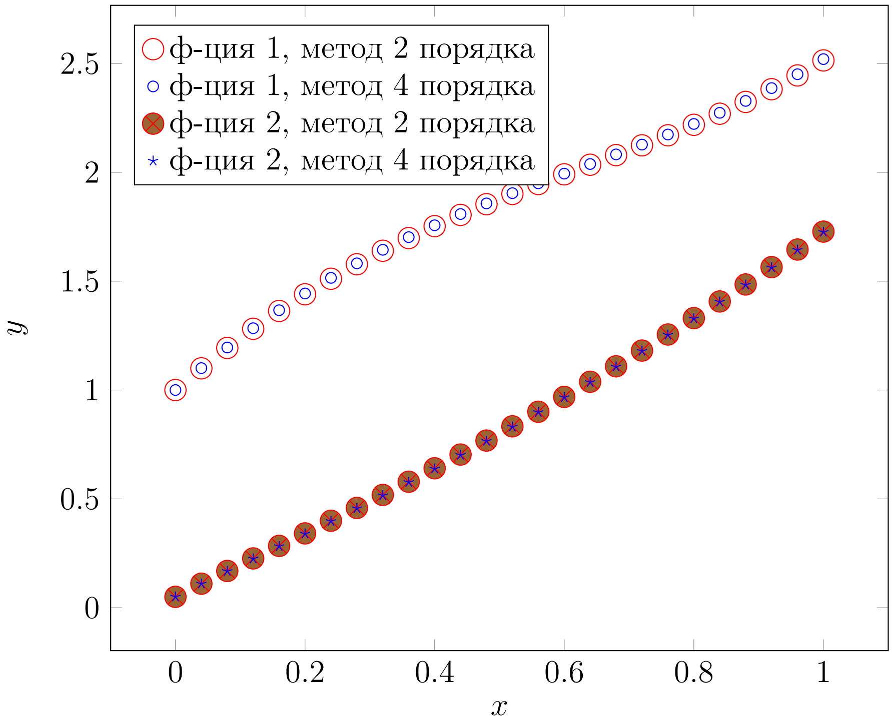

Рис. 5: $n = 25$

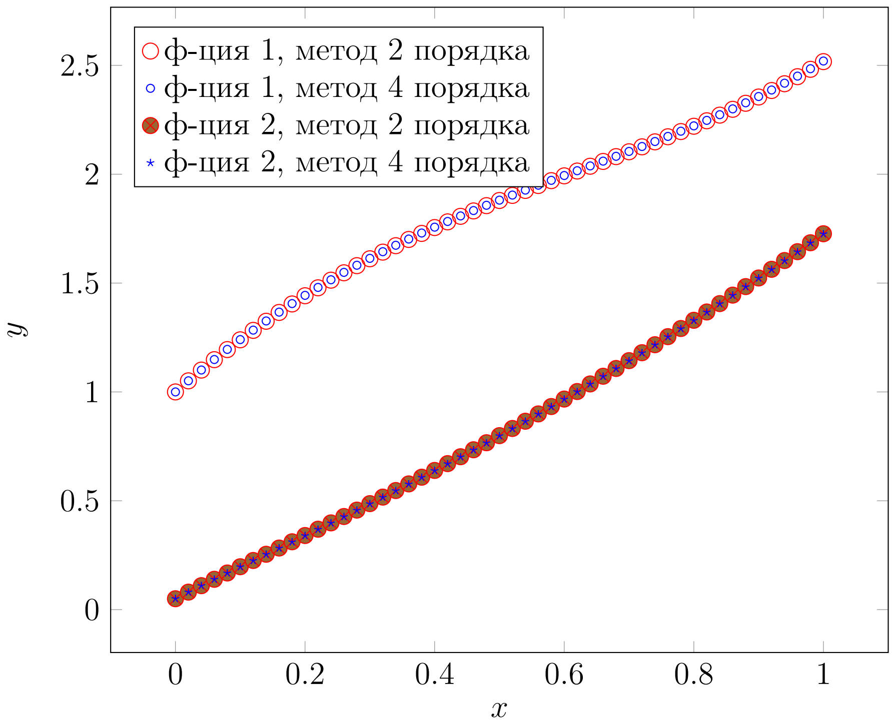

Рис. 6: $n = 50$

1.4 Вывод

Тесты наглядно показывают, что точность метода Рунге-Кнутта возрастает при увеличении:

- числа шагов алгоритма;
- порядка точности алгоритма.

2 Подвариант 2

2.1 Постановка задачи

В данном задании необходимо реализовать алгоритм, позволяющий находить решение краевой задачи для ОДУ 2-го порядка, разрешённого относительно производной. Он должен включать в себя реализацию двух методов:

- Метод конечных разностей;
- Метод прогонки.

Также требуется:

- Найти численное решение краевой задачи и построить график этого решения;
- Сравнить полученные результаты с точными решениями.

2.2 Описание алгоритма

Краевая задача [5]

Рассмотрим на отрезке $[x_0; x_n]$ ДУ

$$y''(x) + p(x)y'(x) + q(x)y(x) = f(x), \qquad x_0 \le x \le x_n \tag{24}$$

с краевыми условиями

$$\begin{cases} r_1 y(x_0) + s_1 y'(x_0) = t_1 \\ r_2 y(x_n) + s_2 y'(x_n) = t_2 \end{cases} \tag{25}$$

Задача поиска функции, удовлетворяющей уравнению (24) и краевым условиям (25), называется краевой задачей (24)(25).

Равномерная сетка и сеточная функция [2] [3] [4]

Рассмотрим отрезок $[x_0; x_n]$. Пусть $h = \frac{x_n - x_0}{n}$ и

$$x_i = x_0 + ih, 0 \le i \le n. \tag{26}$$

Введём следующие обозначения: $p(x_i) = p_i, q(x_i) = q_i, f(x_i) = f_i, y(x_i) = y_i$.

Множество точек $\{x\} = \{x_1; x_2; ...; x_n\}$ называется равномерной сеткой, а множество $\{y\} = \{y_1; y_2; ...; y_n\}$ - сеточной функцией.

Метод конечных разностей [2]

Очевидно, что для сеточной функции невозможно ввести понятие производной в привычном понимании. Вместо этого введём понятия правой и левой разностных апроксимаций 1-ой производной:

$$L_h^+[y_i] = \frac{y_{i+1} - y_i}{h}, \qquad 0 \le i \le n-1; \tag{27}$$

$$L_h^-[y_i] = \frac{y_i - y_{i-1}}{h}, \qquad 1 \le i \le n. \tag{28}$$

С их помощью выводится выражение для разностной аппроксимации 2-ой производной:

$$L_h[y_i] = \begin{cases} \frac{1}{h}(y_1 - y_0), \qquad i = 0 \\ \frac{1}{h}(\frac{y_{i+1} - y_i}{h} - \frac{y_i - y_{i-1}}{h}) = \frac{1}{h^2}(y_{i+1} - 2y_i + y_{i-1}), \qquad 1 \le i \le 21 \\ \frac{1}{h}(y_n - y_{n-1}), \qquad i = n \end{cases} \tag{29}$$

Подставив полученные выражения в краевую задачу (24)(25) вместо производных, получим:

$$\begin{cases} \frac{1}{h^2}(y_{i+1} - 2y_i + yi - 1) + \frac{p_i}{2h}(y_{i+1} - y_{i-1}) + q_i y_i = f_i \\ r_1 y_0 + \frac{s_1}{h}(y_1 - y_0) = t_1 \\ r_2 y_n + \frac{s_2}{h}(y_n - y_{n-1}) = t_2 \end{cases} \tag{30}$$

Приведём подобные слагаемые:

$$\begin{cases} (\frac{1}{h^2} - \frac{p_i}{2h})y_{i-1} - (\frac{2}{h^2} - q_i)y_i + (\frac{1}{h^2} + \frac{p_i}{2h})y_{i+1} = f_i \\ (r_1 - \frac{s_1}{h})y_0 + \frac{s_1}{h}y_1 = t_1 \\ (r_2 - \frac{s_2}{h})y_{n-1} + \frac{s_2}{h}y_n = t_2 \end{cases} \tag{31}$$

Введём следующие обозначения:

$$b_0 = \frac{s_1}{r_1 h - s_1}, \qquad f_0 = \frac{t_1 h}{r_1 h - s_1};$$

$$a_i = \frac{1}{h^2} - \frac{p_i}{2h}, \qquad c_i = \frac{2}{h^2} - q_i, \qquad b_i = \frac{1}{h^2} + \frac{p_i}{2h}, \qquad 1 \le i \le n-1;$$

$$a_n = -\frac{s_2}{r_2 h + s_2}, \qquad f_n = \frac{t_2 h}{r_2 h + s_2}.$$

Тогда краевая задача примет вид:

$$\begin{cases} y_0 + b_0 y_1 = f_0 \\ a_i y_{i-1} - c_i y_i + b_i y_{i+1} = f_i, \qquad 1 \le i \le n-1 \\ a_n y_{n-1} + y_n = f_n \end{cases} \tag{32}$$

Таким образом, краевая задача сведена к трёхдиагональной матрице.

Метод прогонки [2] [3]
Формулы для прямого хода метода прогонки:

$$\alpha_1 = -b_0, \qquad \beta_1 = f_0;$$

$$\alpha_{i+1} = -\frac{b_i}{a_i \alpha_i - c_i}, \qquad \beta_{i+1} = \frac{f_i - a_i \beta_i}{a_i \alpha_i - c_i}, \qquad 1 \le i \le n-1;$$

Формулы для обратного хода метода прогонки:

$$y_n = -\frac{\alpha_n \beta_n + f_n}{1 - a_n \alpha_n}, \qquad y_i = \alpha_i y_{i+1} + \beta_{i+1}, \qquad n-1 \ge i \ge 1.$$

2.3 Тестирование программы

Текст программы находится в Приложении B.

Примеры решения краевой задачи (см. [2])
Краевая задача:

$$y'' - y = -1, \qquad x \in [-1; 1]; \tag{33}$$
$$y(-1) = 0; \tag{34}$$
$$y(1) = 0. \tag{35}$$

Точное аналитическое решение:

$$1 - \frac{ch(x)}{ch(1)}. \tag{36}$$

Рассмотрим работу алгоритма при разном количестве шагов n.

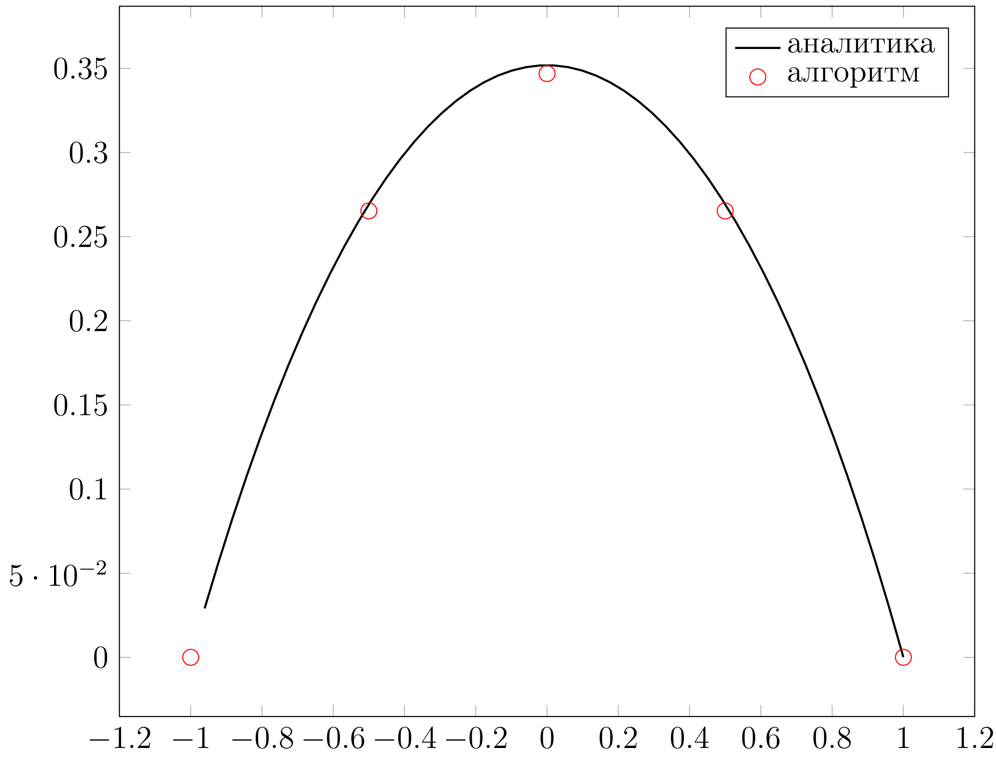

Рис. 7: $n = 4$

Ответ:
(-1.000000, 0.000000), (-0.500000, 0.265306), (0.000000, 0.346939), (0.500000, 0.265306), (1.000000, 0.000000)

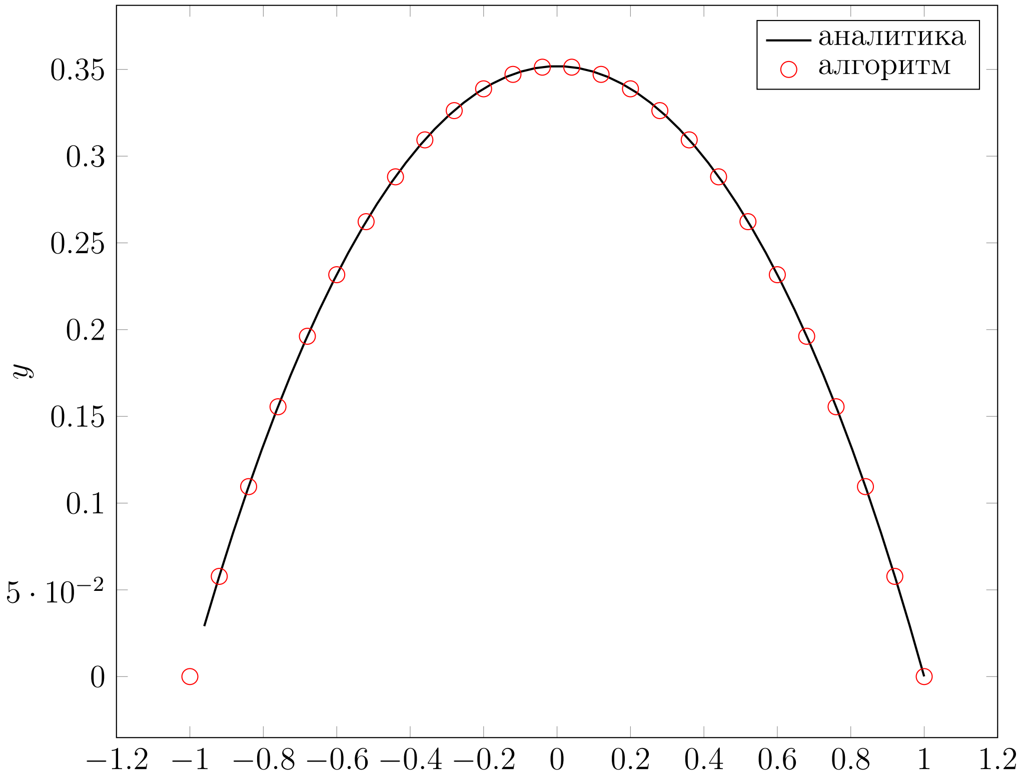

Рис. 8: $n = 25$

Ответ:
(-1.000000, 0.000000), (-0.920000, 0.057767), (-0.840000, 0.109504), (-0.760000, 0.155542), (-0.680000, 0.196175), (-0.600000, 0.231664), (-0.520000, 0.262236), (-0.440000, 0.288086), (-0.360000, 0.309379), (-0.280000, 0.326253), (-0.200000, 0.338814), (-0.120000, 0.347144), (-0.040000, 0.351296), (0.040000, 0.351296), (0.120000, 0.347144), (0.200000, 0.338814), (0.280000, 0.326253), (0.360000, 0.309379), (0.440000, 0.288086), (0.520000, 0.262236), (0.600000, 0.231664), (0.680000, 0.196175), (0.760000, 0.155542), (0.840000, 0.109504), (0.920000, 0.057767), (1.000000, 0.000000)

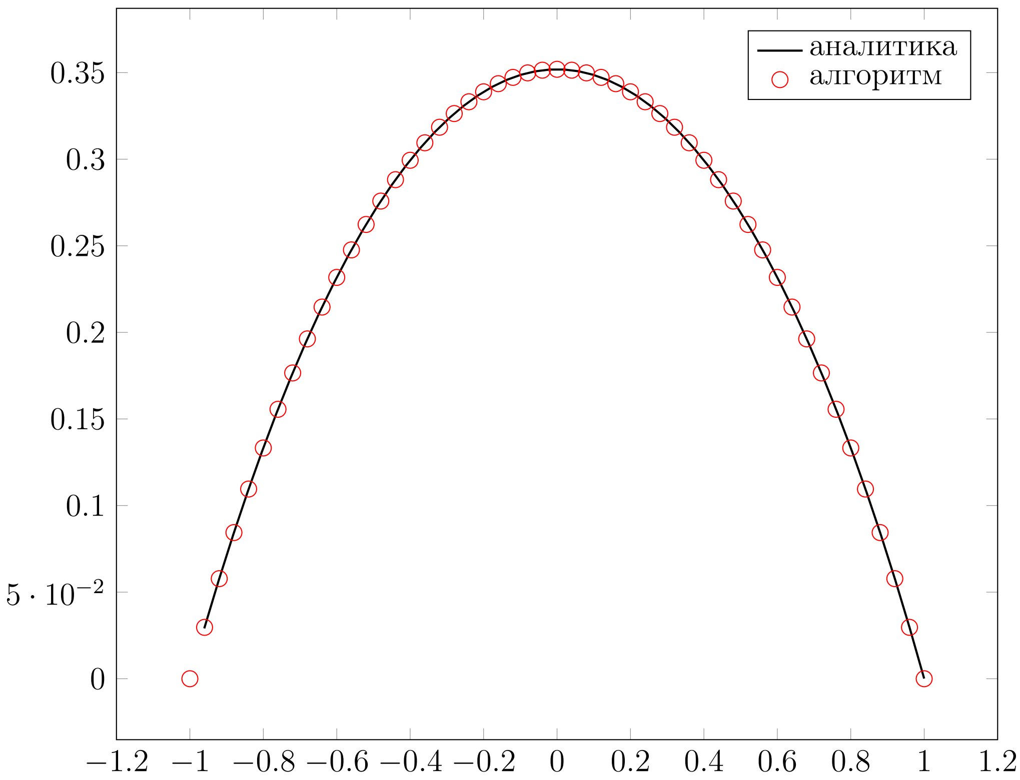

Рис. 9: $n = 50$

Ответ:
(-1.000000, 0.000000), (-0.960000, 0.029669), (-0.920000, 0.057785), (-0.880000, 0.084394), (-0.840000, 0.109537), (-0.800000, 0.133256), (-0.760000, 0.155588), (-0.720000, 0.176570), (-0.680000, 0.196233), (-0.640000, 0.214611), (-0.600000, 0.231732), (-0.560000, 0.247623), (-0.520000, 0.262311), (-0.480000, 0.275819), (-0.440000, 0.288168), (-0.400000, 0.299378), (-0.360000, 0.309467), (-0.320000, 0.318451), (-0.280000, 0.326345), (-0.240000, 0.333161), (-0.200000, 0.338910), (-0.160000, 0.343601), (-0.120000, 0.347242), (-0.080000, 0.349838), (-0.040000, 0.351394), (0.000000, 0.351913), (0.040000, 0.351394), (0.080000, 0.349838), (0.120000, 0.347242), (0.160000, 0.343601), (0.200000, 0.338910), (0.240000, 0.333161), (0.280000, 0.326345), (0.320000, 0.318451), (0.360000, 0.309467), (0.400000, 0.299378), (0.440000, 0.288168), (0.480000, 0.275819), (0.520000, 0.262311), (0.560000, 0.247623), (0.600000, 0.231732), (0.640000, 0.214611), (0.680000, 0.196233), (0.720000, 0.176570), (0.760000, 0.155588), (0.800000, 0.133256), (0.840000, 0.109537), (0.880000, 0.084394), (0.920000, 0.057785), (0.960000, 0.029669), (1.000000, 0.000000)

Краевая задача:

$$y'' + 2y' - \frac{y}{x} = 3, \qquad x \in [0.2; 0.5]; \tag{37}$$
$$y(0.2) = 1; \tag{38}$$
$$0.5y(0.5) - y'(0.5) = 1. \tag{39}$$

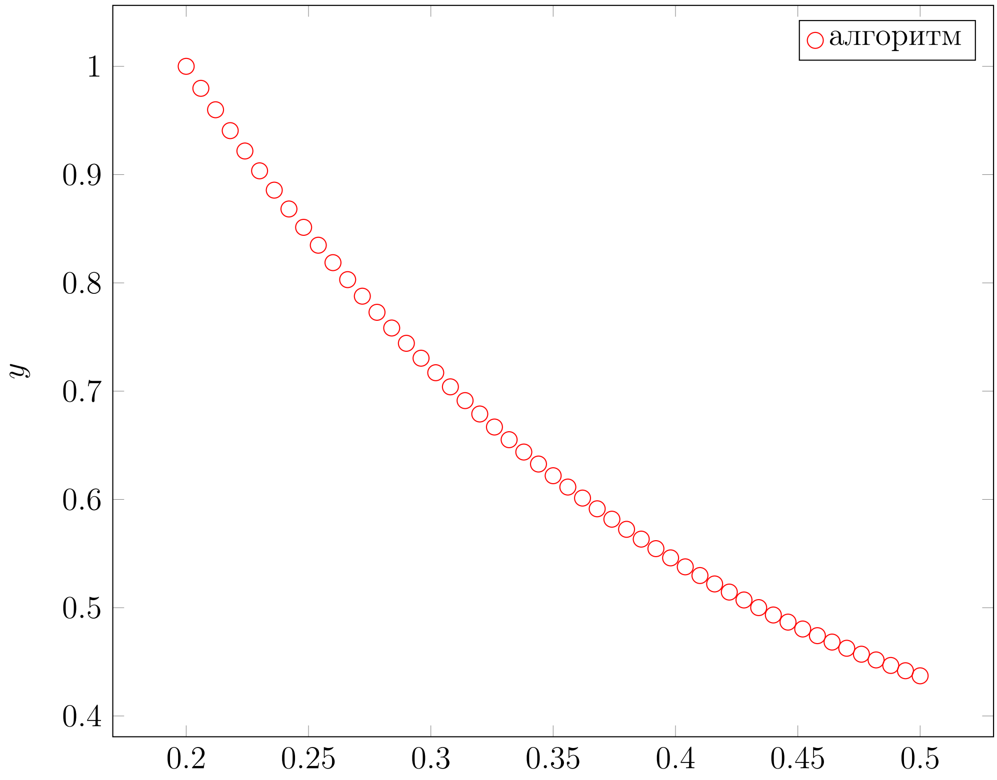

Рис. 10: $n = 50$

Ответ:
(0.200000, 1.000000), (0.206000, 0.979688), (0.212000, 0.959896), (0.218000, 0.940609), (0.224000, 0.921815), (0.230000, 0.903499), (0.236000, 0.885649), (0.242000, 0.868254), (0.248000, 0.851302), (0.254000, 0.834783), (0.260000, 0.818686), (0.266000, 0.803000), (0.272000, 0.787718), (0.278000, 0.772828), (0.284000, 0.758323), (0.290000, 0.744194), (0.296000, 0.730432), (0.302000, 0.717031), (0.308000, 0.703981), (0.314000, 0.691277), (0.320000, 0.678910), (0.326000, 0.666874), (0.332000, 0.655162), (0.338000, 0.643767), (0.344000, 0.632685), (0.350000, 0.621907), (0.356000, 0.611429), (0.362000, 0.601245), (0.368000, 0.591349), (0.374000, 0.581736), (0.380000, 0.572401), (0.386000, 0.563338), (0.392000, 0.554543), (0.398000, 0.546011), (0.404000, 0.537737), (0.410000, 0.529716), (0.416000, 0.521945), (0.422000, 0.514419), (0.428000, 0.507134), (0.434000, 0.500085), (0.440000, 0.493270), (0.446000, 0.486682), (0.452000, 0.480320), (0.458000, 0.474179), (0.464000, 0.468256), (0.470000, 0.462547), (0.476000, 0.457048), (0.482000, 0.451757), (0.488000, 0.446670), (0.494000, 0.441784), (0.500000, 0.437095)

2.4 Вывод

Тесты наглядно показывают, что точность алгоритма возрастает при увеличении числа шагов.

A Приложение 1: метод Рунге-Кутта

Текст программы runge_kutta.py на языке Python (специальные вспомогательные функции ввода и вывода опущены):

```
from math import *

""" Заданные функции """
def f(x, y):
    return (x - x ** 2) * y

def f1(x, u, v):
    return cos(u + 1.1 * v) + 2.1

def f2(x, u, v):
    return 1.1 / (x + 2.1 * u**2) + x + 1

""" Алгоритм Рунге-Кутта 2-ого порядка точности (для уравнения) """
def runge_kutt_eq_2(func, h, n, x, y):
    h = h / n
    for i in range(n):
        print(x, y)
        y += (func(x, y) + func(x + h, y + h * func(x, y))) * h / 2
        x += h
    print(x, y)


""" Алгоритм Рунге-Кутта 4-ого порядка точности (для уравнения) """
def runge_kutt_eq_4(func, h, n, x, y):
    h = h / n
    for i in range(n):
        print(x, y)
        k1 = func(x, y)
        k2 = func(x + h / 2, y + h / 2 * k1)
        k3 = func(x + h / 2, y + h / 2 * k2)
        k4 = func(x + h, y + h * k3)
        y += h / 6 * (k1 + 2 * k2 + 2 * k3 + k4)
        x += h
    print(x, y)

""" Алгоритм Рунге-Кутта 2-ого порядка точности (для системы) """
def runge_kutt_sys_2(funcs, h, n, x, y):
    h = h / n
    syslen = 2
    for i in range(n):
        #print(x, y[0])
```

```
        print(x, y[1])
        tmp1 = [0] * syslen
        tmp2 = [0] * syslen
        for j in range(syslen):
            tmp1[j] = funcs[j](x, y[0], y[1])
            tmp2[j] = funcs[j](x + h, y[0] + h * tmp1[j],
y[1] + h * tmp1[j])
        for j in range(syslen):
            y[j] += (tmp1[j] + tmp2[j]) * h / 2
        x += h
    #print(x, y[0])
    print(x, y[1])

""" Алгоритм Рунге-Кутта 4-ого порядка точности (для системы) """
def runge_kutt_sys_4(funcs, h, n, x, y):
    h = h / n
    syslen = 2
    for i in range(n):
        #print(x, y[0])
        print(x, y[1])
        k1 = [0] * syslen
        k2 = [0] * syslen
        k3 = [0] * syslen
        k4 = [0] * syslen
        for j in range(syslen):
            k1[j] = funcs[j](x, y[0], y[1])
        for j in range(syslen):
            k2[j] = funcs[j](x + h / 2, y[0] + h / 2 * k1[0],
y[1] + h / 2 * k1[1])
        for j in range(syslen):
            k3[j] = funcs[j](x + h / 2, y[0] + h / 2 * k2[0],
y[1] + h / 2 * k2[1])
        for j in range(syslen):
            k4[j] = funcs[j](x + h, y[0] + h * k3[0],
y[1] + h * k3[1])
        for j in range(syslen):
            y[j] += h / 6 * (k1[j] + 2 * k2[j] + 2 * k3[j] + k4[j])
        x += h
    #print(x, y[0])
    print(x, y[1])
<...>
```

B Приложение 2: метод корневых разностей + метод прогонки

Текст программы boundry_value_problem.py на языке Python (специальные вспомогательные функции ввода и вывода опущены):

```
from math import *

""" Заданные функции """
def testp(x):
    return 0

def testq(x):
    return -1

def testf(x):
    return -1

def p(x):
    return 2

def q(x):
    return -1/x

def f(x):
    return 3

""" Реализация алгоритма """
def alg(p, q, f, x0, xn, r1, s1, t1, r2, s2, t2, n):
    res = []
    h = (xn - x0) / n
    alpha = [0 for _ in range(n + 1)]
    alpha[1] = - (s1 / (h * r1 - s1))
    beta = [0 for _ in range(n + 1)]
    beta[1] = t1 / (r1 - s1 / h)
    x = [0 for _ in range(n + 1)]
    x[0] = x0
    x[n] = xn
    y = [0 for _ in range(n + 1)]
    for i in range(1, n):
        x[i] = x[i - 1] + h
        k1 = 1 / h ** 2 - p(x[i]) / (2 * h)
        k2 = 1 / h ** 2 + p(x[i]) / (2 * h)
        k3 = -2 / h ** 2 + q(x[i])
        alpha[i + 1] = -k2 / (k1 * alpha[i] + k3)
        beta[i + 1] = (f(x[i]) - k1 * beta[i]) / (k1 * alpha[i] + k3)
```

```
    tmp1 = s2 * beta[n] + t2 * h
    tmp2 = s2 * (1 - alpha[n]) + r2 * h
    y[n] = tmp1 / tmp2
    for i in range(n, 0, -1):
        y[i - 1] = y[i] * alpha[i] + beta[i]
    for i in range(n + 1):
        res.append((x[i], y[i]))
    return res
<...>
```

Список литературы

[1] Денисов А. М., Разгулин А. В. Обыкновенные дифференциальные уравнения. Часть 1. — Москва, 2009.

[2] Костомаров Д. П., Фаворский А. П. Вводные лекции по численным методам. — Москва: Логос, 2004.

[3] Бахвалов Н. С., Жидков Н. П., Кобельков Г. М. Численные методы - 8-е изд. (эл.). — Москва: БИНОМ. Лаборатория знаний, 2015.

[4] Самарский А. А. Введение в численные методы. Учебное пособие для вузов - 3-е изд. — Санкт-Петербург: Лань, 2005.

[5] Денисов А. М., Разгулин А. В. Обыкновенные дифференциальные уравнения. Часть 2. — Москва, 2009.