Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
Московский государственный технический университет им. Н.Э. Баумана
Факультет «Фундаментальные науки»

Лабораторная работа №10
по курсу «Вычислительная физика»
Тема: «Приближённые методы решения краевых задач для ОДУ»
Вариант 6

Выполнили: студенты группы ФН4-72Б
Хижик А.И., Мистрюкова Л.А.
Проверил: доцент, к.физ.-мат.н.
Хасаншин Р.Х.

Москва, 2019

Оглавление

1. Теоретическая часть

1.1. Метод прогонки

Для систем алгебраических уравнений метод прогонки – аналог метода Гаусса. Используется для решения систем с разреженными матрицами, а именно, трёхдиагональными матрицами.

Пусть требуется найти решение следующей системы трёхточечных уравнений:

$$\begin{cases} c_0 y_0 - b_0 y_1 = f_0, & i = 0, \\ -a_i y_{i-1} + c_i y_i - b_i y_{i+1} = f_i, & 1 \le i \le n-1, \\ -a_n y_{n-1} + c_n y_n = f_n, & i = n, \end{cases} \tag{1}$$

или в векторном виде

$$\mathcal{A}Y = F,$$

где $Y = (y_0, \ldots, y_n)$ – вектор неизвестных, $F = (f_0, \ldots, f_n)$ – вектор правых частей, $\mathcal{A}$ – квадратная $(n+1) \times (n+1)$ матрица,

$$\mathcal{A} = \begin{pmatrix} c_0 & -b_0 & \ldots & 0 & 0 \\ -a_1 & c_1 & \ldots & 0 & 0 \\ 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & \ldots & c_{n-1} & -b_{n-1} \\ 0 & 0 & \ldots & -a_n & c_n \end{pmatrix}.$$

Системы вида (1) возникают при трёхточечной аппроксимации краевых задач для обыкновенных дифференциальных уравнений второго порядка с постоянными и переменными коэффициентами, а также при реализации разностных схем для уравнений в частных производных. В последнем случае обычно требуется решить не единичную задачу (1), а серию задач с различными правыми частями, причем число задач в серии может равняться нескольким десяткам и сотням при числе неизвестных в каждой задаче $n \approx 100$. Поэтому необходимо разработать экономичные методы решения задач вида (1), число действий для которых пропорционально числу неизвестных. Для системы (1) таким методом является метод прогонки.

Возможность построения экономичного метода заключена в специфике системы (1). Соответствующая (1) матрица $\mathcal{A}$ принадлежит к классу разреженных матриц из $(n+1)^2$ элементов ненулевыми являются не более $3n+1$ элементов. Кроме того, она имеет ленточную структуру (является трёхдиагональной матрицей). Такое регулярное расположение ненулевых элементов матрицы $\mathcal{A}$ позволяет получить очень простые расчётные формулы для вычисления решения.

Введем обозначения, полагая $\alpha_1 = \frac{b_0}{c_0}$, $\beta_1 = \frac{f_0}{c_0}$, и запишем (1) в следующем виде:

$$\begin{cases} y_0 - \alpha_1 y_1 = \beta_1, & i = 0, \\ -a_i y_{i-1} + c_i y_i - b_i y_{i+1} = f_i, & 1 \le i \le n-1, \\ -a_n y_{n-1} + c_n y_n = f_n, & i = n. \end{cases} \tag{2}$$

Из первых двух уравнений (2) получим

$$y_1 - \alpha_2 y_2 = \beta_2, \;\; \alpha_2 = \frac{b_1}{c_1 - \alpha_1 a_1}, \;\; \beta_2 = \frac{f_1 + a_1\beta_1}{c_1 - \alpha_1 a_1}.$$

В результате имеем «укороченную» систему

$$\begin{cases} y_1 - \alpha_2 y_2 = \beta_2, & i = 1, \\ -a_i y_{i-1} + c_i y_i - b_i y_{i+1} = f_i, & 2 \le i \le n-1, \\ -a_n y_{n-1} + c_n y_n = f_n, & i = n, \end{cases} \tag{3}$$

которая не содержит неизвестное y_0 и имеет аналогичную (2) структуру. Если эта система будет решена, то неизвестное y_0 найдется по формуле $y_0 = \alpha_1 y_1 + \beta_1$. К системе (3) можно снова применить описанный способ исключения неизвестных. В результате l-го шага получим систему для неизвестных $y_l, \ldots, y_n$:

$$\begin{cases} y_l - \alpha_{l+1} y_{l+1} = \beta_{l+1}, & i = l, \\ -a_i y_{i-1} + c_i y_i - b_i y_{i+1} = f_i, & l+1 \le i \le n-1, \\ -a_n y_{n-1} + c_n y_n = f_n, & i = n, \end{cases} \tag{4}$$

и формулы для нахождения y_i с номерами $i \le l-1$

$$y_i = \alpha_{i+1} y_{i+1} + \beta_{i+1}, \;\; i = \overline{l-1, \ldots, 0}, \tag{5}$$

Коэффициенты α_i и β_i находятся по формулам

$$\alpha_1 = \frac{b_0}{c_0}, \;\; \beta_1 = \frac{f_0}{c_0},$$

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

Полагая в (5) $l = n-1$, получим систему для y_n и y_{n-1}

$$y_{n-1} - \alpha_n y_n = \beta_n, \;\; y_{n-1} + c_n y_n = f_n,$$

из которой найдем $y_n = \beta_{n+1}$, $y_{n-1} = \alpha_n y_n + \beta_n$.

Объединяя эти равенства с (5) получим окончательные формулы для нахождения неизвестных

$$y_i = \alpha_{i+1} y_{i+1} + \beta_{i+1}, \;\; i = \overline{n-1, \ldots, 0}, \tag{6}$$

$$y_n = \beta_{n+1},$$

где α_i и β_i находятся по рекуррентным формулам

$$\alpha_{i+1} = \frac{b_i}{c_i - a_i\alpha_i}, \;\; i = \overline{1, n-1}, \;\; \alpha_1 = \frac{b_0}{c_0}, \tag{7}$$

$$\beta_{i+1} = \frac{f_i + a_i\beta_i}{c_i - a_i\alpha_i}, \quad i = \overline{1,n}, \quad \beta_1 = \frac{f_0}{c_0}. \tag{8}$$

Формулы (6-8) описывают метод Гаусса, который в применении к системе (1) получил специальное название – метод прогонки. Коэффициенты α_i, β_i называют прогоночными коэффициентами, формулы (7-8) описывают прямой ход прогонки, а (6) – обратный ход. Так как значения y_i находятся здесь последовательно при переходе от $i+1$ к i, то формулы (6-8) называют иногда формулами правой прогонки.

Элементарный подсчёт арифметических действий в (6-8) показывает, что реализация метода прогонки по этим формулам требует выполнения $3n$ умножений, $2n+1$ делений и $3n$ сложений и вычитаний. Если не делать различия между арифметическими операциями, то общее их число для метода прогонки есть $Q = 8n+1$. Из этого числа $3n-2$ операции затрачиваются на вычисление α_i и $5n+3$ операций на вычисление β_i и y_i.

Несложно заметить, что коэффициенты α_i не зависят от правой части системы (1), а определяются только коэффициентами разностных уравнений a_i, b_i, c_i. Поэтому, если требуется решить серию задач (1) с различными правыми частями, но с одной и той же матрицей $\mathcal{A}$, то прогоночные коэффициенты α_i вычисляются только при решении первой задачи из серии. Для каждой последующей задачи определяются только коэффициенты β_i и решение y_i, причём используются найденные ранее α_i. Таким образом, на решение только первой из серии задач тратится число арифметических действий $Q = 8n+1$, на решение каждой следующей задачи будет затрачиваться уже только $5n+3$ операции.

В заключение укажем порядок счёта по формулам метода прогонки. Начиная с α_1 и β_1, по формулам (7) и (8) определяются и запоминаются прогоночные коэффициенты α_i и β_i. Затем по формулам (6) находится решение y_i.

Лемма 1. Пусть коэффициенты системы (1) действительны и удовлетворяют условиям $|b_0| \geq 0$, $|a_n| \geq 0$, $|c_0| > 0$, $|c_n| > 0$, $|a_i| > 0$, $|b_i| > 0$, $i = \overline{1,n-1}$,

$$|c_i| \geq |a_i| + |b_i|, \quad i = \overline{1,n-1}, \tag{9}$$

$$|c_0| \geq |b_0|, \quad |c_n| \geq |a_n|, \tag{10}$$

причём хотя бы в одном из неравенств (9) или (10) выполняется строгое неравенство, т. е. матрица $\mathcal{A}$ имеет диагональное преобладание. Тогда для алгоритма (6-8) метода прогонки имеют место неравенства $c_i - a_i\alpha_i \neq 0$, $\alpha_i \leq 1$, $i = \overline{1,n}$ гарантирующие корректность и устойчивость метода.

1.2. Применение метода прогонки для решения ОДУ

Рассмотрим линейное ДУ второго порядка

$$L[y] = y'' + p(x)y'(x) + q(x)y(x) = f(x) \tag{11}$$

на интервале $x \in [a, b]$ с краевыми условиями

$$l_0[y] = \varkappa_1 y(a) + \varkappa_2 y'(a) = f_0, \tag{12}$$

$$l_1[y] = \nu_1 y(b) + \nu_2 y'(b) = f_n.$$

Введем на $[a, b]$ равномерную сетку $x_i = a + ih$, $i = \overline{0, n}$ и заменим (11-12) следующей разностной задачей:

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + \frac{p_i(y_{i+1} - y_{i-1})}{2h} + q_i y_i = f_i, \tag{13}$$

$$\varkappa_1 y_0 + \varkappa_2 \frac{y_1 - y_0}{h} = f_0, \tag{14}$$

$$\nu_1 y_n + \nu_2 \frac{y_n - y_{n-1}}{h} = f_n.$$

Преобразуем выражения (13-14) к виду:

$$\begin{cases} C_0 y_0 - B_0 y_1 = f_0, & i = 0, \\ -A_i y_{i-1} + C_i y_i - B_i y_{i+1} = f_i, & 1 \le i \le n-1, \\ -A_n y_{n-1} + C_n y_n = f_n, & i = n, \end{cases}$$

где

$$C_0 = \varkappa_1 - \frac{\varkappa_2}{h}, \;\; B_0 = -\varkappa_2,$$

$$A_i = \frac{p_i}{2h} - \frac{1}{h^2}, \;\; C_i = \frac{2}{h^2} - q_i, \;\; B_i = -\frac{1}{h^2} - \frac{p_i}{2h},$$

$$A_n = \frac{\nu_2}{h}, \;\; C_n = \nu_1 + \frac{\nu_2}{h}.$$

1.3. Методы минимизации невязки при решения краевых задач для ОДУ

Рассмотрим линейное ДУ второго порядка

$$L[y] = y'' + p(x)y' + q(x)y = f(x), \;\; x \in [a, b] \tag{15}$$

с граничными условиями

$$l_a[y] = \alpha_0 y(a) + \beta_0 y'(a) = A, \tag{16}$$

$$l_b[y] = \alpha_1 y(b) + \beta_1 y'(b) = B.$$

Утверждение. Краевая задача (15-16) имеет единственное решение, если соответствующая однородная задача имеет только тривиальное решение.

Метод коллокации

Будем искать приближённое решение линейной краевой задачи (15-16) в виде функции

$$y_n(x) = \varphi_0(x) + \sum_{i=1}^{n} c_i \varphi_i(x), \tag{17}$$

где определяемые на отрезке $[a,b]$ базисные функции $\varphi_i(x)$, $i=\overline{1,n}$ и дополнительная функция $\varphi_0(x)$ должны быть дважды дифференцируемыми и попарно линейно независимыми. Кроме того, функция $\varphi_0(x)$ должна удовлетворять данным краевым условиям (16), а функции $\varphi_i(x)$ при $i=\overline{1,n}$ – соответствующим однородным краевым условиям, т.е. должны выполняться равенства

$$\begin{cases} \alpha_0\varphi_i(a)+\alpha_1\varphi_i'(a)=0, \quad \forall i\in\mathbb{N}, \\ \beta_0\varphi_i(b)+\beta_1\varphi_i'(b)=0. \end{cases} \tag{18}$$

В таком случае функция $y_n(x)$, определяемая выражением (17), при любых значениях коэффициентов c_i гарантированно удовлетворяет краевым условиям (16).

Представление приближённого решения $y_n(x)$, подобное (17), характерно для многих приближенно-аналитических методов решения краевых задач; главное их различие состоит в том, на какой основе ищутся коэффициенты c_i в линейной комбинации базисных функций $\varphi_i(x)$ выражения (17).

В методе коллокации коэффициенты c_i в представлении (17) приближённого решения $y_n(x)$ подбираются так, чтобы в узлах коллокации x_i таких, что

$$a<x_1<\ldots<x_n<b,$$

значения $y_n(x_i)$ приближённого решения были согласованы с точными значениями $y(x_i)$.

Поскольку точное решение $y(x)$ неизвестно, согласование $y_n(x)$ с $y(x)$ в узлах коллокации x_i проводим подстановкой $y_n(x)$ в уравнение (15). Получаем равенство

$$y_n''(x_i)+p(x_i)y_n'(x_i)+q(x_i)y_n(x_i)=f(x_i), \tag{19}$$

которое, в силу выставляемого требования согласования $y_n(x_i)$ с $y(x_i)$, считаем точным при каждом $i=\overline{1,n}$. Продифференцировав дважды функцию $y_n(x)$ в представлении (17), от равенства (19) переходим к равенству

$$\sum_{j=1}^{n}c_j\varphi_j''(x_i)+p_i\sum_{j=1}^{n}c_j\varphi_j'(x_i)=f_i-\varphi_0''(x_i)-p_i\varphi_0'(x_i)-q_i\varphi_0(x_i). \tag{20}$$

Положим

$$a_{ij}=\varphi_j''(x_i)+p_i\varphi_j'(x_i)+q_i\varphi_j(x_i), \tag{21}$$

$$b_i=f_i-\varphi_0''(x_i)-p_i\varphi_0'(x_i)-q_i\varphi_0(x_i). \tag{22}$$

Тогда (20) приобретает стандартный вид линейной алгебраической системы

$$\sum_{j=1}^{n}a_{ij}c_j=b_i, \quad i=\overline{1,n} \tag{23}$$

относительно коэффициентов $\{c_i\}$.

Успех применения метода коллокации к задаче (15-16), впрочем, как и других приближенно-аналитических методов, сильно зависит от удачного выбора базисных функций $\varphi_i(x)$ в представле-

нии приближённого решения (17). В конкретных задачах выбор таких функций, по возможности, должен опираться на априорные или эмпирические сведения о решении.

Интегральный метод наименьших квадратов

Метод основан на минимизации интеграла

$$I = \int_a^b \psi^2(x, c_1, \ldots, c_n)dx, \tag{24}$$

где $\psi(x, c_1, \ldots, c_n) = L[y_n(x)] - f(x) = L[\varphi_0(x)] + \sum_{i=1}^n c_i L[\varphi_i(x)] - f(x)$ – невязка точного и приближённого решений. Условие минимума I:

$$\int_a^b \psi \frac{\partial \psi}{\partial c_i} = 0, \;\; i = \overline{1, n},$$

$$\begin{cases} c_1(L[\varphi_1], L[\varphi_1]) + \ldots + c_n(L[\varphi_n], L[\varphi_1]) = (f - L[\varphi_0], L[\varphi_1]), \\ \ldots \\ c_1(L[\varphi_1], L[\varphi_n]) + \ldots + c_n(L[\varphi_n], L[\varphi_n]) = (f - L[\varphi_0], L[\varphi_n]). \end{cases} \tag{25}$$

Система (25) имеет единственное решение, если $\{L[\varphi_i]\}$ образуют линейно независимую систему функций.

Определитель Грамма, построенный на базе скалярных произведений системы (25), равен нулю, если функции линейно зависимы.

Дискретный метод наименьших квадратов

Метод основан на минимизации суммы

$$\sum_{i=1}^n \psi^2(x_i, c_1, \ldots, c_n), \;\; x_i \in (a, b). \tag{26}$$

Если в качестве скалярного произведения использовать $(f, g) = \sum_{i=1}^n f(x_i)g(x_i)$, то получим систему (25).

Метод подобластей

Пусть $x_i \in [a, b]$. Потребуем, чтобы

$$\int_{x_{i-1}}^{x_i} \psi(x, c_1, \ldots, c_n)dx = 0, \;\; i = \overline{1, n}.$$

Метод Галёркина

Пусть L – некоторый линейный оператор, действующий в гильбертовом пространстве $\mathcal{H}$, т.е. в полном нормированном пространстве со скалярным произведением $(\circ, \circ)$. Стоит задача приближённого решения операторного уравнения

$$L[y] = f, \tag{27}$$

т.е. задача отыскания некоторого приближения к неизвестному элементу $y \in \mathcal{H}$, соответствующему заданному элементу $f \in \mathcal{H}$. Пусть, далее, $\{\varphi_i(x)\}_{i=1}^{\infty}$ – некоторая полная замкнутая система линейно независимых элементов из $\mathcal{H}$. Ее n первых элементов $\{\varphi_i\}_{i=1}^{n}$ выделяют в $\mathcal{H}$ конечномерное подпространство $\mathcal{H}_n$, в котором и ищется приближённое решение уравнения (27):

$$y_n = \sum_{i=1}^{n} c_i \varphi_i(x) \tag{28}$$

Для удобства будем считать, что элемент $L[y_n]$ принадлежит тому же подпространству $\mathcal{H}_n$. Тогда к тривиальному равенству

$$f = L[y_n] + (f - L[y_n])$$

можно применить одну из центральных теорем теории гильбертовых пространств, согласно которой любой элемент гильбертова пространства может быть представлен в виде суммы определённого элемента подпространства (проекции данного элемента на подпространство) и определённого элемента пространства, ортогонального к выбранному подпространству (реализующего расстояние от исходного элемента до его проекции).

Принадлежность элемента $f - L[y_n]$ ортогональному к $\mathcal{H}_n$ подпространству $\mathcal{H}_n^{\perp}$ означает, что он ортогонален каждому элементу φ_i, входящему в базис пространства $\mathcal{H}_n$. Таким образом, при любом $i = \overline{1,n}$ имеем:

$$(f - L[y_n]) \perp \varphi_i \Leftrightarrow (f - L[y_n], \varphi_i) = 0 \Leftrightarrow (L[y_n], \varphi_i) = (f, \varphi_i).$$

Подставляя сюда выражение y_n (28) и пользуясь простейшими свойствами скалярного произведения, получаем:

$$\left(L\left[\sum_{j=1}^{n} c_j \varphi_j \right], \varphi_i \right) = (f, \varphi_i) \Leftrightarrow \sum_{j=1}^{n} (L[\varphi_j], \varphi_i) c_j = (f, \varphi_i). \tag{29}$$

Метод Галёркина приближённого решения операторного уравнения (27) сводится к нахождению коэффициентов $\{c_i\}_{i=1}^{n}$ линейной комбинации некоторых задаваемых определённым образом линейно независимых функций $\{\varphi_i\}_{i=1}^{n}$ так, чтобы эти коэффициенты удовлетворяли линейной системе

$$\sum_{j=1}^{n} a_{ij} c_j = d_i, \quad i = \overline{1,n}, \tag{30}$$

где

$$a_{ij} = (L[\varphi_j], \varphi_i), \quad d_i = (f, \varphi_i). \tag{31}$$

Вернёмся к рассмотрению краевой задачи (15-16). Естественным для ее рассмотрения пространством является гильбертово пространство $\mathcal{L}_2[a,b]$ функций, интегрируемых на отрезке $[a,b]$ с квадратом. Скалярное произведение здесь определяется равенством

$$(u, v) = \int_a^b u(x) v(x) dx. \tag{32}$$

Будем искать приближённое решение задачи (15-16) в виде задаваемой формулой (17) функции

$y_n(x)$ такой, чтобы она удовлетворяла данным краевым условиям, а это будет, как мы уже знаем, если

$$l_a[\varphi_0] = A, \;\; l_b[\varphi_0] = B,$$

$$l_a[\varphi_i] = 0, \;\; l_b[\varphi_i] = 0, \;\; \forall i = \overline{1,n}.$$

Подставляя $y_n(x)$ в данное уравнение (15), в силу линейности определённого там дифференциального оператора $L[y]$, имеем:

$$L\left[\varphi_0 + \sum_{i=1}^{n} c_i\varphi_i\right] = f \Leftrightarrow L\left[\sum_{i=1}^{n} c_i\varphi_i\right] = f - L[\varphi_0].$$

Следовательно, в таком случае, наличие дополнительного слагаемого φ_0 в представлении приближённого решения (17) вместо представления (28) означает, что равенство (29) нужно переписать в виде

$$\left(L\left[\sum_{i=1}^{n} c_j\varphi_j\right], \varphi_i\right) = (f - L[\varphi_0], \varphi_i),$$

а это, в свою очередь, вносит поправку в выражение d_i, зафиксированное равенством (31). Согласно этому замечанию и определению (10.61) скалярного произведения, находим выражение свободного члена d_i i-го уравнения системы (31):

$$d_i = (f - L[\varphi_0], \varphi_i) = \int_a^b [f(x) - \varphi_0''(x) - p(x)\varphi_0'(x) - q(x)\varphi_0(x)]\varphi_i(x)dx. \tag{33}$$

Выразим коэффициенты СЛАУ (30) в соответствии с записанной в (31) формулой:

$$a_{ij} = (L[\varphi_i], \varphi_i) = \int_a^b [\varphi_j''(x) + p(x)\varphi_j'(x) + q(x)\varphi_j(x)]\varphi_i(x)dx =$$
$$= \int_a^b \varphi_j''(x)\varphi_i(x)dx + \int_a^b p(x)\varphi_j'(x)dx + \int_a^b q(x)\varphi_j(x)\varphi_i(x)dx.$$

Преобразуем «по частям» первый интеграл в выражении и получим

$$a_{ij} = \varphi_i(b)\varphi_j'(b) - \varphi_i(a)\varphi_j'(a) - \int_a^b \varphi_j'(x)\varphi_i'(x)dx + \int_a^b p(x)\varphi_j'(x)\varphi_i(x)dx + \int_a^b q(x)\varphi_j(x)\varphi_i(x)dx. \tag{34}$$

Выполненное преобразование интеграла, во-первых, активизировало роль краевых условий при построении приближённого решения $y_n(x)$ в форме (17), во-вторых, позволило ослабить требования к гладкости базисных функций $\varphi_i(x)$, поскольку теперь, как видно из (34), у них могут быть даже разрывы производных. Подмеченный факт говорит о том, что методом Галёркина при подходящем выборе базисных функций можно находить решения краевых задач, определённые в каком-либо обобщённом смысле.

2. Постановка задачи

- Решить краевую задачу $y'' + y' - 2y = 0$, $x \in [0, 1]$, $y(0) = 2$, $y(1) = \exp(1) + \exp(2)$, используя метод прогонки.

- Аппроксимировать функцию Рунге с помощью кубического сплайна при глобальном способе задания наклонов, используя метод прогонки для решения системы для определения наклонов.

- Решить краевую задачу $L[y] = y'' + \left(1 + x^2\right) y = -1$, $x \in [-1, 1]$, $y(-1) = 0$, $y(1) = 0$. Базисные функции: $\varphi_0(x) = 0$, $\varphi_i(x) = x^{2i-2}\left(1 - x^2\right)$, $i \in \mathbb{N}$. Точки коллокации: $x_0 = 0$, $x_1 = -\frac{1}{2}$, $x_2 = \frac{1}{2}$. Приближённое решение искать в виде: $y_2(x) = c_1\left(1 - x^2\right) + c_2 x^2\left(1 - x^2\right)$.

  Решить краевую задачу, используя интегральный и дискретный метод наименьших квадратов.

- Решить краевую задачу $L[y] = y'' + y = -x$, $x \in [0, 1]$, $y(0) = 0$, $y(1) = 0$, используя метод Галеркина. Базисные функции: $\varphi_0(x) = 0$, $\varphi_i(x) = x^i(1 - x)$, $i \in \mathbb{N}$. Найти приближённые решения y_2 и y_3.

3. Программа

3.1. Задание А

```
#include <iostream>
#include <math.h>
using namespace std;

class NDSolve{
private:
    double *a, *b, *x, *D;
    double left_b, right_b, cond1, cond2, A, B, C, p, q, h;
    int size;
public:
    NDSolve(double in1, double in2, double l, double r, double p1, double q1){
        size = 10;
        a = new double[size+1];
        b = new double[size+1];
        x = new double[size+1];
        D = new double[size];
        left_b = l;
        right_b = r;
        cond1 = in1;
        cond2 = in2;
        h = (right_b - left_b)/(size-1);
        p = p1;
        q = q1;
    }
    void CreateMatrix();
    void Reverse_stroke();
    void Forward_stroke();
    double Function(double t);
};

int main(){
    NDSolve object(2, exp(1) + exp(-2), 0, 1, 1, -2);
    object.CreateMatrix();
    object.Forward_stroke();
    object.Reverse_stroke();
    return 0;
}

double NDSolve::Function(double t){
    return 0;
}
void NDSolve::CreateMatrix(){
    B =  (0.5*p*h + 1);
    C = (-2 + q*h*h);
    A =  (-0.5*p*h + 1);
    for(int i = 0; i < size;i++){
        D[i] = h*h*Function(left_b + h*i);
```

```
    }
}
void NDSolve::Forward_stroke(){
    a[1] = 0;
    b[1] = cond1;
    for (int i = 1; i < size - 1; i++) {
        a[i + 1] = -1 * B / (C + A * a[i]);
        b[i + 1] = (D[i] - A * b[i]) / (C + A * a[i]);
    }
}
void NDSolve::Reverse_stroke(){
    x[size-1] = cond2;
    cout << "{{" << (size-1)*h << ", " << x[size-1] << "}";
    for (int i = size-1; i > 0; i--) {
        x[i - 1] = a[i] * x[i] + b[i];
        cout << ", {" << (i - 1)*h << ", " << x[i-1] << "}";
    }
}
```

3.2. Задание Б

```
#include <iostream>
using namespace std;

class Spline{
private:
    double *a, *b, *m, *D;
    double left_b, right_b, A{}, B{}, C{}, h;
    int size;
public:
    explicit Spline(int N){
        size = N;
        a = new double[size+1];
        b = new double[size+1];
        m = new double[size+1];
        D = new double[size];
        left_b = -1;
        right_b = 1;
        h = (right_b - left_b)/(size-1);
    }
    void CreateMatrix();
    void Reverse_stroke();
    void Forward_stroke();
    double Values(double x, int index);
    void Spline1();
    static double R(double t);
};
double Spline::R(double t){
    return  1.0 / (1.0 + 25.0 * t * t);
}
void Spline::CreateMatrix(){
    B =  1.0;
    C = 4.0;
    A =  1.0;
    D[0] = (-11*R(left_b) + 18*R(left_b + h) - 9*R(left_b + 2*h) + 2*R(left_b + 3*h))/h/6.0;
    for(int i = 1; i < size-1;i++){
        D[i] = 3*(R(left_b + h*(i+1))  - R(left_b + h*(i-1)))/h;
    }
    D[size-1] = (11*R(right_b) - 18*R(right_b - h) + 9*R(right_b - 2*h) - 2*R(right_b - 3*h))/h/6.0;
}
void Spline::Forward_stroke(){
    a[1] = 0;
    b[1] = R(-1);
    for (int i = 1; i < size - 1; i++) {
        a[i + 1] = -1 * B / (C + A * a[i]);
        b[i + 1] = (D[i] - A * b[i]) / (C + A * a[i]);
    }
}
void Spline::Reverse_stroke(){
    m[size-1] = R(1);
```

```
    for (int i = size-1; i > 0; i--) {
        m[i - 1] = a[i] * m[i] + b[i];
    }
}
double Spline::Values(double x, int index){
    double data_x[size+1];
    double data_y[size + 1];

    for (int i = 0; i <= size; i++){
        data_x[i] = left_b + h*i;
        data_y[i] = R(left_b + h*i);
    }
    double s1, s2, s3, s4;
    double xi = data_x[index];
    double xii = data_x[index + 1];
    s1 = (pow((xii - x), 2) * (2 * (x - xi) + h)) * data_y[index] / (h*h*h);
    s2 = (pow((x - xi), 2) * (2 * (xii - x) + h)) * data_y[index + 1] /  (h*h*h);
    s3 = pow((xii - x), 2) * (x - xi) * m[index] /  (h*h);
    s4 = pow((x - xi), 2) * (x - xii) * m[index + 1] /  (h*h);
    return s1 + s2 + s3 + s4;
}
void Spline::Spline1(){
    int M = 25;
    for(int i = 0; i <= size-2; i++){
        for(int j = 0; j <= M; j++){
            cout << "{" << left_b + i*h + j*h/M << ", " << Values(left_b + h * i + h / M * j, i) << "}, ";
        }
    }
}
int main(){
    Spline object(5);
    object.CreateMatrix();
    object.Forward_stroke();
    object.Reverse_stroke();
    object.Spline1();
    return 0;
}
```

4. Результаты вычислений

4.1. Задание A

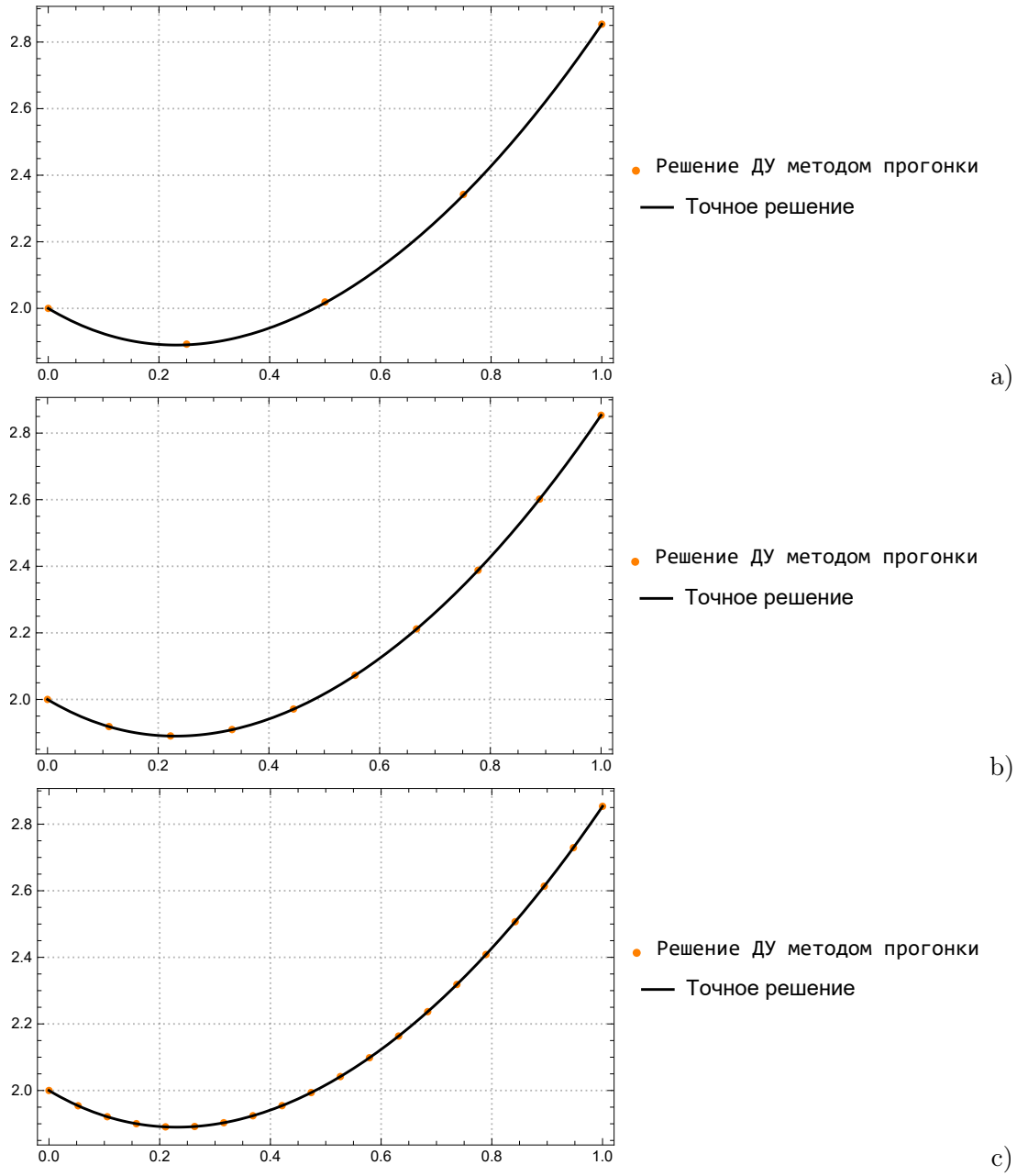

Рис. 1. Точное и численное решения краевой задачи $y'' + y' - 2y = 0$, $x \in [0, 1]$, $y(0) = 2$, $y(1) = \exp(1) + \exp(2)$ методом прогонки для a) $N = 5$, b) $N = 10$, c) $N = 20$.

4.2. Задание Б

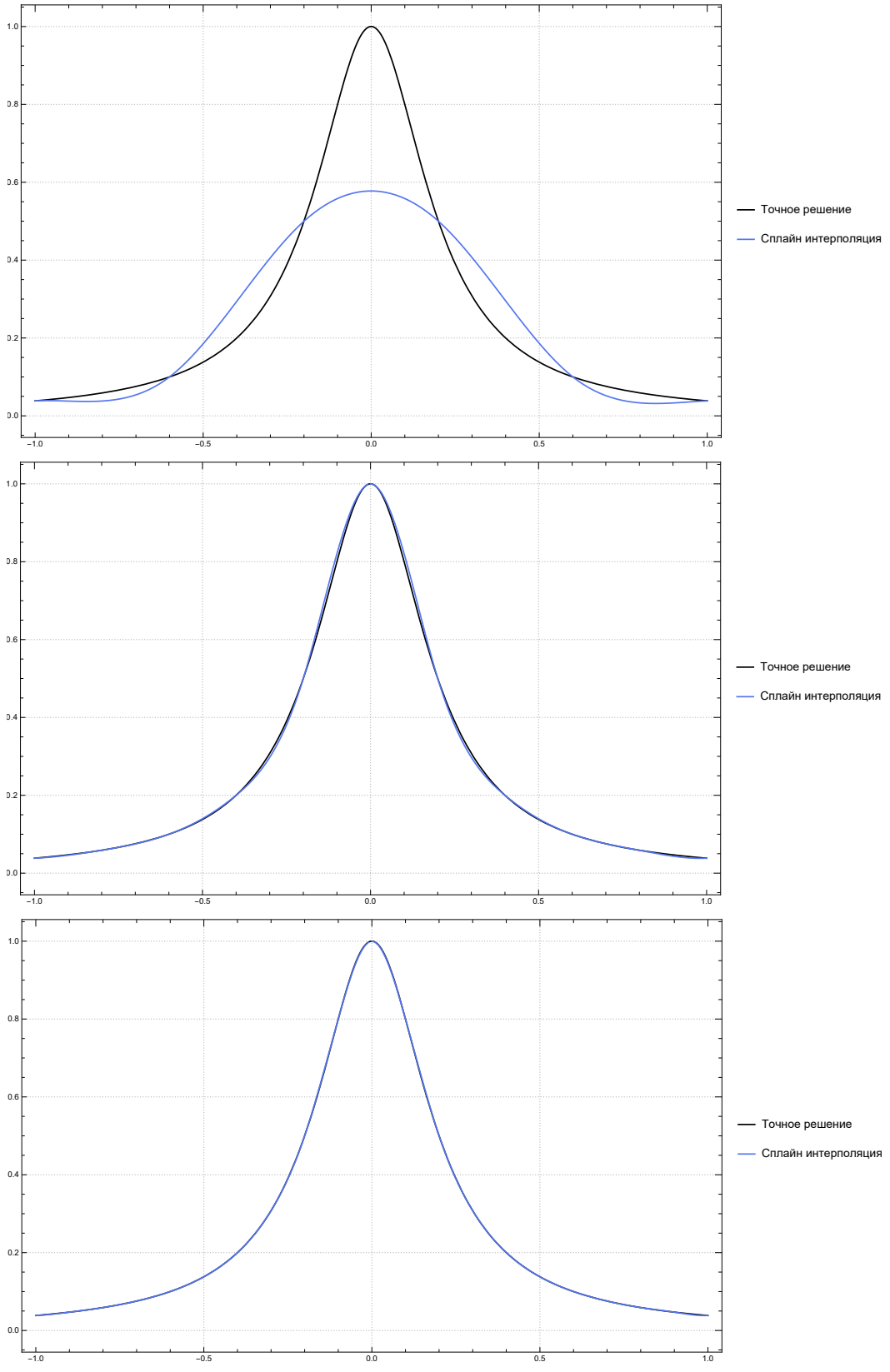

Рис. 2. Аппроксимация функции Рунге $f(x) = \frac{1}{1+25x^2}$ кубическими сплайнами при глобальном способе задании наклонов для a) $N = 5$, b) $N = 10$, c) $N = 20$ частичных отрезков.

4.3. Задание B

Аппроксимирующие функции

- Метод коллокации:

$$y(x) = \frac{16}{17}\left(1 - x^2\right).$$

- Интегральные метод наименьших квадратов:

$$y(x) = \frac{31755933}{34046644}\left(1 - x^2\right) - \frac{2321319}{34046644}x^2\left(1 - x^2\right).$$

- Дискретный метод наименьших квадратов:

$$y(x) = \frac{16}{17}\left(1 - x^2\right).$$

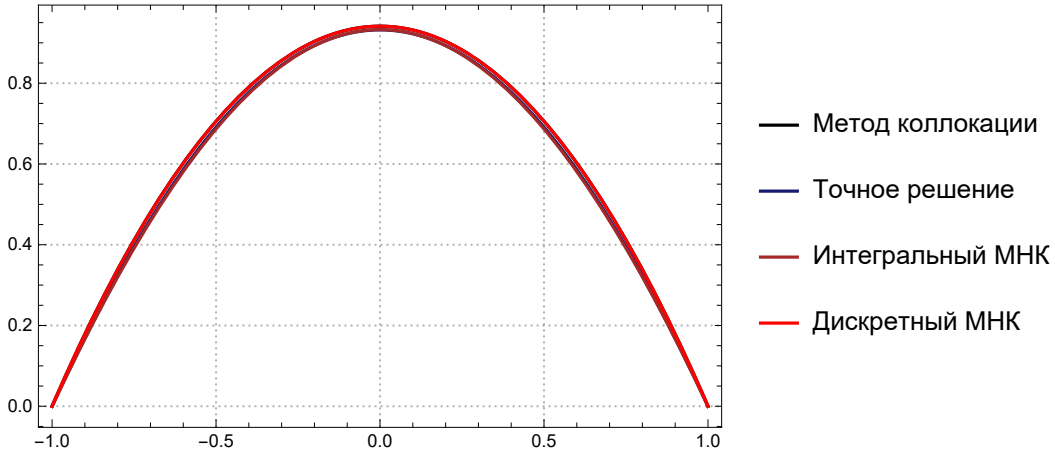

Рис. 3. Точное и численное решения краевой задачи $L[y] = y'' + \left(1 + x^2\right)y = -1$, $x \in [-1, 1]$, $y(-1) = 0$, $y(1) = 0$ для базисных функций $\varphi_0(x) = 0$, $\varphi_i(x) = x^{2i-2}\left(1 - x^2\right)$, $i \in \mathbb{N}$ и точек коллокации $x_0 = 0$, $x_1 = -\frac{1}{2}$, $x_2 = \frac{1}{2}$ при использовании метода коллокации, интегрального и дискретного МНК.

4.4. Задание Г

Аппроксимирующие функции

- Метод Галеркина для $n = 2$:

$$y_2(x) = \frac{71}{369}x(1-x) + \frac{7}{41}x^2(1-x).$$

- Метод Галеркина для $n = 3$:

$$y_3(x) = \frac{13811}{73554}x(1-x) + \frac{2380}{12259}x^2(1-x) - \frac{7}{299}x^3(1-x).$$

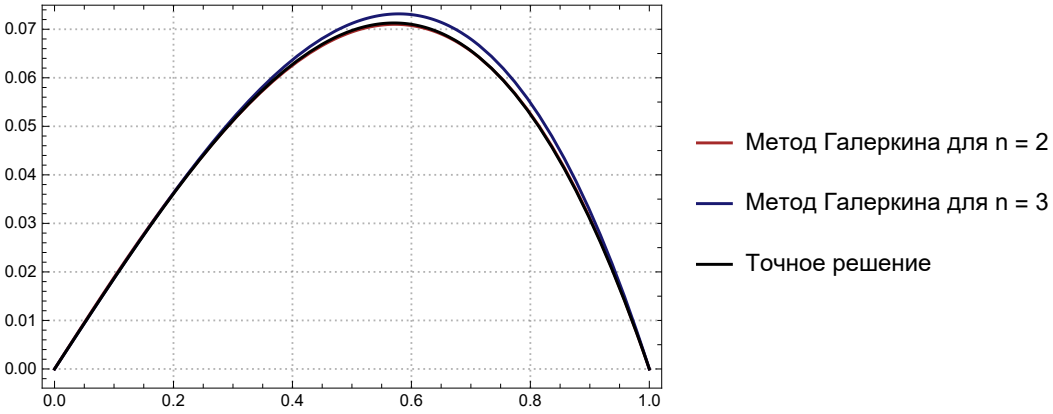

Рис. 4. Точное и численное решение краевой задачи $L[y] = y'' + y = -x$, $x \in [0, 1]$, $y(0) = 0$, $y(1) = 0$ для базисных функций $\varphi_0(x) = 0$, $\varphi_i(x) = x^i(1-x)$, $i \in \mathbb{N}$ при использовании метода Галеркина

5. Вывод

В лабораторной работе рассмотрены такие способы решения краевых задач, как метод прогонки, метод коллокации, интегральный и дискретный методы наименьших квадратов и метод Галеркина; была получена аппроксимация функции Рунге $f(x) = \frac{1}{1+25x^2}$ с помощью кубического сплайна при глобальном способе задания наклонов.

Точность метода Галеркина значительно возросла при использовании большего числа базисных функций. При решении краевой задачи $L[y] = y'' + \left(1 + x^2\right) y = -1$, $x \in [-1, 1]$, $y(-1) = 0$, $y(1) = 0$ для базисных функций $\varphi_0(x) = 0$, $\varphi_i(x) = x^{2i-2}\left(1 - x^2\right)$, $i \in \mathbb{N}$ и точек коллокации $x_0 = 0$, $x_1 = -\frac{1}{2}$, $x_2 = \frac{1}{2}$ методом коллокации, интегральным и дискретным МНК, наибольшую точность показал интегральный МНК.