

Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Лабораторная работа № 1

по дисциплине
«Математические модели»

Выполнил:

Ферапонтов М.В.

Группа:

гр. 3530904/00104

Проверил:

Воскобойников С. П.

Санкт-Петербург
2023

Содержание

1	Вступление	2
1.1	Постановка задачи	2
1.2	Используемое ПО	2
2	Основная часть	3
2.1	Интегро-интеполяционный метод (метод баланса)	3
2.2	Метод прогонки	5
2.3	Оценка погрешности	7
2.3.1	Невязка разностной схемы	7
2.3.2	Структура погрешности разностной схемы	7
2.3.3	Вклад от погрешности решения системы алгебраических уравнений	8
2.3.4	Разложение невязки	8
2.3.5	Зависимость погрешности от числа разбиений	12
2.4	Тестирование	13
2.4.1	Метод прогонки	13
2.4.2	Интегро-интерполяционный метод	13
3	Заключение	16
3.1	Вывод	16
3.2	Код	17

1 Вступление

1.1 Постановка задачи

Вариант СР. Используя интегро-интерполяционный метод (метод баланса), разработать программу для моделирования стационарного распределения температуры в полом цилиндре, описываемого математической моделью вида:

$$-\left[\frac{1}{r}\frac{d}{dr}\left(rk(r)\frac{du}{dr}\right) - q(r)u\right] = f(r), \quad r \in [R_L, R_R], \quad R_L > 0, \\ 0 < c_1 \leq k(r) \leq c_2, \quad 0 \leq q(r)$$

Граничные условия:

$$k \frac{du}{dr} \Big|_{r=R_L} = -\nu_1 \qquad -k \frac{du}{dr} \Big|_{r=R_R} = -\nu_2$$

1.2 Используемое ПО

1. [Boost library](#) - библиотека для тестирования и других функций
2. [GSL](#) - GNU Scientific Library. Математическая библиотека для C и C++.

2 Основная часть

2.1 Интегро-интеполяционный метод (метод баланса)

Введем основную сетку, где N - число разбиений.

$$r_0 < r_1 < \dots < r_N, \quad r_i \in [R_L, R_R], \quad r_0 = R_L, \quad r_N = R_R$$

$$h_i = r_i - r_{i-1}, \quad i = 1, 2, \dots, N$$

$$r_{r-0.5} = \frac{r_i - r_{i-1}}{2}, \quad i = 1, 2, \dots, N$$

Введем дополнительную сетку:

$$\bar{h}_i = \begin{cases} \frac{h_i+1}{2}, & i = 0 \\ \frac{h_i+h_{i+1}}{2}, & i = 1, 2, \dots, N-1 \\ \frac{h_i}{2}, & i = N \end{cases}$$

Проведем аппроксимацию начального уравнения.

Для $i = 0$

$$\begin{aligned} & - \int_{r_i}^{r_{i+0.5}} \left[\frac{d}{dr} \left(rk(r) \frac{du(r)}{dr} \right) - rq(r)u(r) \right] dr = \int_{r_i}^{r_{i+0.5}} rf(r) dr, \\ & - \left[rk(r) \frac{du(r)}{dr} \Big|_{r=r_{i+0.5}} - rk(r) \frac{du(r)}{dr} \Big|_{r_i} - \int_{r_i}^{r_{i+0.5}} rq(r)u(r) dr \right] = \int_{r_i}^{r_{i+0.5}} rf(r) dr, \end{aligned}$$

Формула центральных разностей:

$$\frac{du(r)}{dr} \Big|_{r=r_{i+0.5}} \approx \frac{v_{i+1} - v_i}{h_{i+1}},$$

Граничное условие:

$$k(r) \frac{du(r)}{dr} \Big|_{r=R_L} = -\nu_1,$$

Формула левых прямоугольников:

$$\int_{r_i}^{r_{i+0.5}} r \varphi_i dr = \bar{h}_i r_i \varphi_i$$

Получаем разностную схему для $i = 0$:

$$- \left[r_{i+0.5} \cdot k_{i+0.5} \frac{v_{i+1} - v_i}{h_{i+1}} - r_i \cdot (-\nu_1) - \bar{h}_i r_i q_i v_i \right] = \bar{h}_i r_i f_i$$

Для $i = 1, 2, \dots, N-1$

$$- \int_{r_{i-0.5}}^{r_{i+0.5}} \left[\frac{d}{dr} \left(rk(r) \frac{du(r)}{dr} \right) - rq(r)u(r) \right] dr = \int_{r_{i-0.5}}^{r_{i+0.5}} rf(r) dr,$$

$$\begin{aligned}
& - \left[rk(r) \frac{du(r)}{dr} \Big|_{r=r_{i+0.5}} - rk(r) \frac{du(r)}{dr} \Big|_{r_{i-0.5}} - \int_{r_{i-0.5}}^{r_{i+0.5}} rq(r)u(r) dr \right] = \int_{r_{i-0.5}}^{r_{i+0.5}} rf(r) dr, \\
& \frac{du(r)}{dr} \Big|_{r=r_{i-0.5}} \approx \frac{v_i - v_{i-1}}{h_i} \\
& \int_{r_{i-0.5}}^{r_{i+0.5}} r\varphi_i dr = \hbar r_i \varphi_i
\end{aligned}$$

Получаем разностную схему для $i = 1, 2, \dots, N-1$:

$$- \left[r_{i+0.5} \cdot k_{i+0.5} \frac{v_{i+1} - v_i}{h_{i+1}} - r_{i-0.5} k_{i-0.5} \frac{v_i - v_{i-1}}{h_i} - \hbar r_i q_i v_i \right] = \hbar r_i f_i$$

Для $i = N$:

$$\begin{aligned}
& - \int_{r_{i-0.5}}^{r_i} \left[\frac{d}{dr} \left(rk(r) \frac{du(r)}{dr} \right) - rq(r)u(r) \right] dr = \int_{r_{i-0.5}}^{r_i} rf(r) dr, \\
& - \left[rk(r) \frac{du(r)}{dr} \Big|_{r=r_i} - rk(r) \frac{du(r)}{dr} \Big|_{r_{i-0.5}} - \int_{r_{i-0.5}}^{r_i} rq(r)u(r) dr \right] = \int_{r_{i-0.5}}^{r_i} rf(r) dr, \\
& \frac{du(r)}{dr} \Big|_{r=r_{i-0.5}} \approx \frac{v_i - v_{i-1}}{h_i}, \\
& -k(r) \frac{du(r)}{dr} \Big|_{r=R_R} = -\nu_2 \\
& \int_{r_{i-0.5}}^{r_i} r\varphi(r) dr \approx \hbar r_i \varphi_i
\end{aligned}$$

Получаем разностную схему для $i=N$:

$$- \left[-r_i \cdot (-\nu_2) - r_{i-0.5} k_{i-0.5} \cdot \frac{v_i - v_{i-1}}{h_i} - \hbar r_i q_i u_i \right] = \hbar r_i f_i$$

Сгруппируем полученные уравнения:

$$\begin{aligned}
& - \left[r_{i+0.5} \cdot k_{i+0.5} \frac{v_{i+1} - v_i}{h_{i+1}} - r_i \cdot (-\nu_1) - \hbar r_i q_i v_i \right] = \hbar r_i f_i \quad i = 0 \\
& - \left[r_{i+0.5} \cdot k_{i+0.5} \frac{v_{i+1} - v_i}{h_{i+1}} - r_{i-0.5} k_{i-0.5} \frac{v_i - v_{i-1}}{h_i} - \hbar r_i q_i v_i \right] = \hbar r_i f_i \quad i = 1, 2, \dots, N-1 \\
& - \left[-r_i \cdot (-\nu_2) - r_{i-0.5} k_{i-0.5} \cdot \frac{v_i - v_{i-1}}{h_i} - \hbar r_i q_i u_i \right] = \hbar r_i f_i \quad i = N
\end{aligned}$$

После аппроксимации уравнения можно представить в виде системы из трёх-диагональной матрицы где a , c , b - диагонали матрица A и вектора g . Элементы матрицы:

Для $i = 0$

$$c_i = r_{i+0.5} \frac{k_{i+0.5}}{h_{i+1}} + \hbar r_i q_i \quad b_i = -r_{i+0.5} \cdot \frac{k_{i+0.5}}{h_{i+1}} \quad g_i = \hbar r_i f_i + r \nu_1$$

Для $i = 1, 2, \dots, N-1$

$$a_i = -r_{i-0.5} \frac{k_{i-0.5}}{h_i} \quad c_i = r_{i-0.5} \frac{k_{i-0.5}}{h_i} + r_{i+0.5} \frac{k_{i+0.5}}{h_{i+1}} + \hbar_i r_i q_i \quad b_i = -r_{i+0.5} \frac{k_{i+0.5}}{h_{i+1}} \\ g_i = \hbar_i r_i f_i$$

Для $i = N$:

$$a_i = -r_{i-0.5} \frac{k_{i-0.5}}{h_i} \quad c_i = r_{i-0.5} \frac{k_{i-0.5}}{h_i} + \hbar_i r_i q_i \quad g_i = \hbar_i r_i f_i + r_i \cdot \nu_2$$

2.2 Метод прогонки

Метод прогонки это простой способ решать трёхдиагональные системы.

$$\begin{pmatrix} c_1 & b_1 & & & & 0 \\ a_2 & c_2 & b_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots \\ & & & & a_{n-1} & c_{n-1} & b_{n-1} \\ 0 & & & & a_n & c_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ \vdots \\ \vdots \\ g_{n-1} \\ g_n \end{pmatrix}$$

Этап 1

Строка 1. Разделим первую строку на c_1 :

$$c_1 x_1 + b_1 x_2 = g_1 \\ x_1 + \gamma_1 x_2 = \rho_1, \quad \gamma_1 = \frac{b_1}{c_1}, \quad \rho_1 = \frac{g_1}{c_1}$$

$$\begin{pmatrix} 1 & \gamma_1 & & & & 0 \\ a_2 & c_2 & b_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots \\ & & & & a_{n-1} & c_{n-1} & b_{n-1} \\ 0 & & & & a_n & c_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} \rho_1 \\ g_2 \\ \vdots \\ \vdots \\ \vdots \\ g_{n-1} \\ g_n \end{pmatrix}$$

Строки от 2 до N-1. Здесь представлена общая формула для всех строк в промежуток

$$a_n x_{n-1} + c_n x_n + b_n x_{n+1} = g_n, \quad n = 2, 3, \dots, N-1$$

Умножим $n-1$ строку на a_n и вычтем из строки под номером n . Получим строку

$$(c_n - a_n \cdot \gamma_{n-1}) x_n + c_n x_{n+1} = g_n - a_n \rho_{n-1}$$

Разделим на $(c_n - a_n \cdot \gamma_{n-1})$

$$x_n + \frac{b_n}{c_n - a_n \gamma_{n-1}} x_{n+1} = \frac{g_n - a_n \rho_{n-1}}{c_n - a_n \gamma_{n-1}}$$

$$x_n + \gamma_n x_{n+1} = \rho_n, \quad \gamma_n = \frac{b_n}{c_n - a_n \gamma_{n-1}}, \quad \rho_n = \frac{g_n - a_n \rho_{n-1}}{c_n - a_n \gamma_{n-1}}$$

$$\begin{pmatrix} 1 & \gamma_1 & & & & & 0 \\ 0 & 1 & \gamma_2 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & a_{n-1} & c_{n-1} & b_{n-1} \\ 0 & & & & a_n & c_n & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \vdots \\ \vdots \\ g_{n-1} \\ g_n \end{pmatrix}$$

Строка N.

$$a_n x_{n-1} + c_n x_n = g_n$$

$$x_n = \rho_n, \quad \rho_n = \frac{r_n - a_n \rho_{n-1}}{c_n - a_n \gamma_{n-1}}$$

$$\begin{pmatrix} 1 & \gamma_1 & & & & & 0 \\ 0 & 1 & \gamma_2 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & 0 & 1 & \gamma_{n-1} \\ 0 & & & & 0 & 1 & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \vdots \\ \vdots \\ \rho_{n-1} \\ \rho_n \end{pmatrix}$$

Этап 2

Чтобы узнать значения вектора x нам нужно "подняться" по уже вычисленным значениям.

$$x_n = \rho_n$$

$$x_i = \rho_i - \gamma_i x_{i+1}, \quad i = n-1, n-2, \dots, 1$$

2.3 Оценка погрешности

2.3.1 Невязка разностной схемы

$$Av = g, \quad A - (N+1)r(N+1), \quad v, g \in R^{(N+1)}$$

Пусть v - это точное решение разностной схемы, u - точное решение дифференциального уравнения, \tilde{v} - полученное решение разностной схемы.

Ищем погрешность решения разностной схемы:

$$\varepsilon = \tilde{v} - u$$

Введем обозначения:

- Погрешность решения системы линейных алгебраических уравнений

$$z = \tilde{v} - v$$

- Погрешность от аппроксимации дифференциального уравнения разностной схемой

$$\zeta = v - u$$

- Невязка разностной схемы

$$\xi = g - Au$$

- Невязка алгебраической системы

$$r = g - A\tilde{v}$$

2.3.2 Структура погрешности разностной схемы

$$\|\varepsilon\| = \|\tilde{v} - u\| = \|\tilde{v} - v + v - u\| = \|z + \zeta\| \leq \|z\| + \|\zeta\|$$

Для $\|\zeta\|$:

$$\begin{aligned}\xi &= g - Au = A(A^{-1}g - u) = A(v - u) \\ A\zeta &= \xi\end{aligned}$$

Тем самым погрешность от аппроксимации дифференциального уравнения разностной схемой, связана с невязкой разностной схемы:

$$\zeta = A^{-1}\xi$$

Для $\|z\|$:

$$\begin{aligned}r &= g - A\tilde{v} = A(A^{-1}g - \tilde{v}) = A(v - \tilde{v}) \\ Az &= r\end{aligned}$$

Тем самым погрешность решения системы линейных алгебраических уравнений, связана с невязкой алгебраической системы:

$$z = A^{-1}r$$

Подставим в наше неравенство, тем самым получаем:

$$\|\varepsilon\| \leq \|A^{-1}r\| + \|A^{-1}\xi\| \leq \|A^{-1}\| (\|r\| + \|\xi\|) \quad \|A^{-1}\| < C$$

2.3.3 Вклад от погрешности решения системы алгебраических уравнений

$$\|z\| \leq \|A^{-1}\| \|r\| = \|A\| \|A^{-1}\| \frac{\|r\|}{\|A\|}$$

Знаем что:

$$\|A\| \geq \frac{\|g\|}{\|v\|}$$

Из этого получаем:

$$\|z\| \leq \|A\| \|A^{-1}\| \frac{\|r\|}{\|A\|} \|v\|$$

$$\text{cond}(A) = \|A\| \|A^{-1}\|$$

$$\frac{\|r\|}{\|A\|} \sim \varepsilon_M$$

$$\|z\| \leq \text{cond}(A) \varepsilon_M \|v\|$$

2.3.4 Разложение невязки

Разностная схема

$$\begin{aligned} - \left[\frac{1}{r} \frac{d}{dr} \left(rk \frac{du}{dr} \right) - qu(r) \right] &= f_i \\ - \left[\frac{d}{dr} \left(rk \frac{du}{dr} \right) - rqu(r) \right] &= rf_i \end{aligned}$$

Введем новые обозначения:

$$\tilde{k} = rk \quad \tilde{q} = rq \quad \tilde{f} = rf$$

Получим:

$$- \left[\frac{d}{dr} \left(\tilde{k} \frac{du}{dr} \right) - \tilde{q}u(r) \right] = \tilde{f}_i$$

Еще раз напишем разностную схему:

$$\begin{aligned} - \left[\tilde{k}_{i+0.5} \frac{u_{i+1} - u_i}{h} + \nu_1 - \frac{h}{2} \tilde{q}_i v_i \right] &= \frac{h}{2} \tilde{f}_i \quad i = 0 \\ - \left[\tilde{k}_{i+0.5} \frac{u_{i+1} - u_i}{h} - \tilde{k}_{i-0.5} \frac{u_i - u_{i-1}}{h} - h \tilde{q}_i v_i \right] &= h \tilde{f}_i \quad i = 1, 2, \dots, N-1 \\ - \left[\nu_2 - \tilde{k}_{i-0.5} \cdot \frac{u_i - u_{i-1}}{h} - \frac{h}{2} \tilde{q}_i v_i \right] &= \frac{h}{2} \tilde{f}_i \quad i = N \end{aligned}$$

На левой границе интервала уравнение для невязки выглядит следующим образом:

$$\xi_i = \frac{h}{2} \tilde{f}_i + \left[\tilde{k}_{i+0.5} \frac{u_{i+1} - u_i}{h} + \nu_1 - \frac{h}{2} \tilde{q}_i u_i \right]$$

Внутри интервала:

$$\xi_i = hf_i + \left[\tilde{k}_{i+0.5} \frac{u_{i+1} - u_i}{h} - \tilde{k}_{i-0.5} \frac{u_i - u_{i-1}}{h} - h\tilde{q}_i u_i \right]$$

На правой границе:

$$\xi_i = \frac{h}{2} \tilde{f}_i + \left[\nu_2 - \tilde{k}_{i-0.5} \frac{u_i - u_{i-1}}{h} - \frac{h}{2} \tilde{q}_i u_i \right]$$

Найдем разложение разностной схемы

$$\begin{aligned} u_i &= u(x_i + h) = u_i + h \frac{du_i}{dr} + \frac{h^2}{2} \frac{d^2 u_i}{dr^2} + \frac{h^3}{6} \frac{d^3 u_i}{dr^3} + \frac{h^4}{24} \frac{d^4 u_i}{dr^4} + \mathcal{O}(h^5) \\ \frac{u_{i+1} - u_i}{h} &= \frac{du_i}{dr} + \frac{h}{2} \frac{d^2 u_i}{dr^2} + \frac{h^2}{6} \frac{d^3 u_i}{dr^3} + \frac{h^3}{24} \frac{d^4 u_i}{dr^4} + \mathcal{O}(h^4) \\ \tilde{k}_{i+\frac{1}{2}} &= \tilde{k}(x_i + \frac{h}{2}) = \tilde{k}_i + \frac{h}{2} \frac{d\tilde{k}_i}{dr} + \frac{h^2}{8} \frac{d^2 \tilde{k}_i}{dr^2} + \frac{h^3}{48} \frac{d^3 \tilde{k}_i}{dr^3} + \mathcal{O}(h^3) \end{aligned}$$

Получаем:

$$\begin{aligned} \tilde{k}_{i+\frac{1}{2}} \frac{u_{i+1} - u_i}{h} &= \tilde{k}_i \frac{du_i}{dr} \\ &+ h \left[\frac{1}{2} \tilde{k}_i \frac{d^2 u_i}{dr^2} + \frac{1}{2} \frac{d\tilde{k}_i}{dr} \frac{du_i}{dr} \right] \\ &+ h^2 \left[\frac{1}{6} \tilde{k}_i \frac{d^3 u_i}{dr^3} + \frac{1}{4} \frac{d\tilde{k}_i}{dr} \frac{d^2 u_i}{dr^2} + \frac{1}{8} \frac{d^2 \tilde{k}_i}{dr^2} \frac{du_i}{dr} \right] \\ &+ h^3 \left[\frac{1}{24} \tilde{k}_i \frac{d^4 u_i}{dr^4} + \frac{1}{12} \frac{d\tilde{k}_i}{dr} \frac{d^3 u_i}{dr^3} + \frac{1}{16} \frac{d^2 \tilde{k}_i}{dr^2} \frac{d^2 u_i}{dr^2} + \frac{1}{48} \frac{d^3 \tilde{k}_i}{dr^3} \frac{du_i}{dr} \right] \\ &+ \mathcal{O}(h^4) \end{aligned}$$

$$\begin{aligned} u_{i-1} &= u(x_i - h) = u_i - h \frac{du_i}{dr} + \frac{h^2}{2} \frac{d^2 u_i}{dr^2} - \frac{h^3}{6} \frac{d^3 u_i}{dr^3} + \frac{h^4}{24} \frac{d^4 u_i}{dr^4} + \mathcal{O}(h^5) \\ \frac{u_i - u_{i-1}}{h} &= \frac{du_i}{dr} - \frac{h}{2} \frac{d^2 u_i}{dr^2} + \frac{h^2}{6} \frac{d^3 u_i}{dr^3} - \frac{h^3}{24} \frac{d^4 u_i}{dr^4} + \mathcal{O}(h^4) \\ \tilde{k}_{i-\frac{1}{2}} &= \tilde{k}(x_i - \frac{h}{2}) = \tilde{k}_i - \frac{h}{2} \frac{d\tilde{k}_i}{dr} + \frac{h^2}{8} \frac{d^2 \tilde{k}_i}{dr^2} - \frac{h^3}{48} \frac{d^3 \tilde{k}_i}{dr^3} + \mathcal{O}(h^3) \end{aligned}$$

Получаем:

$$\begin{aligned}
\tilde{k}_{i-\frac{1}{2}} \frac{u_i - u_{i-1}}{h} &= \tilde{k}_i \frac{du_i}{dr} \\
&- h \left[\frac{1}{2} \tilde{k}_i \frac{d^2 u_i}{dr^2} + \frac{1}{2} \frac{d\tilde{k}_i}{dr} \frac{du_i}{dr} \right] \\
&+ h^2 \left[\frac{1}{6} \tilde{k}_i \frac{d^3 u_i}{dr^3} + \frac{1}{4} \frac{d\tilde{k}_i}{dr} \frac{d^2 u_i}{dr^2} + \frac{1}{8} \frac{d^2 \tilde{k}_i}{dr^2} \frac{du_i}{dr} \right] \\
&- h^3 \left[\frac{1}{24} \tilde{k}_i \frac{d^4 u_i}{dr^4} + \frac{1}{12} \frac{d\tilde{k}_i}{dr} \frac{d^3 u_i}{dr^3} + \frac{1}{16} \frac{d^2 \tilde{k}_i}{dr^2} \frac{d^2 u_i}{dr^2} + \frac{1}{48} \frac{d^3 \tilde{k}_i}{dr^3} \frac{du_i}{dr} \right] \\
&+ \mathcal{O}(h^4)
\end{aligned}$$

Подставим это в уравнение невязки внутри интервала:

$$\begin{aligned}
\xi_i &= h \tilde{f}_i + \left[\tilde{k}_i \frac{du_i}{dr} + h \left[\frac{1}{2} \tilde{k}_i \frac{d^2 u_i}{dr^2} + \frac{1}{2} \frac{d\tilde{k}_i}{dr} \frac{du_i}{dr} \right] + h^2 \left[\frac{1}{6} \tilde{k}_i \frac{d^3 u_i}{dr^3} + \frac{1}{4} \frac{d\tilde{k}_i}{dr} \frac{d^2 u_i}{dr^2} + \frac{1}{8} \frac{d^2 \tilde{k}_i}{dr^2} \frac{du_i}{dr} \right] \right. \\
&+ h^3 \left[\frac{1}{24} \tilde{k}_i \frac{d^4 u_i}{dr^4} + \frac{1}{12} \frac{d\tilde{k}_i}{dr} \frac{d^3 u_i}{dr^3} + \frac{1}{16} \frac{d^2 \tilde{k}_i}{dr^2} \frac{d^2 u_i}{dr^2} + \frac{1}{48} \frac{d^3 \tilde{k}_i}{dr^3} \frac{du_i}{dr} \right] \\
&- \tilde{k}_i \frac{du_i}{dr} + h \left[\frac{1}{2} \tilde{k}_i \frac{d^2 u_i}{dr^2} + \frac{1}{2} \frac{d\tilde{k}_i}{dr} \frac{du_i}{dr} \right] \\
&- h^2 \left[\frac{1}{6} \tilde{k}_i \frac{d^3 u_i}{dr^3} + \frac{1}{4} \frac{d\tilde{k}_i}{dr} \frac{d^2 u_i}{dr^2} + \frac{1}{8} \frac{d^2 \tilde{k}_i}{dr^2} \frac{du_i}{dr} \right] \\
&\left. - h^3 \left[\frac{1}{24} \tilde{k}_i \frac{d^4 u_i}{dr^4} + \frac{1}{12} \frac{d\tilde{k}_i}{dr} \frac{d^3 u_i}{dr^3} + \frac{1}{16} \frac{d^2 \tilde{k}_i}{dr^2} \frac{d^2 u_i}{dr^2} + \frac{1}{48} \frac{d^3 \tilde{k}_i}{dr^3} \frac{du_i}{dr} \right] \right]
\end{aligned}$$

Приводим подобные слагаемые:

$$\begin{aligned}
\xi_i &= h \left[\tilde{f} + \tilde{k} \frac{d^2 u}{dr^2} \frac{d\tilde{k}}{dr} \frac{du}{dr} - \tilde{q}u \right] \\
&+ h^3 \left[\frac{1}{12} \tilde{k} \frac{d^4 u_i}{dr^4} + \frac{1}{6} \frac{d\tilde{k}}{dr} \frac{d^3 u_i}{dr^3} + \frac{1}{8} \frac{d^2 \tilde{k}}{dr^2} \frac{d^3 u_i}{dr^3} + \frac{1}{24} \frac{d^3 \tilde{k}}{dr^3} \frac{du_i}{dr} \right] + \mathcal{O}(h^4)
\end{aligned}$$

Можно заметить что:

$$\left[\tilde{k} \frac{d^2 u}{dr^2} + \frac{d\tilde{k}}{dr} \frac{du}{dr} \right] = \frac{d}{dr} \left(\tilde{k} \frac{du}{dr} \right)$$

При этом:

$$\tilde{f} + \frac{d}{dr} \left(\tilde{k} \frac{du}{dr} \right) - \tilde{q}u = 0$$

Тем самым получаем:

$$\xi_i = h^3 \left[\frac{1}{12} \tilde{k} \frac{d^4 u_i}{dr^4} + \frac{1}{6} \frac{d\tilde{k}}{dr} \frac{d^3 u_i}{dr^3} + \frac{1}{8} \frac{d^2 \tilde{k}}{dr^2} \frac{d^3 u_i}{dr^3} + \frac{1}{24} \frac{d^3 \tilde{k}}{dr^3} \frac{du_i}{dr} \right] + \mathcal{O}(h^4)$$

Найдем разложение невязки для граничного условия слева:

$$\xi_i = \frac{h}{2} \tilde{f}_i + \left[\tilde{k}_{i+0.5} \frac{u_{i+1} - u_i}{h} + \nu_1 - \frac{h}{2} \tilde{q}_i u_i \right] \quad i = 0$$

Получаем:

$$\begin{aligned} \xi_i = & \frac{h}{2} \tilde{f}_i + \tilde{k}_i \frac{du_i}{dr} + h \left[\frac{1}{2} \tilde{k}_i \frac{d^2 u_i}{dr^2} + \frac{1}{2} \frac{d\tilde{k}_i}{dr} \frac{du_i}{dr} \right] \\ & + h^2 \left[\frac{1}{6} \tilde{k}_i \frac{d^3 u_i}{dr^3} + \frac{1}{4} \frac{d\tilde{k}_i}{dr} \frac{d^2 u_i}{dr^2} + \frac{1}{8} \frac{d^2 \tilde{k}_i}{dr^2} \frac{du_i}{dr} \right] + \mathcal{O}(h^3) + \nu_1 - \frac{h}{2} \tilde{q}_i u_i \end{aligned}$$

Сгруппируем слагаемые при одинаковых степенях h :

$$\begin{aligned} \xi_i = & h^0 \left[\nu_1 + \tilde{k}_i \frac{du_i}{dr} \right] + \frac{h}{2} \left[\tilde{f}_i + \tilde{k}_i \frac{d^2 u_i}{dr^2} + \frac{d\tilde{k}_i}{dr} \frac{du_i}{dr} - \tilde{q}_i u_i \right] \\ & + h^2 \left[\frac{1}{6} \tilde{k}_i \frac{d^3 u_i}{dr^3} + \frac{1}{4} \frac{d\tilde{k}_i}{dr} \frac{d^2 u_i}{dr^2} + \frac{1}{8} \frac{d^2 \tilde{k}_i}{dr^2} \frac{du_i}{dr} \right] + \mathcal{O}(h^3) \end{aligned}$$

Из условия:

$$k \frac{du}{dr} \Big|_{r=R_L} = -\nu_1$$

Также:

$$\begin{aligned} \left[\tilde{k} \frac{d^2 u}{dr^2} + \frac{d\tilde{k}}{dr} \frac{du}{dr} \right] &= \frac{d}{dr} \left(\tilde{k} \frac{du}{dr} \right) \\ \tilde{f} + \frac{d}{dr} \left(\tilde{k} \frac{du}{dr} \right) - \tilde{q} u &= 0 \end{aligned}$$

Тем самым получаем:

$$\xi_i = h^2 \left[\frac{1}{6} \tilde{k}_i \frac{d^3 u_i}{dr^3} + \frac{1}{4} \frac{d\tilde{k}_i}{dr} \frac{d^2 u_i}{dr^2} + \frac{1}{8} \frac{d^2 \tilde{k}_i}{dr^2} \frac{du_i}{dr} \right] + \mathcal{O}(h^3)$$

Теперь найдем разложение невязки для правой границы

$$\xi_i = \frac{h}{2} \tilde{f}_i + \left[\nu_2 - \tilde{k}_{i-0.5} \frac{u_i - u_{i-1}}{h} - \frac{h}{2} \tilde{q}_i u_i \right], \quad i = N$$

Получаем:

$$\begin{aligned} \xi_i = & \frac{h}{2} \tilde{f}_i - \tilde{k}_i \frac{du_i}{dr} + h \left[\frac{1}{2} \tilde{k}_i \frac{d^2 u_i}{dr^2} + \frac{1}{2} \frac{d\tilde{k}_i}{dr} \frac{du_i}{dr} \right] \\ & - h^2 \left[\frac{1}{6} \tilde{k}_i \frac{d^3 u_i}{dr^3} + \frac{1}{4} \frac{d\tilde{k}_i}{dr} \frac{d^2 u_i}{dr^2} + \frac{1}{8} \frac{d^2 \tilde{k}_i}{dr^2} \frac{du_i}{dr} \right] + \mathcal{O}(h^3) + \nu_2 - \frac{h}{2} \tilde{q}_i u_i \end{aligned}$$

Сгруппируем слагаемые при одинаковых степенях h :

$$\begin{aligned}\xi_i = & h^0 \left[\nu_2 - \tilde{k}_i \frac{du_i}{dr} \right] + \frac{h}{2} \left[\tilde{f}_i + \tilde{k}_i \frac{d^2 u_i}{dr^2} + \frac{d\tilde{k}_i}{dr} \frac{du_i}{dr} - \tilde{q}_i u_i \right] \\ & - h^2 \left[\frac{1}{6} \tilde{k}_i \frac{d^3 u_i}{dr^3} + \frac{1}{4} \frac{d\tilde{k}_i}{dr} \frac{d^2 u_i}{dr^2} + \frac{1}{8} \frac{d^2 \tilde{k}_i}{dr^2} \frac{du_i}{dr} \right] + \mathcal{O}(h^3)\end{aligned}$$

Из условия:

$$-k \frac{du}{dr} \Big|_{r=R_R} = -\nu_2$$

Также:

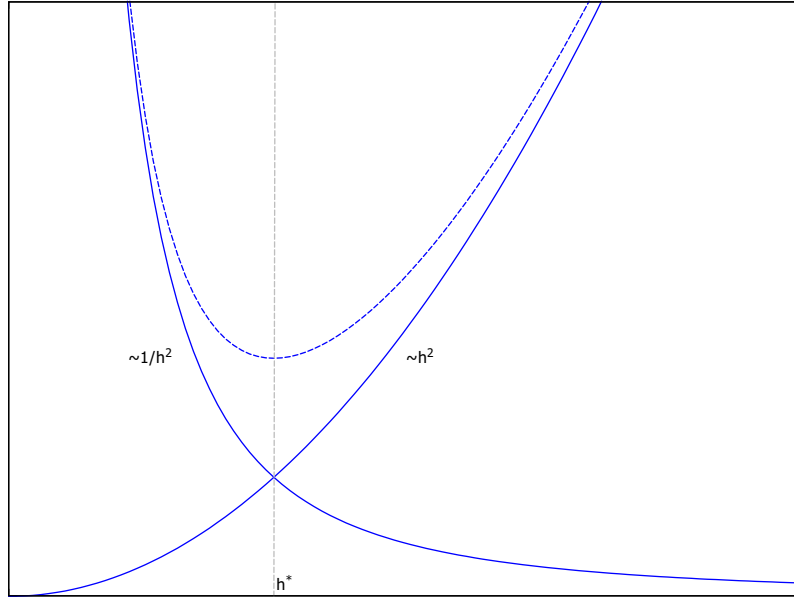
$$\begin{aligned}\left[\tilde{k} \frac{d^2 u}{dr^2} + \frac{d\tilde{k}}{dr} \frac{du}{dr} \right] &= \frac{d}{dr} \left(\tilde{k} \frac{du}{dr} \right) \\ \tilde{f} + \frac{d}{dr} \left(\tilde{k} \frac{du}{dr} \right) - \tilde{q}u &= 0\end{aligned}$$

Тем самым получаем:

$$\xi_i = -h^2 \left[\frac{1}{6} \tilde{k}_i \frac{d^3 u_i}{dr^3} + \frac{1}{4} \frac{d\tilde{k}_i}{dr} \frac{d^2 u_i}{dr^2} + \frac{1}{8} \frac{d^2 \tilde{k}_i}{dr^2} \frac{du_i}{dr} \right] + \mathcal{O}(h^3)$$

2.3.5 Зависимость погрешности от числа разбиений

Как было показано выше, невязка $\|\xi\| \sim h^2$, когда погрешность решения системы алгебраических уравнений $\|z\| \sim \frac{1}{h^2}$.



Значит существует такое число разбиений где погрешность минимальная.

2.4 Тестирование

2.4.1 Метод прогонки

Нижняя и верхняя диагональ заполняются случайными числами. Главная диагональ заполняется следующим образом:

$$c_i = |a_i| + |b_i| + 1$$

Главная диагональ заполняется таким образом для уменьшения возможности появления плохо обусловленной матрицы. Также случайными числами заполняется вектор решения x .

Чтобы получить вектор g , мы умножаем матрицу на вектор решения. После получения вектора g используем наш метод прогонки, чтобы получить вектор решения \tilde{x} . Для прохождения теста, полученные данные должны соответствовать следующему неравенству:

$$\|\tilde{x} - x\| < \text{cond}(A)\varepsilon_M \|x\|$$

Результат выполнения тестов:

```
● mushroom@mushroom-HP-Notebook:~/Documents/mm2/lab1/build$ ./tma_test --log_level=test_suite
Running 2 test cases...
Entering test module "TMA_TEST"
/home/mushroom/Documents/mm2/lab1/test/test_tma.cpp(6): Entering test case "tma_float"
/home/mushroom/Documents/mm2/lab1/test/test_tma.cpp(6): Leaving test case "tma_float"; testing time: 23315727us
/home/mushroom/Documents/mm2/lab1/test/test_tma.cpp(11): Entering test case "tma_double"
/home/mushroom/Documents/mm2/lab1/test/test_tma.cpp(11): Leaving test case "tma_double"; testing time: 23323196us
Leaving test module "TMA_TEST"; testing time: 46639058us

*** No errors detected
○ mushroom@mushroom-HP-Notebook:~/Documents/mm2/lab1/build$
```

2.4.2 Интегро-интерполяционный метод

Выполним нашу программу на ряде входных параметров

Номер теста	k	q	u	f	ν_1	ν_2	R_L	R_R
1	r	1	$2r + 3$	$2r - 1$	-2	4	1	2
2	r^2	1	$2r^2 + 3$	$-14r^2 + 3$	-4	32	1	2
3	r^3	1	$2r^3 + 3$	$-36r^4 + 2r^3 + 3$	-6	192	1	2

Результаты тестов:

N	$\ \varepsilon\ $, одинарная точность	$\ \varepsilon\ $, двойная точность
8	2.45905e-03	2.46952e-03
16	6.10828e-04	6.19519e-04
32	1.39713e-04	1.55015e-04
64	6.91414e-05	3.87621e-05
128	6.19888e-05	9.69106e-06
256	2.66743e-03	2.42279e-06
512	9.03511e-03	6.05706e-07
1024	9.06944e-03	1.51435e-07
2048	4.98871e-01	3.78822e-08

Таблица 1: Погрешность теста №1

N	$\ \varepsilon\ $, одинарная точность	$\ \varepsilon\ $, двойная точность
8	1.49273e-01	1.49265e-01
16	3.73564e-02	3.73383e-02
32	9.32503e-03	9.33596e-03
64	1.68228e-03	2.33408e-03
128	1.65176e-03	5.83525e-04
256	1.83487e-03	1.45882e-04
512	2.59638e-02	3.64704e-05
1024	7.11317e-01	9.11763e-06
2048	5.93484	2.27931e-06

Таблица 2: Погрешность теста №2

N	$\ \varepsilon\ $, одинарная точность	$\ \varepsilon\ $, двойная точность
8	2.52285	2.52289
16	6.31502e-01	6.31225e-01
32	1.5601e-01	1.57838e-01
64	4.27036e-02	3.94614e-02
128	1.83411e-02	9.86546e-03
256	7.73621e-03	2.46637e-03
512	2.19547e-01	6.16594e-04
1024	1.32325	1.54148e-04
2048	17.763	3.85369e-05

Таблица 3: Погрешность теста №3

Добавим еще один тест

$$k = \sin(r) \quad q = \cos(r) \quad u = \tan(r)$$

$$r \in [\pi; \frac{3 \cdot \pi}{2}]$$

Находим значение f как:

$$- \left[\frac{1}{r} \frac{d}{dr} \left(r \cdot \sin(r) \cdot \frac{d \tan(r)}{dr} \right) - \cos(r) \cdot \tan(r) \right] = f(r),$$

Получаем:

$$f(r) = \frac{\cos(r)^2}{r} + \sin(2 \cdot r) + \frac{\sin(r)^2}{\cos(r)}$$

Находим значения ν_1 и ν_2 :

$$\sin(r) \frac{d \tan(r)}{dr} \Big|_{r=\pi} = -\nu_1 \qquad -\sin(r) \frac{d \tan(r)}{dr} \Big|_{r=\frac{3 \cdot \pi}{2}} = -\nu_2$$

$$\nu_1 = 1$$

$$\nu_2 = 0$$

N	$\ \varepsilon\ $, одинарная точность	$\ \varepsilon\ $, двойная точность
8	1.1626e-02	1.16233e-02
16	4.26462e-03	4.26086e-03
32	9.20491e-02	9.30977e-02
64	1.34945e-04	1.13523e-04
128	5.553e-05	3.75747e-05
256	3.72678e-05	1.3292e-05
512	2.75075e-04	5.95552e-06
1024	1.2811e-03	8.8951e-07
2048	1.11288e-02	1.11339e-07

Таблица 4: Погрешность теста №4

3 Заключение

3.1 Вывод

Задание выполнено в полном объеме. Был написан метод приближенного вычисления краевой задачи и метод прогонки, а также написаны тесты к ним. Была оценена погрешность и выявлена зависимость погрешности от числа разбиений.

3.2 Код

```
1  #include <iostream>
2  #include <vector>
3  #include <iterator>
4  #include <cmath>
5  #include <algorithm>
6
7  #include "utils.hpp"
8  #include "tma.hpp"
9  #include "balance.hpp"
10 #include "data_table.hpp"
11
12 int main()
13 {
14     auto data_table = get_data< float >();
15     auto data_table2 = get_data< double >();
16
17     for (std::size_t N = 8; N <= 2048; N *= 2)
18     {
19         std::vector< float > first(N + 1);
20         std::vector< float > second(N + 1);
21         std::cout << "N = " << N << "\n";
22         std::cout << "float \n";
23         for (auto data : data_table)
24         {
25             balance_solve(N, data, first, second);
26
27             std::cout << "eps = " << eps(first, second) << "\n";
28         }
29
30         std::vector< double > first1(N + 1);
31         std::vector< double > second1(N + 1);
32         std::cout << "double \n";
33         for (auto data : data_table2)
34         {
35             balance_solve(N, data, first1, second1);
36
37             std::cout << "eps = " << eps(first1, second1) << "\n";
38         }
39     }
40
41     return 0;
42 }
```

```

1  #ifndef DATA_TABLE_HPP
2  #define DATA_TABLE_HPP
3
4  #include <cmath>
5  #include <vector>
6  #include "utils/data.hpp"
7
8  template <typename T = double>
9  std::vector< Data< T > > get_data()
10 {
11     std::vector< Data< T > > data_table =
12     {
13         {
14             1, 2,
15             [](T r) -> T { return 2 * r + 3; },
16             [](T r) -> T { return r; },
17             [](T r) -> T { return 1; },
18             [](T r) -> T { return 2 * r - 1; },
19             -2, 4
20         },
21         {
22             1, 2,
23             [](T r) -> T { return 2 * std::pow(r, 2) + 3; },
24             [](T r) -> T { return std::pow(r, 2); },
25             [](T r) -> T { return 1; },
26             [](T r) -> T { return -14 * std::pow(r, 2) + 3; },
27             -4, 32
28         },
29         {
30             1, 2,
31             [](T r) -> T { return 2 * std::pow(r, 3) + 3; },
32             [](T r) -> T { return std::pow(r, 3); },
33             [](T r) -> T { return 1; },
34             [](T r) -> T { return -36 * std::pow(r, 4) + 2 * std::pow(r, 3) + 3; },
35             -6, 192
36         },
37         {
38             M_PI, 3 * M_PI / 2,
39             [](T r) -> T { return std::sin(r); },
40             [](T r) -> T { return std::cos(r); },
41             [](T r) -> T { return std::tan(r); },
42             [](T r) -> T {
43                 return - std::pow(std::cos(r), 2) / r + sin(2 * r) + std::pow(std::sin(r), 2) / std::cos(r)
44             },
45             -1, 0
46         },

```

```

47     };
48
49     return data_table;
50 }
51
52 #endif

```

```

1  #ifndef TMA_HPP
2  #define TMA_HPP
3
4  #include <vector>
5  #include <cstdint>
6  #include <type_traits>
7  #include <cassert>
8
9  template <typename T = double>
10 void tma(
11     const std::vector< T >& a,
12     const std::vector< T >& c,
13     const std::vector< T >& b,
14     const std::vector< T >& r,
15     std::vector< T >& x
16 ) {
17     static_assert(std::is_floating_point< T >::value);
18
19     std::size_t sz = r.size();
20
21     assert(sz == b.size());
22     assert(sz == c.size());
23     assert(sz == x.size());
24     assert(sz == r.size());
25     assert(a[0] == 0);
26     assert(b[sz - 1] == 0);
27
28     std::vector< T > y(sz);
29     std::vector< T > p(sz);
30
31     y[0] = b[0] / c[0];
32     p[0] = r[0] / c[0];
33
34     for( size_t i = 1; i < sz; i++)
35     {
36         y[i] = b[i] / (c[i] - a[i] * y[i - 1]);
37         p[i] = ( r[i] - a[i] * p[i - 1] ) / ( c[i] - a[i] * y[i - 1] );
38     }
39

```

```

40     for (int i = sz - 1; i >= 0; i--)
41     {
42         x[i] = p[i] - y[i] * x[i + 1];
43     }
44 }
45
46 #endif

```

```

1  #ifndef BALANCE_HPP
2  #define BALANCE_HPP
3
4  #include <cstdint>
5  #include <vector>
6  #include <iterator>
7
8  #include "utils/data.hpp"
9  #include "utils/balance_utils.hpp"
10 #include "tma.hpp"
11
12 template <typename T = double>
13 void balance_solve(std::size_t N, const Data< T >& data,
14                   std::vector< T >& balance_result, std::vector< T >& tma_result)
15 {
16     static_assert(std::is_floating_point< T >::value);
17     assert(data.R_L < data.R_R);
18
19     std::vector< T > a(N + 1);
20     std::vector< T > c(N + 1);
21     std::vector< T > b(N + 1);
22     std::vector< T > r(N + 1);
23
24     init_balance(N, data, a, c, b, r, balance_result);
25
26     tma(a, c, b, r, tma_result);
27 };
28
29 #endif

```

```

1  #ifndef UTILS_HPP
2  #define UTILS_HPP
3
4  #include <vector>
5  #include <iomanip>
6  #include <cmath>

```

```

7
8  #include <boost/range/combine.hpp>
9
10 #include <gsl/gsl_matrix.h>
11 #include <gsl/gsl_linalg.h>
12
13 template< typename T >
14 T eps(const std::vector< T > &v1,
15       const std::vector< T > &v2)
16 {
17     T eps_ = 0;
18
19     for (auto &&tuple: boost::combine(v1, v2))
20     {
21         T x1, x2;
22         boost::tie(x1, x2) = tuple;
23
24         T del = std::fabs(x1 - x2);
25
26         if (del > eps_)
27         {
28             eps_ = del;
29         }
30     }
31
32     return eps_;
33 }
34
35 template <typename T>
36 T cond(
37     const std::vector< T >& a,
38     const std::vector< T >& c,
39     const std::vector< T >& b
40 )
41 {
42     std::size_t N = b.size() - 1;
43     double rez = 1;
44
45     gsl_matrix* m = gsl_matrix_alloc(N + 1, N + 1);
46     gsl_matrix* n = gsl_matrix_alloc(N + 1, N + 1);
47
48     gsl_matrix_set_zero(m);
49
50     gsl_matrix_set(m, 0, 0, c[0]);
51     gsl_matrix_set(m, 0, 1, b[0]);
52

```

```

53     for (std::size_t i = 1; i < N; i++)
54     {
55         gsl_matrix_set(m, i, i - 1, a[i]);
56         gsl_matrix_set(m, i, i, c[i]);
57         gsl_matrix_set(m, i, i + 1, b[i]);
58     }
59
60     gsl_matrix_set(m, N, N - 1, a[N]);
61     gsl_matrix_set(m, N, N, c[N]);
62
63     rez *= gsl_matrix_norm1(m);
64
65     int s;
66     gsl_permutation* p = gsl_permutation_alloc(c.size());
67
68     gsl_linalg_LU_decomp(m, p, &s);
69     gsl_linalg_LU_invert(m, p, n);
70
71     rez *= gsl_matrix_norm1(n);
72
73     gsl_permutation_free(p);
74     gsl_matrix_free(m);
75     gsl_matrix_free(n);
76
77     return rez;
78 }
79
80
81 #endif

```

```

1  #ifndef BALANCE_UTILS_HPP
2  #define BALANCE_UTILS_HPP
3
4  #include <vector>
5
6  #include "data.hpp"
7  #include "grid.hpp"
8
9  template <typename T = double>
10 void init_balance(
11     std::size_t N,
12     const Data< T >& data,
13     std::vector< T >& a,
14     std::vector< T >& c,
15     std::vector< T >& b,
16     std::vector< T >& r,

```

```

17     std::vector< T >& x
18 )
19 {
20     static_assert(std::is_floating_point< T >::value);
21
22     std::size_t sz = N + 1;
23     assert(a.size() == sz);
24     assert(c.size() == sz);
25     assert(b.size() == sz);
26     assert(r.size() == sz);
27
28     auto nu_1 = data.nu_1;
29     auto nu_2 = data.nu_2;
30
31     auto h_1 = h1< T >(N, data);
32     auto h_2 = h2< T >(N, data);
33
34     auto r_1 = r1< T >(N, data);
35     auto r_2 = r2< T >(N, data);
36
37     auto k_1 = k1< T >(N, data);
38     auto k_2 = k2< T >(N, data);
39
40     auto q_1 = q< T >(N, data);
41     auto f_1 = f< T >(N, data);
42
43     a[0] = 0;
44     c[0] = r_2(0) * k_2(0) / h_1(1) + h_2(0) * r_1(0) * q_1(0);
45     b[0] = -r_2(0) * k_2(0) / h_1(1);
46     r[0] = h_2(0) * r_1(0) * f_1(0) + r_1(0) * nu_1;
47     x[0] = data.u(r_1(0));
48
49     for (int i = 1; i < N; i++)
50     {
51         a[i] = -r_2(i - 1) * k_2(i - 1) / h_1(i);
52         c[i] = r_2(i - 1) * k_2(i - 1) / h_1(i)
53             + r_2(i) * k_2(i) / h_1(i + 1)
54             + h_2(i) * r_1(i) * q_1(i);
55         b[i] = -r_2(i) * k_2(i) / h_1(i + 1);
56         r[i] = h_2(i) * r_1(i) * f_1(i);
57         x[i] = data.u(r_1(i));
58     }
59
60     a[N] = -r_2(N - 1) * k_2(N - 1) / h_1(N);
61     c[N] = r_2(N - 1) * k_2(N - 1) / h_1(N)
62         + h_2(N) * r_1(N) * q_1(N);

```



```

63   r[N] = h_2(N) * r_1(N) * f_1(N) + r_1(N) * nu_2;
64   b[N] = 0;
65   x[N] = data.u(r_1(N));
66 }
67
68 #endif

```

```

1  #ifndef DATA_HPP
2  #define DATA_HPP
3
4  template< typename T = double >
5  using func_t = T (*)(T);
6
7  template <typename T = double>
8  struct Data
9  {
10     const T R_L;
11     const T R_R;
12     const func_t< T > u;
13     const func_t< T > k;
14     const func_t< T > q;
15     const func_t< T > f;
16     const T nu_1;
17     const T nu_2;
18 };
19
20 #endif

```

```

1  #ifndef GRID_HPP
2  #define GRID_HPP
3
4  #include <cstddef>
5  #include <iostream>
6  #include "data.hpp"
7
8  template <typename T = double>
9  class h1
10 {
11     public:
12     h1(std::size_t N, const Data< T >& data):
13         N_(N),
14         data_(data)
15     {}
16

```

```

17     T operator() (std::size_t i)
18     {
19         assert(i > 0);
20         assert(i <= N_);
21         auto R_L = data_.R_L;
22         auto R_R = data_.R_R;
23
24         return (R_R - R_L) / N_;
25     }
26
27     private:
28         std::size_t N_;
29         const Data< T >& data_;
30 };
31
32 template <typename T = double>
33 class h2
34 {
35     public:
36         h2(std::size_t N, const Data< T >& data):
37             N_(N),
38             data_(data)
39         {}
40
41         T operator() (std::size_t i)
42         {
43             assert(i >= 0);
44             assert(i <= N_);
45
46             auto h_1 = h1(N_, data_);
47
48             if (i == 0)
49             {
50                 return h_1(i + 1) / 2;
51             }
52
53             if (i == N_)
54             {
55                 return h_1(i) / 2;
56             }
57
58             return (h_1(i) + h_1(i + 1)) / 2;
59         }
60
61     private:
62         std::size_t N_;

```

```

63     const Data< T >& data_;
64 };
65
66 template <typename T = double>
67 class r1
68 {
69     public:
70         r1(std::size_t N, const Data<T>& data):
71             N_(N),
72             data_(data)
73         {}
74
75         T operator() (std::size_t i)
76         {
77             auto h_1 = h1(N_, data_);
78             auto R_L = data_.R_L;
79
80             auto r = R_L;
81
82             for (int j = 1; j <= i; j++)
83             {
84                 r += h_1(j);
85             }
86
87             return r;
88         }
89     private:
90         std::size_t N_;
91         const Data< T >& data_;
92 };
93
94
95 template <typename T = double>
96 class r2
97 {
98     public:
99         r2(std::size_t N, const Data< T >& data):
100             N_(N),
101             data_(data)
102         {}
103
104         T operator() (std::size_t i)
105         {
106             auto h_2 = h2(N_, data_);
107             auto R_L = data_.R_L;
108

```

```

109         auto r = R_L;
110
111         for (int j = 0; j <= i; j++)
112         {
113             r += h_2(j);
114         }
115
116         return r;
117     }
118 private:
119     std::size_t N_;
120     const Data< T >& data_;
121 };
122
123 template <typename T = double>
124 class k1
125 {
126 public:
127     k1(std::size_t N, const Data< T > data):
128         N_(N),
129         data_(data)
130     {}
131
132     T operator() (std::size_t i)
133     {
134         auto r_1 = r1(N_, data_);
135
136         auto r = r_1(i);
137         return data_.k(r);
138     }
139
140 private:
141     std::size_t N_;
142     const Data< T >& data_;
143 };
144
145 template <typename T = double>
146 class k2
147 {
148 public:
149     k2(std::size_t N, const Data< T >& data):
150         N_(N),
151         data_(data)
152     {}
153
154     T operator() (std::size_t i)

```

```

155     {
156         auto r_2 = r2(N_, data_);
157         auto r = r_2(i);
158
159         return data_.k(r);
160     }
161
162     private:
163         size_t N_;
164         const Data< T >& data_;
165 };
166
167 template <typename T = double>
168 class q
169 {
170     public:
171         q(std::size_t N, const Data< T >& data):
172             N_(N),
173             data_(data)
174         {}
175
176         T operator() (std::size_t i)
177         {
178             auto r_1 = r1(N_, data_);
179             auto r = r_1(i);
180
181             return data_.q(r);
182         }
183
184     private:
185         std::size_t N_;
186         const Data< T >& data_;
187 };
188
189 template <typename T = double>
190 class f
191 {
192     public:
193         f(std::size_t N, const Data< T >& data):
194             N_(N),
195             data_(data)
196         {}
197
198         T operator() (std::size_t i)
199         {
200             auto r_1 = r1(N_, data_);

```

```

201         auto r = r_1(i);
202
203         return data_.f(r);
204     }
205
206     private:
207         std::size_t N_;
208         const Data< T >& data_;
209 };
210
211 #endif

```

```

1  #define BOOST_TEST_MODULE TMA_TEST
2  #include <boost/test/unit_test.hpp>
3
4  #include "test_utils.hpp"
5
6  BOOST_AUTO_TEST_CASE( tma_float )
7  {
8      test_tma< float >();
9  }
10
11  BOOST_AUTO_TEST_CASE( tma_double )
12  {
13      test_tma< double >();
14  }

```

```

1  #ifndef TEST_UTILS_HPP
2  #define TEST_UTILS_HPP
3
4  #include <cstdint>
5  #include <random>
6  #include <functional>
7  #include <vector>
8  #include <type_traits>
9
10 #include <boost/test/unit_test.hpp>
11 #include <boost/test/tools/floating_point_comparison.hpp>
12 #include <boost/range/combine.hpp>
13 #include <boost/range/join.hpp>
14
15 #include "tma.hpp"
16 #include "utils.hpp"
17

```

```

18 constexpr std::size_t MATRIX_SIZE = 100;
19 constexpr std::size_t NUM_ITER = 10000;
20
21 template <typename T = double>
22 void init_test_data(
23     std::vector< T >& a,
24     std::vector< T >& c,
25     std::vector< T >& b,
26     std::vector< T >& r,
27     std::vector< T >& x
28 )
29 {
30     static_assert(std::is_floating_point< T >::value);
31
32     std::size_t sz = x.size();
33
34     assert(sz == a.size());
35     assert(sz == c.size());
36     assert(sz == b.size());
37     assert(sz == r.size());
38
39     std::random_device rd;
40     std::mt19937 eng(rd());
41     std::uniform_real_distribution dist(-10.0, 10.0);
42     auto gen = std::bind(dist, eng);
43
44     auto j_range = boost::join(
45         boost::join(
46             boost::make_iterator_range(a.begin(), a.end()),
47             boost::make_iterator_range(b.begin(), b.end())
48         ),
49         boost::make_iterator_range(x.begin(), x.end())
50     );
51
52     std::generate(j_range.begin(), j_range.end(), gen);
53
54     auto range = boost::combine(a, b);
55     std::transform(range.begin(), range.end(),
56                   c.begin(),
57                   [](auto&& tuple) {
58                       auto x = tuple.template get<0>();
59                       auto y = tuple.template get<1>();
60                       return std::fabs(x) + std::fabs(y) + 1;
61                   });
62
63     a[0] = 0;

```

```

64     b[sz - 1] = 0;
65
66     r[0] = c[0] * x[0] + b[0] * x[1];
67
68     for (std::size_t i = 1; i < sz - 1; i++)
69     {
70         r[i] = a[i] * x[i - 1] + c[i] * x[i] + b[i] * x[i + 1];
71     }
72
73     r[sz - 1] = a[sz - 1] * x[sz - 2] + c[sz - 1] * x[sz - 1];
74 }
75
76 template <typename T>
77 void test_tma()
78 {
79     std::size_t sz = MATRIX_SIZE;
80     for (std::size_t i = 0; i < NUM_ITER; i++)
81     {
82         std::vector< T > a(sz);
83         std::vector< T > c(sz);
84         std::vector< T > b(sz);
85         std::vector< T > r(sz);
86         std::vector< T > x(sz);
87
88         std::vector< T > result(sz);
89         init_test_data(a, c, b, r, x);
90
91         tma(a, c, b, r, result);
92
93         double cond_num = cond(a, c, b);
94         double eps = std::numeric_limits< T >::epsilon();
95
96         double norm1 = 0;
97         double norm2 = 0;
98
99         for (auto&& tuple : boost::combine(x, result))
100         {
101             T y1, y2;
102             boost::tie(y1, y2) = tuple;
103
104             double v1 = std::fabs(y1 - y2);
105             double v2 = std::fabs(y1);
106
107             norm1 = (v1 > norm1 ? v1 : norm1);
108             norm2 = (v2 > norm2 ? v2 : norm2);
109         }

```



```
110
111     BOOST_REQUIRE_SMALL(norm1, cond_num * eps * norm2);
112 }
113 }
114
115 #endif
```
