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. 3530904/01004

Проверил: Дед Пихто

Санкт-Петербург
2023

Содержание

1 Вступление

1.1 Постановка задачи

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

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

Граничные условия:

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

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.

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

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

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

Введем дополнительную сетку:

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

Проведем аппроксимацию начального уравнения.
Для i = 0

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

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

Формула центральных разностей:

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

Граничное условие:

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

Формула левых прямоугольников:

$$\int_{r_i}^{r_{r+0.5}} r\varphi_i\, dr = \hbar_i r_i \varphi_i$$

Получаем разностную схему для i = 0:

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

Для i = 1, 2, …, N-1

$$-\int_{r_{i-0,5}}^{r_{i+0.5}} \left[\frac{d}{dr}\left(rk(r)\frac{du(r)}{dr}\right) - rq(r)u(r)\right] dr = \int_{r_{i-0.5}}^{r_{i+0.5}} rf(r)\, dr,$$

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

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

$$\int_{r_{i-0.5}}^{r_{r+0.5}} r\varphi_i\,dr \ = \ \hbar_i r_i \varphi_i$$

Получаем разностную схему для i = 1, 2, . . . , N-1:

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

Для i = N:

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

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

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

$$-k(r)\left.\frac{du(r)}{dr}\right|_{r=R_R} \ = \ -\nu_2$$

$$\int_{r_{i-0.5}}^{r_i} r\varphi(r)\,dr \approx \hbar_i r_i \varphi_i$$

Получаем разностную схему для i=N:

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

Сгрупиируем полученные уравнения:

$$-\left[r_{i+0.5}\cdot k_{i+0.5}\frac{v_{i+1}-v_i}{h_{i+1}} - r_i\cdot(-\nu_1) - \hbar r_i q_i v_i \right] \ = \ \hbar_i r_i f_i \quad i=0$$

$$-\left[r_{i+0.5}\cdot k_{i+0.5}\frac{v_{i+1}-v_i}{h_{i+1}} - r_{i-0.5}k_{i-0.5}\frac{v_i \ - \ v_{i-1}}{h_i} - \hbar r_i q_i v_i \right] \ = \ \hbar_i r_i f_i \quad i=1,2,...,N-1$$

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

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

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

Для i = 1, 2, ..., 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:

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

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

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

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

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

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

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

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

Строка N.

$$a_n x_{n-1} + c_n x_n = g_n$$
$$x_n = \rho_n, \;\; \rho_n = \frac{r_n - a_n\rho_{n-1}}{c_n - a_n\gamma_{n-1}}$$

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

Этап 2

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

$$x_n = \rho_n$$
$$x_i = \rho_i - \gamma_i x_{i+1}, \;\; i = n-1, n-2, \ldots, 1$$

2.3 Оценка погрешности

2.3.1 Невязка разностной схемы

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

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 $\|\zeta\|$:

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

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

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

Для $\|z\|$:

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

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

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

Подставим в наше неравенство, тем самым получаем:

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

2.3.3 Вклад от погрешности решения системы алгебраических уравнений

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

Знаем что:

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

Из этого получаем:

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

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

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

$$\|z\| \leq cond(A)\varepsilon_M \|v\|$$

2.3.4 Разложение невязки

Разностная схема

$$-\left[\frac{1}{r}\frac{d}{dr}\left(rk\frac{du}{dr}\right) \; - \; qu(r)\right] = f_i$$

$$-\left[\frac{d}{dr}\left(rk\frac{du}{dr}\right) \; - \; rqu(r)\right] = rf_i$$

Введем новые обозначения:

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

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:

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

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

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

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:

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

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

$$u_i = u(x_i+h) = u_i + h\frac{du_i}{dr} + \frac{h^2}{2}\frac{d^2u_i}{dr^2} + \frac{h^3}{6}\frac{d^4u_i}{dr^3} + \frac{h^4}{24}\frac{d^4u_i}{dr^4} + \mathcal{O}(h^5)$$

$$\frac{u_{i+1}-u_i}{h} = \frac{du_i}{dr} + \frac{h}{2}\frac{d^2u_i}{dr^2} + \frac{h^2}{6}\frac{d^3u_i}{dr^3} + \frac{h^3}{24}\frac{d^4u_i}{dr^4} + \mathcal{O}(h^4)$$

$$\tilde{k}_{i+\frac{1}{2}} = \tilde{k}(x_i+\frac{h}{2}) = \tilde{k}_i + \frac{h^2}{2}\frac{d\tilde{k}_i}{dr} + \frac{h^2}{8}\frac{d^2\tilde{k}_i}{dr^2} + \frac{h^3}{48}\frac{d^3\tilde{k}_i}{dr^3} + \mathcal{O}(h^3)$$

Получаем:

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

$$u_{i-1} = u(x_i-h) = u_i - h\frac{du_i}{dr} + \frac{h^2}{2}\frac{d^2u_i}{dr^2} - \frac{h^3}{6}\frac{d^4u_i}{dr^3} + \frac{h^4}{24}\frac{d^4u_i}{dr^4} + \mathcal{O}(h^5)$$

$$\frac{u_i-u_{i-1}}{h} = \frac{du_i}{dr} - \frac{h}{2}\frac{d^2u_i}{dr^2} + \frac{h^2}{6}\frac{d^3u_i}{dr^3} - \frac{h^3}{24}\frac{d^4u_i}{dr^4} + \mathcal{O}(h^4)$$

$$\tilde{k}_{i-\frac{1}{2}} = \tilde{k}(x_i-\frac{h}{2}) = \tilde{k}_i - \frac{h^2}{2}\frac{d\tilde{k}_i}{dr} + \frac{h^2}{8}\frac{d^2\tilde{k}_i}{dr^2} - \frac{h^3}{48}\frac{d^3\tilde{k}_i}{dr^3} + \mathcal{O}(h^3)$$

Получаем:

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

Подставим это в уравнение невязки внутри интервала:

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

Приводим подобные слагаемые:

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

Можно заметить что:

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

При этом:

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

Тем самым получаем:

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

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:

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

Сгруппируем слагаемые при одинаковых степенях h:

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

Из условия:

$$k\left.\frac{du}{dr}\right|_{r=R_L} = -\nu_1$$

Также:

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

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

Тем самым получаем:

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

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:

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

Сгруппируем слагаемые при одинаковых степенях h:

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

Из условия:

$$-k\left.\frac{du}{dr}\right|_{r=R_R} = -\nu_2$$

Также:

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

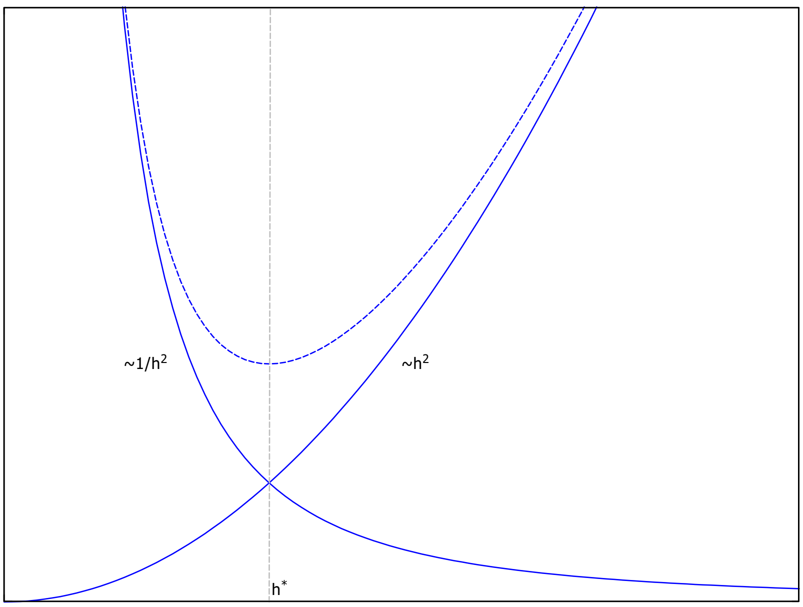
$$\tilde{f} + \frac{d}{dr}\left(\tilde{k}\frac{du}{dr}\right) - \tilde{q}u = 0$$

Тем самым получаем:

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

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:

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

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:

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

$\nu_1 = 1$

$\nu_2 = 0$

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 написан метод приближенного вычисления краевой задачи и метод прогонки, а также написаны тесты к ним. Была оценена погрешность и выявлена зависимость погрешности от числа разбиения.

3.2 Код

```cpp
#include <iostream>
#include <vector>
#include <iterator>
#include <cmath>
#include <algorithm>

#include "utils.hpp"
#include "tma.hpp"
#include "balance.hpp"
#include "data_table.hpp"

int main()
{
  auto data_table = get_data< float >();
  auto data_table2 = get_data< double >();

  for (std::size_t N = 8; N <= 2048; N *= 2)
  {
    std::vector< float > first(N + 1);
    std::vector< float > second(N + 1);
    std::cout << "N = " << N << "\n";
    std::cout << "float \n";
    for (auto data : data_table)
    {
      balance_solve(N, data, first, second);

      std::cout << "eps = " << eps(first, second) << "\n";
    }

    std::vector< double > first1(N + 1);
    std::vector< double > second1(N + 1);
    std::cout << "double \n";
    for (auto data : data_table2)
    {
      balance_solve(N, data, first1, second1);

      std::cout << "eps = " << eps(first1, second1) << "\n";
    }
  }

  return 0;
}
```

```cpp
#ifndef DATA_TABLE_HPP
#define DATA_TABLE_HPP

#include <cmath>
#include <vector>
#include "utils/data.hpp"

template <typename T = double>
std::vector< Data< T > > get_data()
{
  std::vector< Data< T > > data_table =
  {
    {
      1, 2,
      [](T r) -> T { return 2 * r + 3; },
      [](T r) -> T { return r; },
      [](T r) -> T { return 1; },
      [](T r) -> T { return 2 * r - 1; },
      -2, 4
    },
    {
      1, 2,
      [](T r) -> T { return 2 * std::pow(r, 2) + 3; },
      [](T r) -> T { return std::pow(r, 2); },
      [](T r) -> T { return 1; },
      [](T r) -> T { return -14 * std::pow(r, 2) + 3; },
      -4, 32
    },
    {
      1, 2,
      [](T r) -> T { return 2 * std::pow(r, 3) + 3; },
      [](T r) -> T { return std::pow(r, 3); },
      [](T r) -> T { return 1; },
      [](T r) -> T { return -36 * std::pow(r, 4) + 2 * std::pow(r, 3) + 3; },
      -6, 192
    },
    {
      M_PI, 3 * M_PI / 2,
      [](T r) -> T { return std::sin(r); },
      [](T r) -> T { return std::cos(r); },
      [](T r) -> T { return std::tan(r); },
      [](T r) -> T {
        return - std::pow(std::cos(r), 2) / r + sin(2 * r) + std::pow(std::sin(r), 2) / std::cos(r)
      },
      -1, 0
    },
```

```
  };

  return data_table;
}

#endif
```

```
#ifndef TMA_HPP
#define TMA_HPP

#include <vector>
#include <cstddef>
#include <type_traits>
#include <cassert>

template <typename T = double>
void tma(
  const std::vector< T >& a,
  const std::vector< T >& c,
  const std::vector< T >& b,
  const std::vector< T >& r,
        std::vector< T >& x
) {
  static_assert(std::is_floating_point< T >::value);

  std::size_t sz = r.size();

  assert(sz == b.size());
  assert(sz == c.size());
  assert(sz == x.size());
  assert(sz == r.size());
  assert(a[0] == 0);
  assert(b[sz - 1] == 0);

  std::vector< T > y(sz);
  std::vector< T > p(sz);

  y[0] = b[0] / c[0];
  p[0] = r[0] / c[0];

  for( size_t i = 1; i < sz; i++)
  {
    y[i] = b[i] / (c[i] - a[i] * y[i - 1]);
    p[i] = ( r[i] - a[i] * p[i - 1] ) / ( c[i] - a[i] * y[i - 1] );
  }

```

```
  for (int i = sz - 1; i >= 0; i--)
  {
    x[i] = p[i] - y[i] * x[i + 1];
  }
}

#endif
```

```
#ifndef BALANCE_HPP
#define BALANCE_HPP

#include <cstddef>
#include <vector>
#include <iterator>

#include "utils/data.hpp"
#include "utils/balance_utils.hpp"
#include "tma.hpp"

template <typename T = double>
void balance_solve(std::size_t N, const Data< T >& data,
                   std::vector< T >& balance_result, std::vector< T >& tma_result)
{
  static_assert(std::is_floating_point< T >::value);
  assert(data.R_L < data.R_R);

  std::vector< T > a(N + 1);
  std::vector< T > c(N + 1);
  std::vector< T > b(N + 1);
  std::vector< T > r(N + 1);

  init_balance(N, data, a, c, b, r, balance_result);

  tma(a, c, b, r, tma_result);
};

#endif
```

```
#ifndef UTILS_HPP
#define UTILS_HPP

#include <vector>
#include <iomanip>
#include <cmath>
```

```
#include <boost/range/combine.hpp>

#include <gsl/gsl_matrix.h>
#include <gsl/gsl_linalg.h>

template< typename T >
T eps(const std::vector< T > &v1,
             const std::vector< T > &v2)
{
  T eps_ = 0;

  for (auto &&tuple: boost::combine(v1, v2))
  {
  T x1, x2;
  boost::tie(x1, x2) = tuple;

  T del = std::fabs(x1 - x2);

    if (del > eps_)
    {
      eps_ = del;
    }
  }

  return eps_;
}

template <typename T>
T cond(
  const std::vector< T >& a,
  const std::vector< T >& c,
  const std::vector< T >& b
)
{
  std::size_t N = b.size() - 1;
  double rez = 1;

  gsl_matrix* m = gsl_matrix_alloc(N + 1, N + 1);
  gsl_matrix* n = gsl_matrix_alloc(N + 1, N + 1);

  gsl_matrix_set_zero(m);

  gsl_matrix_set(m, 0, 0, c[0]);
  gsl_matrix_set(m, 0, 1, b[0]);

```

```
  for (std::size_t i = 1; i < N; i++)
  {
    gsl_matrix_set(m, i, i - 1, a[i]);
    gsl_matrix_set(m, i, i, c[i]);
    gsl_matrix_set(m, i, i + 1, b[i]);
  }

  gsl_matrix_set(m, N, N - 1, a[N]);
  gsl_matrix_set(m, N, N, c[N]);

  rez *= gsl_matrix_norm1(m);

  int s;
  gsl_permutation* p = gsl_permutation_alloc(c.size());

  gsl_linalg_LU_decomp(m, p, &s);
  gsl_linalg_LU_invert(m, p, n);

  rez *= gsl_matrix_norm1(n);

  gsl_permutation_free(p);
  gsl_matrix_free(m);
  gsl_matrix_free(n);

  return rez;
}


#endif
```

```
#ifndef BALANCE_UTILS_HPP
#define BALANCE_UTILS_HPP

#include <vector>

#include "data.hpp"
#include "grid.hpp"

template <typename T = double>
void init_balance(
  std::size_t N,
  const Data< T >& data,
  std::vector< T >& a,
  std::vector< T >& c,
  std::vector< T >& b,
  std::vector< T >& r,
```

```
  std::vector< T >& x
)
{
  static_assert(std::is_floating_point< T >::value);

  std::size_t sz = N + 1;
  assert(a.size() == sz);
  assert(c.size() == sz);
  assert(b.size() == sz);
  assert(r.size() == sz);

  auto nu_1 = data.nu_1;
  auto nu_2 = data.nu_2;

  auto h_1 = h1< T >(N, data);
  auto h_2 = h2< T >(N, data);

  auto r_1 = r1< T >(N, data);
  auto r_2 = r2< T >(N, data);

  auto k_1 = k1< T >(N, data);
  auto k_2 = k2< T >(N, data);

  auto q_1 = q< T >(N, data);
  auto f_1 = f< T >(N, data);

  a[0] = 0;
  c[0] = r_2(0) * k_2(0) / h_1(1) + h_2(0) * r_1(0) * q_1(0);
  b[0] = -r_2(0) * k_2(0) / h_1(1);
  r[0] = h_2(0) * r_1(0) * f_1(0) + r_1(0) * nu_1;
  x[0] = data.u(r_1(0));

  for (int i = 1; i < N; i++)
  {
    a[i] = -r_2(i - 1) * k_2(i - 1) / h_1(i);
    c[i] = r_2(i - 1) * k_2(i - 1) / h_1(i)
          + r_2(i) * k_2(i) / h_1(i + 1)
          + h_2(i) * r_1(i) * q_1(i);
    b[i] = -r_2(i) * k_2(i) / h_1(i + 1);
    r[i] = h_2(i) * r_1(i) * f_1(i);
    x[i] = data.u(r_1(i));
  }

  a[N] = -r_2(N - 1) * k_2(N - 1) / h_1(N);
  c[N] = r_2(N - 1) * k_2(N - 1) / h_1(N)
        + h_2(N) * r_1(N) * q_1(N);
```

```
  r[N] = h_2(N) * r_1(N) * f_1(N) + r_1(N) * nu_2;
  b[N] = 0;
  x[N] = data.u(r_1(N));
}

#endif
```

```
#ifndef DATA_HPP
#define DATA_HPP

template< typename T = double >
using func_t = T (*)(T);

template <typename T = double>
struct Data
{
  const T R_L;
  const T R_R;
  const func_t< T > u;
  const func_t< T > k;
  const func_t< T > q;
  const func_t< T > f;
  const T nu_1;
  const T nu_2;
};

#endif
```

```
#ifndef GRID_HPP
#define GRID_HPP

#include <cstddef>
#include <iostream>
#include "data.hpp"

template <typename T = double>
class h1
{
  public:
    h1(std::size_t N, const Data< T >& data):
      N_(N),
      data_(data)
    {}

```

```
    T operator() (std::size_t i)
    {
      assert(i > 0);
      assert(i <= N_);
      auto R_L = data_.R_L;
      auto R_R = data_.R_R;

      return (R_R - R_L) / N_;
    }

  private:
    std::size_t N_;
    const Data< T >& data_;
};

template <typename T = double>
class h2
{
  public:
    h2(std::size_t N, const Data< T >& data):
      N_(N),
      data_(data)
    {}

    T operator() (std::size_t i)
    {
      assert(i >= 0);
      assert(i <= N_);

      auto h_1 = h1(N_, data_);

      if (i == 0)
      {
        return h_1(i + 1) / 2;
      }

      if (i == N_)
      {
        return h_1(i) / 2;
      }

      return (h_1(i) + h_1(i + 1)) / 2;
    }

  private:
    std::size_t N_;
```

```
    const Data< T >& data_;
};

template <typename T = double>
class r1
{
  public:
    r1(std::size_t N, const Data<T>& data):
      N_(N),
      data_(data)
    {}

    T operator() (std::size_t i)
    {
      auto h_1 = h1(N_, data_);
      auto R_L = data_.R_L;

      auto r = R_L;

      for (int j = 1; j <= i; j++)
      {
        r += h_1(j);
      }

      return r;
    }
  private:
    std::size_t N_;
    const Data< T >& data_;
};


template <typename T = double>
class r2
{
  public:
    r2(std::size_t N, const Data< T >& data):
      N_(N),
      data_(data)
    {}

    T operator() (std::size_t i)
    {
      auto h_2 = h2(N_, data_);
      auto R_L = data_.R_L;

```

```
      auto r = R_L;

      for (int j = 0; j <= i; j++)
      {
        r += h_2(j);
      }

      return r;
    }
  private:
    std::size_t N_;
    const Data< T >& data_;
};

template <typename T = double>
class k1
{
  public:
    k1(std::size_t N, const Data< T > data):
      N_(N),
      data_(data)
    {}

    T operator() (std::size_t i)
    {
      auto r_1 = r1(N_, data_);

      auto r = r_1(i);
      return data_.k(r);
    }

  private:
    std::size_t N_;
    const Data< T >& data_;
};

template <typename T = double>
class k2
{
  public:
    k2(std::size_t N, const Data< T >& data):
      N_(N),
      data_(data)
    {}

    T operator() (std::size_t i)
```

```
    {
      auto r_2 = r2(N_, data_);
      auto r = r_2(i);

      return data_.k(r);
    }

  private:
    size_t N_;
    const Data< T >& data_;
};

template <typename T = double>
class q
{
  public:
    q(std::size_t N, const Data< T >& data):
      N_(N),
      data_(data)
    {}

    T operator() (std::size_t i)
    {
      auto r_1 = r1(N_, data_);
      auto r = r_1(i);

      return data_.q(r);
    }

  private:
    std::size_t N_;
    const Data< T >& data_;
};

template <typename T = double>
class f
{
  public:
    f(std::size_t N, const Data< T >& data):
      N_(N),
      data_(data)
    {}

    T operator() (std::size_t i)
    {
      auto r_1 = r1(N_, data_);
```

```
      auto r = r_1(i);

      return data_.f(r);
    }

  private:
    std::size_t N_;
    const Data< T >& data_;
};

#endif
```

```
#define BOOST_TEST_MODULE TMA_TEST
#include <boost/test/unit_test.hpp>

#include "test_utils.hpp"

BOOST_AUTO_TEST_CASE( tma_float )
{
  test_tma< float >();
}

BOOST_AUTO_TEST_CASE( tma_double )
{
  test_tma< double >();
}
```

```
#ifndef TEST_UTILS_HPP
#define TEST_UTILS_HPP

#include <cstddef>
#include <random>
#include <functional>
#include <vector>
#include <type_traits>

#include <boost/test/unit_test.hpp>
#include <boost/test/tools/floating_point_comparison.hpp>
#include <boost/range/combine.hpp>
#include <boost/range/join.hpp>

#include "tma.hpp"
#include "utils.hpp"

```

```
constexpr std::size_t MATRIX_SIZE = 100;
constexpr std::size_t NUM_ITER = 10000;

template <typename T = double>
void init_test_data(
  std::vector< T >& a,
  std::vector< T >& c,
  std::vector< T >& b,
  std::vector< T >& r,
  std::vector< T >& x
)
{
  static_assert(std::is_floating_point< T >::value);

  std::size_t sz = x.size();

  assert(sz == a.size());
  assert(sz == c.size());
  assert(sz == b.size());
  assert(sz == r.size());

  std::random_device rd;
  std::mt19937 eng(rd());
  std::uniform_real_distribution dist(-10.0, 10.0);
  auto gen = std::bind(dist, eng);

  auto j_range = boost::join(
    boost::join(
      boost::make_iterator_range(a.begin(), a.end()),
      boost::make_iterator_range(b.begin(), b.end())
    ),
    boost::make_iterator_range(x.begin(), x.end())
  );

  std::generate(j_range.begin(), j_range.end(), gen);

  auto range = boost::combine(a, b);
  std::transform(range.begin(), range.end(),
                 c.begin(),
                 [](auto&& tuple) {
                  auto x = tuple.template get<0>();
                  auto y = tuple.template get<1>();
                  return std::fabs(x) + std::fabs(y) + 1;
                 });

  a[0] = 0;
```

```
  b[sz - 1] = 0;

  r[0] = c[0] * x[0] + b[0] * x[1];

  for (std::size_t i = 1; i < sz - 1; i++)
  {
    r[i] = a[i] * x[i - 1] + c[i] * x[i] + b[i] * x[i + 1];
  }

  r[sz - 1] = a[sz - 1] * x[sz - 2] + c[sz - 1] * x[sz - 1];
}

template <typename T>
void test_tma()
{
  std::size_t sz = MATRIX_SIZE;
  for (std::size_t i = 0; i < NUM_ITER; i++)
  {
    std::vector< T > a(sz);
    std::vector< T > c(sz);
    std::vector< T > b(sz);
    std::vector< T > r(sz);
    std::vector< T > x(sz);

    std::vector< T > result(sz);
    init_test_data(a, c, b, r, x);

    tma(a, c, b, r, result);

    double cond_num = cond(a, c, b);
    double eps =  std::numeric_limits< T >::epsilon();

    double norm1 = 0;
    double norm2 = 0;

    for (auto&& tuple : boost::combine(x, result))
    {
      T y1, y2;
      boost::tie(y1, y2) = tuple;

      double v1 = std::fabs(y1 - y2);
      double v2 = std::fabs(y1);

      norm1 = (v1 > norm1 ? v1 : norm1);
      norm2 = (v2 > norm2 ? v2 : norm2);
    }
```

```
    BOOST_REQUIRE_SMALL(norm1, cond_num * eps * norm2);
  }
}

#endif
```