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

Лабораторная работа № 2

по дисциплине
«Математические модели»

Выполнил: Ферапонтов М.В.
Группа: гр. 3530904/00104

Проверил: Воскобойников С. П.

Санкт-Петербург
2023

Содержание

1 Вступление

1.1 Постановка задачи

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

$$\frac{\partial u}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(rk(r,t)\frac{\partial u}{\partial r}\right) - q(r,t)u + f(r,t),\ r \in [R_L,\ R_R]\,,\ t \in [0,T],$$
$$0 < c_1 \leq k(r,t) \leq c_2,\ 0 \leq q(r,t)$$

Начальное условие:

$$u|_{t=0} = \varphi(r)$$

Граничные условия:

$$k\left.\frac{\partial u}{\partial r}\right|_{r=R_L} = -\nu_1(t) \qquad -k\left.\frac{\partial u}{\partial r}\right|_{r=R_R} = -\nu_2(t)$$

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 Разностная схема

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:

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

Напишем наше уравнение:

$$\frac{\partial u}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(rk(r,t)\frac{\partial u}{\partial r}\right) - q(r,t)u + f(r,t)$$

$$r \cdot \frac{\partial u}{\partial t} = \frac{\partial}{\partial r}\left(rk\frac{\partial u}{\partial r}\right) - r \cdot qu + r \cdot f$$

Проинтегрируем наше уравнение:

$$\int\limits_{r_{i-\frac{1}{2}}}^{r_{i+\frac{1}{2}}} r\frac{\partial u}{\partial t}dt = \int\limits_{r_{i-\frac{1}{2}}}^{r_{i+\frac{1}{2}}} \frac{\partial}{\partial r}\left(rk\frac{\partial u}{\partial r}\right)dr - \int\limits_{r_{i-\frac{1}{2}}}^{r_{i+\frac{1}{2}}} rqudr - \int\limits_{r_{i-\frac{1}{2}}}^{r_{i+\frac{1}{2}}} rfdr$$

Используем формулу левых прямоугольников:

$$\int\limits_{r_{i-\frac{1}{2}}}^{r_{i+\frac{1}{2}}} \varphi(x)dx \approx \hbar_i\varphi(x)$$

$$\hbar_i r_i \frac{dv_i}{dt} = rk\frac{\partial u}{\partial r}\bigg|_{r=r_{i+\frac{1}{2}}} - rk\frac{\partial u}{\partial r}\bigg|_{r=r_{i-\frac{1}{2}}} - \hbar_i r_i q v_i + \hbar_i r_i f_i$$

Используем формулы численного дифференцирования:

$$rk\frac{\partial u}{\partial r}\bigg|_{r=r_{i+\frac{1}{2}}} = r_{i+\frac{1}{2}}k_{i+\frac{1}{2}}\frac{v_{i+1}-v_i}{h_{i+1}}$$

$$rk\frac{\partial u}{\partial r}\bigg|_{r=r_{i-\frac{1}{2}}} == r_{i-\frac{1}{2}}k_{i-\frac{1}{2}}\frac{v_i-v_{i-1}}{h_i}$$

Тем самым мы получили нашу разностную схему внутри промежутка:

$$\hbar_i r_i \frac{dv_i}{dt} = r_{i+\frac{1}{2}} k_{i+\frac{1}{2}} \frac{v_{i+1} - v_i}{h_{i+1}} - r_{i-\frac{1}{2}} k_{i-\frac{1}{2}} \frac{v_i - v_{i-1}}{h_i} - \hbar_i r_i q v_i + \hbar_i r_i f_i, \ i = 1, 2, \ldots, N-1$$

$$\frac{dv_i}{dt} = \frac{r_{i+\frac{1}{2}} k_{i+\frac{1}{2}}}{\hbar_i r_i h_{i+1}} (v_{i+1} - v_i) - \frac{r_{i-\frac{1}{2}} k_{i-\frac{1}{2}}}{\hbar_i r_i h_i} (v_i - v_{i-1}) - q v_i + f_i, \ i = 1, 2, \ldots, N-1$$

Проведем аппроксимацию граничного условия слева:

$$\int\limits_{r_i}^{r_{i+\frac{1}{2}}} r \frac{\partial u}{\partial t} dt = \int\limits_{r_i}^{r_{i+\frac{1}{2}}} \frac{\partial}{\partial r} \left(r k \frac{\partial u}{\partial r} \right) dr - \int\limits_{r_i}^{r_{i+\frac{1}{2}}} r q u dr - \int\limits_{r_i}^{r_{i+\frac{1}{2}}} r f dr, \ i = 0$$

$$\hbar_i r_i \frac{dv_i}{dt} = r k \frac{\partial u}{\partial r} \bigg|_{r = r_{i+\frac{1}{2}}} - r k \frac{\partial u}{\partial r} \bigg|_{r = r_i} - \hbar_i r_i q v_i + \hbar_i r_i f_i, \ i = 0$$

Граничное условие

$$k \left. \frac{\partial u}{\partial r} \right|_{r = R_L} = -\nu_1(t)$$

Получаем:

$$\hbar_i r_i \frac{dv_i}{dt} = r_{i+\frac{1}{2}} k_{i+\frac{1}{2}} \frac{v_{i+1} - v_i}{h_{i+1}} + r_i \cdot \nu_1 - \hbar_i r_i q v_i + \hbar_i r_i f_i, \ i = 0$$

$$\frac{dv_i}{dt} = \frac{r_{i+\frac{1}{2}} k_{i+\frac{1}{2}}}{\hbar_i r_i h_{i+1}} (v_{i+1} - v_i) + \frac{\nu_1}{\hbar_i} - q v_i + f_i, \ i = 0$$

Проведем аппроксимацию граничного условия справа:

$$\int\limits_{r_{i-\frac{1}{2}}}^{r_i} r \frac{\partial u}{\partial t} dt = \int\limits_{r_{i-\frac{1}{2}}}^{r_i} \frac{\partial}{\partial r} \left(r k \frac{\partial u}{\partial r} \right) dr - \int\limits_{r_{i-\frac{1}{2}}}^{r_i} r q u dr - \int\limits_{r_{i-\frac{1}{2}}}^{r_i} r f dr, \ i = N$$

$$\hbar_i r_i \frac{dv_i}{dt} = r k \frac{\partial u}{\partial r} \bigg|_{r = r_i} - r k \frac{\partial u}{\partial r} \bigg|_{r = r_{i-\frac{1}{2}}} - \hbar_i r_i q v_i + \hbar_i r_i f_i, \ i = N$$

Граничное условие:

$$-k \left. \frac{\partial u}{\partial r} \right|_{r = R_R} = -\nu_2(t)$$

Получаем:

$$\hbar_i r_i \frac{dv_i}{dt} = r_i \frac{\nu_2}{h_{i+1}} - r_{i-\frac{1}{2}} k_{i-\frac{1}{2}} \frac{v_i - v_{i-1}}{h_i} - \hbar_i r_i q v_i + \hbar_i r_i f_i, \ i = N$$

$$\frac{dv_i}{dt} = \frac{\nu_2}{\hbar_i} - \frac{r_{i-\frac{1}{2}} k_{i-\frac{1}{2}}}{\hbar_i r_i h_i} (v_i - v_{i-1}) - q v_i + f_i, \ i = N$$

Введем обозначения:

$$B_1 = \frac{r_{i+\frac{1}{2}}k_{i+\frac{1}{2}}}{\hbar_i r_i h_{i+1}} \quad B_2 = \frac{r_{i-\frac{1}{2}}k_{i-\frac{1}{2}}}{\hbar_i r_i h_i}$$

Запишем полученные уравнения:

$$\begin{cases} \frac{dv_i}{dt} = B_1(v_{i+1} - v_i) + \frac{\nu_1}{\hbar_i} - qv_i + f_i, \ i = 0 \\ \frac{dv_i}{dt} = B_1(v_{i+1} - v_i) - B_2(v_i - v_{i-1}) - qv_i + f_i, \ i = 1, 2, \ldots, N-1 \\ \frac{dv_i}{dt} = \frac{\nu_2}{\hbar_i} - B_2(v_i - v_{i-1}) - qv_i + f_i, \ i = N \end{cases}$$

Сгруппируем значения:

$$\begin{cases} \frac{dv_i}{dt} = B_1 v_{i+1} - (B_1 + q_i)v_i + \frac{\nu_1}{\hbar_i} + f_i, \ i = 0 \\ \frac{dv_i}{dt} = B_1 v_{i+1} - (B_1 + B_2 + q_i)v_i - B_2 v_{i-1} + f_i, \ i = 1, 2, \ldots, N-1 \\ \frac{dv_i}{dt} = -(B_2 + q_i)v_i - B_2 v_{i-1} + \frac{\nu_2}{\hbar_i} + f_i, \ i = N \end{cases}$$

Мы получили систему:

$$\begin{pmatrix} \frac{dv_0}{dt} \\ \frac{dv_1}{dt} \\ \vdots \\ \vdots \\ \vdots \\ \frac{dv_{n-1}}{dt} \\ \frac{dv_n}{dt} \end{pmatrix} = \begin{pmatrix} c_0 & b_0 & & & & & 0 \\ a_1 & c_1 & b_1 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & a_{n-1} & c_{n-1} & b_{n-1} \\ 0 & & & & & a_n & c_n \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} + \begin{pmatrix} g_0 \\ g_1 \\ \vdots \\ \vdots \\ \vdots \\ g_{n-1} \\ g_n \end{pmatrix}$$

где:

$$\begin{aligned} & & c_0 &= -(B_1 + q_0) & b_0 &= B_1 & g_0 &= \frac{\nu_1}{\hbar_0} + f_0 \\ a_i &= B_2 & c_i &= -(B_1 + B_2 + q_i) & b_i &= B_1 & g_i &= f_i \\ a_N &= B_2 & c_N &= -(B_2 + q_N) & & & g_N &= \frac{\nu_2}{\hbar_N} + f_N \end{aligned}$$

2.2 Решение системы ОДУ

Система имеет вид:

$$\frac{dv_i}{dt} = Av + g$$

Введем дискретизацию по времени и проинтегрируем на одном из промежутков:

$$\int\limits_{t_n}^{t_n+1} \frac{dv_i}{dt}\, dt = \int\limits_{t_n}^{t_n+1} (Av + g) dt$$

$$v(t_{n+1}) - v(t_n) = \int\limits_{t_n}^{t_n+1} (Av + g)dt$$

$$v(t_{n+1}) = v(t_n) + \int\limits_{t_n}^{t_n+1} (Av + g)dt$$

Добавим к полученному уравнению начальное условие и тем самым получаем систему:

$$\begin{cases} v(t_{n+1}) = v(t_n) + \int\limits_{t_n}^{t_n+1} (Av + g)dt \\ v(t_0) = \varphi(r) \end{cases}$$

Решение задачи сводится к поиску значения интеграла.

2.2.1 Явный метод Эйлера

Мы имеем уравнение:

$$v(t_{n+1}) = v(t_n) + \int\limits_{t_n}^{t_n+1} (Av + g)dt$$

Интерполлируем интеграл по формуле левых треугольников:

$$\int\limits_a^b f(x)\mathrm{d}x \approx f(a)(b - a)$$

Тем самым получаем:

$$v(t_{n+1}) = v(t_n) + (t_{n+1} - t_n)Av(t_n) + (t_{n+1} - t_n)g$$

Введем обозначение:

$$H = (t_{n+1} - t_n)$$

$$v(t_{n+1}) = v(t_n) + HAv(t_n) + Hg$$

$$v(t_{n+1}) = (E + HA)v(t_n) + Hg$$

Явный метод ломанных Эйлера:

$$\begin{cases} v(t_{n+1}) = (E + HA)v(t_n) + Hg \\ v(t_0) = \varphi(r) \end{cases}$$

2.2.2 Неявный метод Эйлера

Теперь проинтерполируем интеграл формулой правых треугольников:

$$\int_a^b f(x)\mathrm{d}x \approx f(b)(b-a)$$

Получаем:

$$v(t_{n+1}) = v(t_n) + HAv(t_{n+1}) + Hg$$
$$(E - HA)v(t_{t_{n+1}}) = v(t_n) + Hg$$

Неявный метод ломанных Эйлера:

$$\begin{cases} (E - HA)v(t_{t_{n+1}}) = v(t_n) + Hg \\ v(t_0) = \varphi(r) \end{cases}$$

2.3 Тестирование

Погрешность решения находится следующим образом:

$$\max_{t_0,\cdots,t_n} = \|\varepsilon(t)\|$$

$$\|\varepsilon(t)\| = \left\|\tilde{v(t)} - v(t)\right\|$$

Для всех тестов возьмем одинаковое интервал для r и t.

$$r \in [\frac{\pi}{6}, \frac{\pi}{3}]$$

$$T \in [0, 1]$$

Для всех тестов возьмем одинаковые значения k и q:

$$k = \frac{\cos(r)}{2} + 3$$

$$q = \frac{\sin(r)}{2} + 2$$

Также для всех тестов возьмем одинаковое число разбиений:

$$N = 16$$

2.3.1 Стационарное решение

$$u(r, t) = r$$

Тогда получаем значение f, как:

$$f(r, t) = \frac{-\cos(r) + r\sin(r) - 6 + r^2\sin(r) + 4r^2}{2r}$$

Значения ν_1 и ν_2 такие:

$$\nu_1 = \frac{\sqrt{3}}{4} + 3$$
$$\nu_2 = 3.25$$

После выполнения программы получаем следующие значения:

H	$\|\varepsilon\|$, явный метод	$\|\varepsilon\|$, неявный метод
1e-1	2.25e+22	5.05e-5
1e-2	2.13e+140	5.05e-5
1e-3	1.09e+330	5.05e-5
1e-4	3.57e-10	5.05e-5
1e-5	3.88e-9	5.05e-5
1e-6	3.84e-8	5.05e-5

Таблица 1: Погрешность теста ДУ со стационарномым решением

2.3.2 Решение стремящееся к стационарному

$$u(r,t) = 4 + e^{-10t}$$

Тогда получаем значение f, как:

$$f(r,t) = \frac{-16 + 4e^{10t}\sin(r) + \sin(r) + 16e^{10t}}{2e^{10t}}$$

Значения ν_1 и ν_2 такие:

$$\nu_1 = 0$$
$$\nu_2 = 0$$

После выполнения программы получаем следующие значения:

H	$\|\varepsilon\|$, явный метод	$\|\varepsilon\|$, неявный метод
1e-1	2.96e+14	1.65e-01
1e-2	4.55e+140	1.36e-02
1e-3	4.52e+292	2.94e-03
1e-4	2.76e-05	4.54e-04
1e-5	2.76e-05	2.98e-04
1e-6	2.76e-03	2.44e-04

Таблица 2: Погрешность теста ДУ с решением стремящимся к стационарному

2.3.3 Нестационарное решение

$$u(r,t) = r + t$$

Тогда получаем значение f, как:

$$f(r,t) = 1 - \frac{\cos(r) - r\sin(r) + 6}{2r} + \frac{t\sin(r) + \sin(r)}{2} + 2t + 2r$$

Значения ν_1 и ν_2 такие:

$$\nu_1 = \frac{\sqrt{3}}{4} + 3$$
$$\nu_2 = 3.25$$

После выполнения программы получаем следующие значения:

H	$\|\varepsilon\|$, явный метод	$\|\varepsilon\|$, неявный метод
1e-1	5.57e+20	3.18e-01
1e-2	1.18e+139	3.3e-02
1e-3	6.48e+300	3.31e-03
1e-4	1.69e-05	3.57e-04
1e-5	1.69e-05	1.29e-05
1e-6	1.69e-05	8.48e-05

Таблица 3: Погрешность теста ДУ с нестационарным решением

Полученные данные дают нам повод убедиться, что явный метод Эйлера непременим для решения жестких систем. При небольшом шаге можно заметить "взрыв погрешности".

В свою же очередь неявный метод Эйлера требует решения системы линейных алгебраических уравнений, что требует значительного увеличения числа вычислений.

3 Заключение

3.1 Вывод

Задание выполнено полностью. Были написаны: метод для приближенного решения дифференциального уравнения, явный и неявный метод Эйлера. Все методы были протестированы. Были выявлены недостатки и преимущества явного и неявного метода Эйлера.

3.2 Код

```cpp
#include <iostream>
#include <iomanip>
#include <vector>
#include <algorithm>
#include <initializer_list>

#include "utils/balance_utils.hpp"
#include "data_table.hpp"
#include "euler.hpp"

int main()
{
  std::size_t N = 16;

  auto data_table = get_data();

  std::cout << std::scientific;
  std::cout << "forward\n";
  for(auto&& H : {1e-1, 1e-2, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6})
  {
    double t0 = 0;
    std::cout << std::setprecision(0) << H << " ";
    std::vector< double > first(N + 1);
    std::vector< double > final(N + 1);

    for (auto&& data : data_table)
    {
      init_solution(N, data, first, t0);
      forward_euler(N, t0, H, data, first, final);

      double x = 0;
      for(double t = t0 + H; t < 1; t += H)
      {
        first.swap(final);

        double eps = forward_euler(N, t, H, data, first, final);
        x = (eps > x ? eps : x);
      }

      std::cout << std::setw(10) << std::setprecision(2) << x << " ";
    }
    std::cout << "\n";
  }

```

```

  std::cout << "backward\n";
  for(auto&& H : {1e-1, 1e-2, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6})
  {
    double t0 = 0;
    std::cout << std::setprecision(0) << H << " ";

    std::vector< double > first(N + 1);
    std::vector< double > final(N + 1);

    for (auto&& data : data_table)
    {
      init_solution(N, data, first, t0);
      backward_euler(N, t0, H, data, first, final);

      double x = 0;
      for(double t = t0 + H; t < 1; t += H)
      {
        first.swap(final);

        double eps = backward_euler(N, t, H, data, first, final);
        x = (eps > x ? eps : x);
      }

      std::cout << std::setw(10) << std::setprecision(2) << x << " ";
    }
    std::cout << "\n";
  }
  return 0;
}
```

```
#ifndef DATA_TABLE_HPP
#define DATA_TABLE_HPP

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

std::vector< Data< double > > get_data()
{
  std::vector< Data< double > > data_table =
  {
    {
      M_PI / 6, M_PI / 3,
      [](double r, double t) -> double { return r; }, //u
      [](double r, double t) -> double { return std::cos(r) / 2 + 3; },//k
```

```
    [](double r, double t) -> double { return std::sin(r) / 2 + 2; }, //q
    [](double r, double t) -> double {
      return (-std::cos(r) + r * std::sin(r) - 6 + r * r * std::sin(r) + 4 * r * r) / (2 * r);
    },
    [](double t) -> double { return std::sqrt(3) / 4 + 3; },
    [](double t) -> double { return 3.25; },
  },
};

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 EULER_HPP
#define EULER_HPP

#include <cstddef>
#include <vector>
#include <iostream>
#include "utils/data.hpp"
#include "utils/balance_utils.hpp"
#include "utils.hpp"
#include "tma.hpp"

template <typename T = double>
double forward_euler(
  std::size_t N, T t, T H,
  const Data< T >& data,
  const std::vector< T >& prev,
  std::vector< T >& next
)
{
  std::vector< T > a(N + 1);
  std::vector< T > c(N + 1);
  std::vector< T > b(N + 1);
  std::vector< T > r(N + 1);
  std::vector< T > x(N + 1);

  std::vector< T > rez(N + 1);

  init_balance(N, data, a, c, b, r, x, t);
  init_solution(N, data, x, t + H);

  for(int i = 0; i <= N; i++)
```

```
  {
    a[i] *= H;
    c[i] = 1 + H * c[i];
    b[i] *= H;
    r[i] *= H;
  }

  multiply(a, c, b, prev, next);
  add(next, r, next);
  return eps(x, next);
}

template <typename T = double>
double backward_euler(
  std::size_t N, T t, T H,
  const Data< T >& data,
  const std::vector< T >& prev,
  std::vector< T >& next
)
{
  std::vector< T > a(N + 1);
  std::vector< T > c(N + 1);
  std::vector< T > b(N + 1);
  std::vector< T > r(N + 1);
  std::vector< T > x(N + 1);

  std::vector< T > rez(N + 1);

  init_balance(N, data, a, c, b, r, x, t);
  init_solution(N, data, x, t + H);

  for (int i = 0; i <= N; i++)
  {
    a[i] *= -H;
    c[i] = 1 - H * c[i];
    b[i] *= -H;
    r[i] *= H;
  }

  add(r, prev, r);
  tma(a, c, b, r, next);
  return eps(x, next);
}

#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 >
void add( const std::vector< T >& v1 ,
          const std::vector< T >& v2 ,
          std::vector< T >& rez
)
{
  std::size_t size = rez.size ();
  assert(v1.size() == size );
  assert(v2.size() == size );

  std::vector< double > tmp(size);

```

```
  for( std::size_t i = 0; i < size; i++)
  {
    tmp [i] = v1[i] + v2[i];
  }

  eps(v1, v2);
  rez.swap(tmp);
}

template< typename T >
void multiply(
  const std::vector< T >& a,
  const std::vector< T >& c,
  const std::vector< T >& b,
  const std::vector< T >& x,
  std::vector< T >& rez
)
{
  std::size_t size = rez.size();

  assert(a.size() == size);
  assert(c.size() == size);
  assert(b.size() == size);
  assert(x.size() == size);

  std::vector< T > tmp(size);

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

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

  tmp[size - 1] = a[size - 1] * x[size-2] + b[size - 1] * x[size - 1];

  rez.swap(tmp);
}

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 <iostream>
```

```
#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,
  T t
)
{
  static_assert(std::is_floating_point< T >::value);

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

  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);
  double B1 = (r_2(0) * k_2(0, t)) / (h_1(1) * r_1(0) * h_2(0));
  double B2 = 0;

  a[0] = 0;
  c[0] = -(B1 + q_1(0, t));
  b[0] = B1;
  r[0] = nu_1(t) / h_2(0) + f_1(0, t);
  x[0] = data.u(r_1(0), t);

```

```
  for (int i = 1; i < N; i++)
  {
    B1 = (r_2(i) * k_2(i, t)) / (h_1(i + 1) * r_1(i) * h_2(i));
    B2 = (r_2(i - 1) * k_2(i - 1, t)) / (h_1(i) * r_1(i) * h_2(i));

    a[i] = B2;
    c[i] = -(B1 + B2 + q_1(i, t));
    b[i] = B1;
    r[i] = f_1(i, t);
    x[i] = data.u(r_1(i), t);
  }

  B2 = (r_2(N - 1) * k_2(N - 1, t)) / (h_1(N) * r_1(N) * h_2(N));

  a[N] = B2;
  c[N] = -(B2 + q_1(N, t));
  b[N] = 0;
  r[N] = nu_2(t) / h_2(N)  + f_1(N, t);
  x[N] = data.u(r_1(N), t);
}


template< typename T >
void init_solution(
  std::size_t N,
  const Data< T >& data,
  std::vector< T >& x,
  T t
)
{
  static_assert(std::is_floating_point< T >::value);

  auto r_1 = r1< T >(N, data);
  for (std::size_t i = 1; i <= N; i++)
  {
    x[i - 1] = data.u(r_1(i), t);
  }
}
#endif
```

```
#ifndef DATA_HPP
#define DATA_HPP

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

```

```
template< typename T = double >
using time_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 time_func_t< T > nu_1;
  const time_func_t< T > nu_2;
};

#endif
```

```
#ifndef GRID_HPP
#define GRID_HPP

#include <cstddef>
#include <iostream>
#include <cassert>
#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, T t)
    {
      auto r_1 = r1(N_, data_);

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

  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, T t)
    {
      auto r_2 = r2(N_, data_);
      auto r = r_2(i);
      return data_.k(r, t);
    }

  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, T t)
    {
      auto r_1 = r1(N_, data_);
      auto r = r_1(i);

      return data_.q(r, t);
    }

  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, T t)
    {
      auto r_1 = r1(N_, data_);
      auto r = r_1(i);

      return data_.f(r, t);
    }

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

#endif
```