Laboratorium 6

Artem Buhera
GĆ01 135678

19.04.2021

Dany laboratorium polega na zaimplementowaniu programu do rozwiązywania układów równań w postaci $A\vec{x}=\vec{b}$, gdzie *A* - macierz rzadka trójdiagonalna, przy użyciu algorytm Thomasa bez wyboru elementów podstawowych.

Algorytm Thomasa stosuje dekompozycję LU, eksploatując to, że macierz *A* jest trójdiagonalna. Możemy traktować macierz *A* jako trzy wektory $\vec{l}, \vec{d}, \vec{u}$ odpowiednio pod, na oraz nad przekątną macierzy.

$$\begin{pmatrix} d_1 & u_1 & & & & 0 \\ l_2 & d_2 & u_2 & & & \\ & l_3 & d_3 & u_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & l_{N-1} & d_{N-1} & u_{N-1} \\ 0 & & & & l_N & u_N \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{N-1} \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{N-1} \\ b_N \end{pmatrix}$$

Wprowadzamy współczynniki η_i zamiast głównej przekątnej d_i oraz r_i zamiast elementów wektora b_i

Teraz zaczynamy redukcję, w wyniku której otrzymamy nowe wartości $\vec{\eta}$ oraz $\vec{r}$

$$\eta_1 = d_1$$
$$r_1 = b_1$$

$$dla \quad i=2,\ldots,N:$$
$$\begin{cases} \eta_i = d_i - l_i \, \eta_{i-1}^{-1} \, u_{u-1} \\ r_i = b_i - l_i \, \eta_{i-1}^{-1} \, r_{u-1} \end{cases}$$

To przekształcenie pozwala nam znaleźć wektor rozwiązań początkowego układu równań $\vec{x}$

$$x_N = \eta_N^{-1} \, r_N$$
$$dla \quad i=N-1,\ldots,1:$$
$$x_i = \eta_i^{-1} \left(r_i - u_i \, x_{i+1}\right)$$

Do implementacji programu w języku C++ zostały stworzone dwie dodatkowe klasy *Matrix* oraz *Vector* na podstawie struktury danych *std::array* ze standardowej biblioteki szablonów.

Wybór danej struktury zamiast zwykłych list pozwala na użycie stworzonych metod i funkcji dla macierzy i wektorów o dynamicznym rozmiarze praktycznie bez żadnych obciążeń pamięciowych i obliczeniowych.

vector.h:

```cpp
#include <array>
#include <ostream>
#include <iostream>
#include <iomanip>

#ifndef VECTOR_H
#define VECTOR_H
using namespace std;

template <int N>
class Vector: public array<double,N> {
public:

    void swapElements(int a, int b) {
        std::swap((*this)[a], (*this)[b]);
    }

    void print(int width=10) {
        for (auto value : (*this)) {
            if (abs(value) < 1.0e-300) value = 0.0;
            std::cout << std::setw(width) << value << " ";
        }
        std::cout << std::endl;
    }

    void print(const std::string &text, int width=10) {
        std::cout << text << std::endl;
        print(width);
    }
};

#endif
```

matrix.h:

```
#include "vector.h"

#ifndef MATRIX_H
#define MATRIX_H
using namespace std;

template<int N, int M>
class Matrix : array<Vector<N>, M> {
public:
    Matrix<N, M>() = default;
    Matrix<N, M>(array<Vector<N>, M> a) : array<Vector<N>, M> (a) {}

    using array<Vector<N>, M>::operator[];

    void swapRows(int a, int b) {
        swap((*this)[a], (*this)[b]);
    };

    void print(int width=8) const {
        for (Vector row : (*this))
            row.print(width);
        std::cout << std::endl;
    }

    void print(string text, int width=8) const {
        std::cout << text << std::endl;
        for (Vector row : (*this))
            row.print(width);
        std::cout << std::endl;
    }
};

#endif
```

Główny program:

```
#include <iostream>
#include <vector.h>

using namespace std;


template <int N>
void calculateEta(Vector<N-1> &l, Vector<N> &d, Vector<N-1> &u);
template <int N>
void calculateR(Vector<N> &b, Vector<N - 1> &l, Vector<N> &eta);
template <int N>
Vector<N> solve(Vector<N> &eta, Vector<N> &r, Vector<N-1> &u);


int main() {
    Vector<6> b = {31.0, 165.0/4.0, 917.0/30, 851.0/28.0, 3637.0/90,
332.0/11.0};
    cout << "Rozwiązujemy układ równań Ax = b używając algorytm Thomasa" <<
endl;
    b.print("Wektor b:");

    cout << endl << "Macierz trójdiagonalną A przedstawiamy jako trzy wektory l,
d oraz u" << endl;
    Vector<5> l = {1.0/3.0, 1.0/5.0, 1.0/7.0, 1.0/9.0, 1.0/11.0};
    Vector<6> d = {10.0, 20.0, 30.0, 30.0, 20.0, 10};
    Vector<5> u = {1.0/2.0, 1.0/4.0, 1.0/6.0, 1.0/8.0, 1.0/10.0};
    l.print("Wektor l:");
    d.print("Wektor d:");
    u.print("Wektor u:");


    cout << endl << "Najpierw przekształcamy macierz A - zamiast wektora d
obliczamy nowy wektor eta" << endl;
    calculateEta(l, d, u);
    Vector eta = d;
    eta.print("Wektor eta: ");

    cout << endl << "Następnie z wektor b przekształcamy na r" << endl;
    calculateR(b, l, eta);
    Vector r = b;
    b.print("Wektor r: ");

    cout << endl << "Ostatecznie obliczamy wektor rozwiązań x" << endl;
    Vector x = solve(eta, r, u);
    x.print("Wektor x:");
```

```
    cout << endl << "Sprawdźmy wynik mnożenia Ax" << endl;
    Vector<6> expected_b;
    expected_b[0] = d[0]*x[0] + u[0]*x[1];
    for (int i = 1; i < 6; i++) {
        expected_b[i] = l[i-1]*x[i-1] + d[i]*x[i] + u[i]*x[i+1];
    }

    expected_b.print("Wektor Ax:");
    cout << endl << "Otrzymaliśmy wyniki bardzo bliskie do wektora b";
}


template <int N>
void calculateEta(Vector<N - 1> &l, Vector<N> &d, Vector<N - 1> &u) {
    for (int i = 1; i < N; i++)
        d[i] -= l[i-1] / d[i-1] * u[i-1];
}

template <int N>
void calculateR(Vector<N> &b, Vector<N - 1> &l, Vector<N> &eta) {
    for (int i = 1; i < N; i++)
        b[i] -= l[i-1] / eta[i-1] * b[i-1];
}

template <int N>
Vector<N> solve(Vector<N> &eta, Vector<N> &r, Vector<N-1> &u) {
    Vector<N> x;

    x[N-1] = r[N-1] / eta[N-1];
    for (int i = N-2; i >= 0; i--)
        x[i] = (r[i] - u[i] * x[i+1]) / eta[i];

    return x;
}
```

Wynik działana programu:

```
Rozwiązujemy układ równań Ax = b używając algorytm Thomasa
Wektor b:
        31      41.25    30.5667    30.3929    40.4111    30.1818

Macierz trójdiagonalną A przedstawiamy jako trzy wektory l, d oraz u
Wektor l:
  0.333333        0.2   0.142857   0.111111  0.0909091
Wektor d:
        10         20         30         30         20         10
Wektor u:
       0.5       0.25   0.166667      0.125        0.1

Najpierw przekształcamy macierz A - zamiast wektora d obliczamy nowy wektor eta
Wektor eta:
        10    19.9833    29.9975    29.9992    19.9995    9.99955

Następnie z wektor b przekształcamy na r
Wektor r:
        31    40.2167    30.1642    30.2492    40.2991    29.9986

Ostatecznie obliczamy wektor rozwiązań x
Wektor x:
         3          2          1          1          2          3

Sprawdźmy wynik mnożenia Ax
Wektor Ax:
        31    41.2167    30.5642    30.3921    40.4102    30.1805

Otrzymaliśmy wyniki bardzo bliskie do wektora b
```