METODY NUMERYCZNE

ZADANIE 1

Artur Guniewicz

1 O. Dobierając odpowiednie algorytmy (wybór trzeba uzasadnić!), rozwiązać następujące układy równań:

(a)

$$\begin{bmatrix} 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{bmatrix} \tag{1}$$

(b)

$$\begin{bmatrix} 4 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{bmatrix} \tag{2}$$

a) **Metoda**: Faktoryzacja Choleskiego

Jest to macierz symetryczna i dodatnio określona więc można zastosować faktoryzację Choleskiego. Polega ona na przedstawieniu macierzy A jako iloczynu macierzy C i jej układu transponowanego ($A=CC^T$). Zdecydowałem się na wybór tej metody ponieważ wymaga ona zaledwie jednej macierzy dodatkowej C, a dzięki konstrukcji macierzy A obliczenie współczynników macierzy Choleskiego odbywa się w czasie liniowym. Ponadto niektóre współczynniki będą się zerować co ułatwi rozwiązanie równania.

Musimy rozwiązać układ równań macierzowych:

$Cy=b$ -> forwardsubstitution
$C^Tx=y$ -> backsubstitution

Kod programu:

```
/*
*************************************************************
*                                                           *
*                Artur Guniewicz - Zadanie 1a               *
*                                                           *
*************************************************************
*/

#include <iostream>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <iomanip> // zawiera funkcję setprecision()

using namespace std;

// drukuje równanie Ax=b
void printEquation(const long double A[7][7], const long double x[7],
const long double b[7])
{
   for (int i = 0; i < 7; i++)
   {
       cout << setprecision(3) << fixed;

       cout << "[ ";

       for (int j = 0; j < 7; j++)
           cout << A[i][j] << " ";

       cout << "] [ " << x[i] << " ] [ " << b[i] << " ]" << endl;
   }

   cout << endl;
}

// drukuje wektor
void printVector(string name, const long double x[7])
{
```

```
    cout << setprecision(10);

    cout << "Wektor " << name << ":" << endl;

    cout << name << " = [ ";
    for (int i = 0; i < 7; i++)
        cout << x[i] << " ";
    cout << "]" << endl;
}

// faktoryzacja Choleskiego
void CholeskyDecomposition(const long double A[7][7], long double *x,
const long double *b)
{
    long double C[7][7] = {0};

    C[0][0] = sqrt((A[0][0]));

    for (int i = 1; i < 7; i++)
    {
        C[i][i - 1] = A[i][i - 1] / C[i - 1][i - 1];
        C[i][i] = sqrt((A[i][i] - C[i][i - 1] * C[i][i - 1]));
    }

    double long y[7] = {0};

    //forwardsubstitution Cy=b, gdzie y=C^Tx
    y[0] = b[0] / C[0][0];

    for (int i = 1; i < 7; i++)
        y[i] = (b[i] - y[i - 1] * C[i][i - 1]) / C[i][i];

    //backsubstitution C^Tx=y
    x[6] = y[6] / C[6][6];

    for (int i = 5; i >= 0; i--)
        x[i] = (y[i] - C[i + 1][i] * x[i + 1]) / C[i][i];
}

int main()
```

```cpp
{
    long double A[7][7] = {0};
    long double x[7] = {0};
    long double b[7] = {0};

    // uzupełnienie macierzy A i b
    for (int i = 0; i < 7; i++)
    {
        if (i == 0)
            A[i][i + 1] = 1;

        else if (i == 6)
            A[i][i - 1] = 1;

        else
        {
            A[i][i - 1] = 1;
            A[i][i + 1] = 1;
        }

        A[i][i] = 4;
        b[i] = i + 1;
    }

    cout << endl;

    CholeskyDecomposition(A, x, b);

    printEquation(A, x, b);

    printVector("x", x);

    return 0;
}
```

Kompilacja:

cd “adres pliku” && g++ Zadanie1_a.cpp -g -std=c++14 -ansi -pedantic -Wall -lm -o Zadanie1_a && ./Zadanie1_a

Wyniki:

```
[ 4.000 1.000 0.000 0.000 0.000 0.000 0.000 ] [ 0.167 ] [ 1.000 ]
[ 1.000 4.000 1.000 0.000 0.000 0.000 0.000 ] [ 0.333 ] [ 2.000 ]
[ 0.000 1.000 4.000 1.000 0.000 0.000 0.000 ] [ 0.502 ] [ 3.000 ]
[ 0.000 0.000 1.000 4.000 1.000 0.000 0.000 ] [ 0.660 ] [ 4.000 ]
[ 0.000 0.000 0.000 1.000 4.000 1.000 0.000 ] [ 0.859 ] [ 5.000 ]
[ 0.000 0.000 0.000 0.000 1.000 4.000 1.000 ] [ 0.904 ] [ 6.000 ]
[ 0.000 0.000 0.000 0.000 0.000 1.000 4.000 ] [ 1.524 ] [ 7.000 ]

Wektor x:
x = [ 0.1667893962 0.3328424153 0.5018409426 0.6597938144 0.8589837997 0.9042709867 1.5239322533 ]
```

Komentarz:

Program dokonuje faktoryzacji Choleskiego macierzy A, a następnie rozwiązuje równania macierzowe wyliczając wartości wektora x.

b) **Metoda**: Wzór Shermana-Morrisona + faktoryzacja Choleskiego

Skorzystanie z tej metody pozwoli rozwiązać równanie w czasie liniowym, ponieważ wymaga rozwiązania układów równań z macierzami trójdiagonalnymi (jak w podpunkcie a), a następnie wyliczenia bezpośrednio wartości wektora x.

Mamy:

$$A=\begin{pmatrix}3&1&0&0&0&0&0\\1&4&1&0&0&0&0\\0&1&4&1&0&0&0\\0&0&1&4&1&0&0\\0&0&0&1&4&1&0\\0&0&0&0&1&4&1\\0&0&0&0&0&1&3\end{pmatrix},\ v=u^T,\ u=\begin{pmatrix}1\\0\\0\\0\\0\\0\\1\end{pmatrix},\ uv=\begin{pmatrix}1&0&0&0&0&0&1\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\1&0&0&0&0&0&1\end{pmatrix},\ A_1=A+u\,v^T$$

Obliczamy równania:

$Az=b$

$Aq=u$

$$\mathbf{w}=\mathbf{A}_1^{-1}\mathbf{b}=\left(\mathbf{A}^{-1}-\frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^T\mathbf{A}^{-1}}{1+\mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}}\right)\mathbf{b}$$

$$\mathbf{w}=\mathbf{z}-\frac{\mathbf{v}^T\mathbf{z}}{1+\mathbf{v}^T\mathbf{q}}\mathbf{q}\,.$$

i dalej:

$A_1x=b$

$A_1^{-1}A_1x=A_1^{-1}b$

i wreszcie:

$x=A_1^{-1}b$

Kod programu:

```
/*
*************************************************************
*                                                           *
*                Artur Guniewicz - Zadanie 1b               *
*                                                           *
*************************************************************
*/

#include <iostream>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <iomanip> // zawiera funkcję setprecision()

using namespace std;

// drukuje równanie Ax=b
void printEquation(const long double A[7][7], const long double x[7],
const long double b[7])
{
   for (int i = 0; i < 7; i++)
   {
       cout << setprecision(3) << fixed;

       cout << "[ ";

       for (int j = 0; j < 7; j++)
           cout << A[i][j] << " ";

       cout << "] [ " << x[i] << " ] [ " << b[i] << " ]" << endl;
   }

   cout << endl;
}

// drukuje wektor
void printVector(string name, const long double x[7])
{
```

```cpp
    cout << setprecision(10);

    cout << "Wektor " << name << ":" << endl;

    cout << name << " = [ ";
    for (int i = 0; i < 7; i++)
        cout << x[i] << " ";
    cout << "]" << endl;
}

// faktoryzacja Choleskiego
void CholeskyDecomposition(const long double A[7][7], long double *x,
const long double *b)
{
    long double C[7][7] = {0};

    C[0][0] = sqrt((A[0][0]));

    for (int i = 1; i < 7; i++)
    {
        C[i][i - 1] = A[i][i - 1] / C[i - 1][i - 1];
        C[i][i] = sqrt((A[i][i] - C[i][i - 1] * C[i][i - 1]));
    }

    double long y[7] = {0};

    //forwardsubstitution Cy=b, gdzie y=C^Tx
    y[0] = b[0] / C[0][0];

    for (int i = 1; i < 7; i++)
        y[i] = (b[i] - y[i - 1] * C[i][i - 1]) / C[i][i];

    //backsubstitution C^Tx=y
    x[6] = y[6] / C[6][6];

    for (int i = 5; i >= 0; i--)
        x[i] = (y[i] - C[i + 1][i] * x[i + 1]) / C[i][i];
}
```

```
void ShermanMorrison(long double A[7][7], long double *x, const long
double *b, long double uv[7][7])
{
    //Az=b
    long double z[7] = {0};
    CholeskyDecomposition(A, z, b);

    //Aq=u
    long double q[7] = {0};
    CholeskyDecomposition(A, q, uv[0]);

    //x=A1^(-1)*b, gdzie A1=A+uv
    long double constant = (z[0] + z[6]) / (1 + q[0] + q[6]);
    for (int i = 0; i < 7; i++)
    {
        x[i] = z[i] - constant * q[i];
    }

    //A+uv
    A[0][0] = 4;
    A[6][6] = 4;
    A[0][6] = 1;
    A[6][0] = 1;
}

int main()
{
    long double A[7][7] = {0};
    long double x[7] = {0};
    long double b[7] = {0};
    long double uv[7][7] = {0};

    // uzupełnienie macierzy A, b i uv
    for (int i = 0; i < 7; i++)
    {
        A[i][i] = 4;

        if (i == 0)
        {
            A[i][i + 1] = 1;
```

```
            A[i][i] = 3;
            uv[0][0] = 1;
            uv[0][6] = 1;
        }
        else if (i == 6)
        {
            A[i][i - 1] = 1;
            A[i][i] = 3;
            uv[6][0] = 1;
            uv[6][6] = 1;
        }
        else
        {
            A[i][i - 1] = 1;
            A[i][i + 1] = 1;
        }

        b[i] = i + 1;
    }

    cout << endl;

    ShermanMorrison(A, x, b, uv);

    printEquation(A, x, b);

    printVector("x", x);

    return 0;
}
```

Kompilacja:

cd “adres pliku” && g++ Zadanie1_b.cpp -g -std=c++14 -ansi -pedantic -Wall -lm -o Zadanie1_b && ./Zadanie1_b

Wyniki:

```
[ 4.000 1.000 0.000 0.000 0.000 0.000 1.000 ] [ -0.260 ] [ 1.000 ]
[ 1.000 4.000 1.000 0.000 0.000 0.000 0.000 ] [ 0.447 ] [ 2.000 ]
[ 0.000 1.000 4.000 1.000 0.000 0.000 0.000 ] [ 0.472 ] [ 3.000 ]
[ 0.000 0.000 1.000 4.000 1.000 0.000 0.000 ] [ 0.667 ] [ 4.000 ]
[ 0.000 0.000 0.000 1.000 4.000 1.000 0.000 ] [ 0.862 ] [ 5.000 ]
[ 0.000 0.000 0.000 0.000 1.000 4.000 1.000 ] [ 0.886 ] [ 6.000 ]
[ 1.000 0.000 0.000 0.000 0.000 1.000 4.000 ] [ 1.593 ] [ 7.000 ]

Wektor x:
x = [ -0.2601626016 0.4471544715 0.4715447154 0.6666666667 0.8617886179 0.8861788618 1.5934959350 ]
```

Komentarz:

Program dwukrotnie rozwiązuje układ równań z macierzami trójdiagonalnymi za pomocą faktoryzacji Choleskiego. Następnie oblicza wektory z i q, których używa do policzenia wektora x za pomocą wzoru Shermana-Morrisona. Sposób ten jest szybki i dokładny.