Wydział WIMiIP	Imię i nazwisko: Zuzanna Będkowska	Rok: 2	Grupa: 1	Data: 10.04.2022
Metody Numeryczne	Temat: Rozwiązywanie układu równań liniowych metodą Jacobiego			

Zadanie 1: wyznaczenie rozwiązania układu równań liniowych po określonej liczbie iteracji za pomocą metody Jacobiego

Do realizacji zadania stworzono następujący kod:

```cpp
#include <iostream>
#include <vector>
#include <fstream>
#include <string>
#include <stdio.h>
#include <cmath>
#include <iomanip>

using namespace std;

void wypisz_macierz(vector <vector< double>>& macierz)
{
    for (auto i : macierz)
    {
        for (auto j : i)
        {
            cout << setw(10) << right << j << " ";
        }
        cout << "\n";
    }
}

vector <double> operator * (vector <vector <double>>& macierz1, vector <double>& macierz2)
{
    vector <double> wynik(macierz2.size(), 0);
    for (int i = 0; i < macierz1.size(); ++i)
    {
        double pomoc = 0.0;
        for (int k = 0; k < macierz1[i].size(); ++k)
        {
            pomoc += macierz1[i][k] * macierz2[k];
        }
        wynik[i] = pomoc;
    }
    return wynik;
}

vector <double> operator - (vector <double>& macierz1, vector <double>& macierz2)
{
    vector <double> wynik(macierz1.size(), 0.0);
    for (int i = 0; i < macierz1.size(); ++i)
    {
        wynik[i] = macierz1[i] - macierz2[i];
    }
    return wynik;
}

int main()
{
    int n; //wymiary macierzy nxn
    int z; //iteracje
    ifstream czytaj("RURL_dane1.txt"); //wczytywanie z pliku
    czytaj >> n; //wczytanie ilosci wezlow
    vector <double> wolne(n, 0); //na wyrazy wolne
    vector <vector <double>> wyniki(1, wolne); // na rozwiazania
    vector <vector<double>> wspolczynniki(n, wolne); //na wspolczynniki
    vector <vector<double>> L(n, wolne);
    vector <vector<double>> U(n, wolne);
    vector <vector<double>> D(n, wolne);
    vector <vector<double>> DO(n, wolne);
    vector <vector<double>> L_U(n, wolne);
    for (int i = 0; i < n; ++i) //wczytywanie wspolrzednych wezlow
    {
        for (int j = 0; j < n; ++j)
        {
            double a;
            czytaj >> a;
            wspolczynniki[i][j] = a;
        }
        double b;
        czytaj >> b;
        wolne[i] = b;
    }
    cout << "Uklad Rownan:\n";
```

```cpp
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < n; ++j)
        {
            cout << setw(5) << right << wspolczynniki[i][j] << " ";
        }
        cout << setw(5) << right << wolne[i] << "\n";
    }
    bool test1 = true; //jesli chociaz 1 test sie wywali to zmiana na false
    bool test2 = false; //jesli chociaz 1 test jest ok to zmiana na true
    for (int i = 0; i < n; ++i)
    {
        if (test2 = true)
        {
            break;
        }
        double suma = 0.0;
        for (int j = 0; j < n; ++i)
        {
            if (i == j)
            {
                continue;
            }
            else
            {
                suma += wspolczynniki[i][j];
            }
        }
        if (suma > wspolczynniki[i][i])
        {
            test1 = false;
            cout << "Warunek nie jest spelniony, przerwanie programu!\n";
            return 0;
        }
        if (suma < wspolczynniki[i][i])
        {
            test2 = true;
```

```cpp
            break;
        }
    }
    if (test1 == true && test2 == true)
    {
        cout << "Macierz diagonalnie slabo dominujaca! Program kontunuuje dzialanie\n";
    }
    else
    {
        cout << "Warunek nie jest spelniony, przerwanie programu!\n";
        return 0;
    }
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < n; ++j)
        {
            if (i == j)
            {
                D[i][i] = wspolczynniki[i][i];
                DO[i][i] = wspolczynniki[i][i];
                continue;
            }
            if (i > j)
            {
                L[i][j] = wspolczynniki[i][j];
                L_U[i][j] = wspolczynniki[i][j];
            }
            if (i < j)
            {
                U[i][j] = wspolczynniki[i][j];
                L_U[i][j] = wspolczynniki[i][j];
            }
        }
    }
    cout << "Macierz dolna:\n";
    wypisz_macierz(L);
    cout << "Macierz gorna:\n";
```

```cpp
    wypisz_macierz(U);
    cout << "Macierz dolna + gorna:\n";
    wypisz_macierz(L_U);
    cout << "Macierz diagonalna:\n";
    wypisz_macierz(D);

    //wyznaczanie odwrotnej dopelnieniami algebraicznymi
    for (int i = 0; i < n; ++i)
    {
        DO[i][i] = 1/D[i][i] * pow(-1, i + i);
    }
    cout << "Macierz diagonalna odwrotna:\n";
    wypisz_macierz(DO);
    cout << "Podaj ilosc iteracji: ";
    cin >> z;
    cout << "Rozwiazanie po " << z << "iteracjach:\n";
    for (int Z = 0; Z < z; ++Z)
    {
        vector <double> puste(n, 0);
        wyniki.push_back(puste);
        vector <double> iloczyn = L_U * wyniki[Z];
        vector <double> skladnik = wolne - iloczyn;
        wyniki[Z + 1] = DO * skladnik;
    }
    //wypisz_macierz(wyniki);
    for (int i = 0; i < n; ++i)
    {
        cout << "x" << i << " = " << wyniki[z][i] << "\n";
    }
}
```

W pierwszych krokach algorytmu sprawdzono czy dana macierz jest diagonalnie słabo dominująca uzywając wzorów z instrukcji .Jeśli dana macierz spełniała warunki zadania, wyznaczano macierze: górną (w programie oznaczoną jako U), dolną (w programie oznaczoną jako L), sumę górnej i dolnej (w programie oznaczoną jako L_U) i diagonalną (w programie oznaczoną jako D). Następnie wyznaczono macierz diagonalną odwrotną (w programie oznaczoną jako DO) za pomocą metody dopełnień algebraicznych.

Korki:

I. Dla każdego elementu a_{ii} dla $i \in < 0, n)$ dopełnienie algebraiczne elementu to: $\frac{wyznacznik\ macierzy}{a_{ii}} \times (-1)^{i^2}$. Z własności macierzy dopełnień algebraicznych wiadomo, że otrzymana w ten sposób macierz również jest macierzą diagonalną.

II. Macierz transponowana dla macierzy diagonalnej to ta sama macierz, więc pominięto transpozycję macierzy.

III. Aby otrzymać macierz odwrotną, macierz dopełnień trzeba podzielić przez wyznacznik macierzy danej, co oznacza, że wartość każdego elementu macierzy to: $\frac{1}{a_{ii}} \times (-1)^{i^2}$.

Po otrzymaniu macierzy DO wykorzystano następujący wzór do wyznaczenia rozwiązań (wykorzystując oznaczenia z programu):

$$wyniki[i + 1] = DO \times (wolne - L\_U \times wyniki[i])$$

gdzie $i \in < 0, n)$. Aby ułatwić obliczenia na macierzach, zdefiniowano dla nich podstawowe działania matematyczne i ich operatory (wypisywanie, odejmowanie i dodawanie).

Dla przykładu z zajęć i 5 iteracji, otrzymano następujące wyniki:

```
Konsola debugowania programu Microsoft Visual Studio
Uklad Rownan:
    8     2     2     4     5
    2     5     1     1    -4
    0     3     4     1     2
   -1    -2     1     5     7
Macierz diagonalnie slabo dominujaca! Program kontunuuje dzialanie
Macierz dolna:
         0          0          0          0
         2          0          0          0
         0          3          0          0
        -1         -2          1          0
Macierz gorna:
         0          2          2          4
         0          0          1          1
         0          0          0          1
         0          0          0          0
Macierz dolna + gorna:
         0          2          2          4
         2          0          1          1
         0          3          0          1
        -1         -2          1          0
Macierz diagonalna:
         8          0          0          0
         0          5          0          0
         0          0          4          0
         0          0          0          5
Macierz diagonalna odwrotna:
     0.125          0          0          0
         0        0.2          0          0
         0          0       0.25          0
         0          0          0        0.2
Podaj ilosc iteracji: 5
Rozwiazanie po 5iteracjach:
x0 = 0.28535
x1 = -1.2878
x2 = 1.28868
x3 = 0.692447

C:\Users\Zuza\source\repos\MetodyNumeryczne_lab5\x64\Debug\MetodyNumeryczne_lab5.exe (proces 36256) zakończono z kodem 0
.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

W porównaniu z wynikami uzyskanymi za pomocą metody Gaussa-Crouta:

```
Konsola debugowania programu Microsoft Visual Studio
Podaj ilosc powtorzen: 1
Przed obliczeniami:
         8          2          2          4          5
         2          5          1          1         -4
         0          3          4          1          2
        -1         -2          1          5          7
Po doprowadzeniu do macierzy trojkatnej:
         8          2          2          4          5
         0        4.5        0.5          0      -5.25
         0          0    3.66667          1        5.5
         0          0          0    5.10606    3.41667
Rozwiazanie:
         x1 = 0.289318
         x2 = -1.31306
         x3 = 1.31751
         x4 = 0.669139

C:\Users\Zuza\source\repos\MetodyNumeryczne_lab4\x64\Debug\MetodyNumeryczne_lab4.exe (proces 30520) zakończono z kodem 0
.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Porównanie wyników z metodą Gaussa i wyznaczenie błędu bezwzględnego i względnego:

Lp	x_{Gauss}	x_{Jacobi}	$\Delta x = x_{Gauss} - x_{Jacobi}$
1	0.289318	0.28535	0,003968
2	-1.31306	-1.2878	0,02526
3	1.31751	1.28868	0,02883
4	0.669139	0.692447	0,023308

Zadanie 2: wyznaczenie rozwiązania układu równań liniowych z określoną dokładnością za pomocą metody Jacobiego

Kod z poprzedniego zadania rozszerzono o wypisywanie wartości błędu i sprawdzanie, czy wszystkie wartości będące przybliżonym rozwiązaniem są obarczone błędem mniejszym niż dany:

```cpp
    cout << "Przyblizanie wartosci rozwiazan - wartosc bledu:" << 0.000001 << "\nRoznice miedzy kolejnymi iteracjami:\n" << setw(15) << right << "
    for (int i = 1; i <= n; ++i)
    {
        cout << setw(15) << right << "x" <<i << "";
    }
    cout << "\n";
    int licz = 0; //tu bedzie max ilosc iteracji
    while (licz <= 1000)
    {
        licz++;
        wynik1 = wynik2;
        vector <double> iloczyn = L_U * wynik1;
        vector <double> skladnik = wolne - iloczyn;
        wynik2 = DO * skladnik;
        bool test = false;
        cout << setw(15) << right << licz << ":";
        for (int i = 0; i < n; ++i)
        {
            cout << setw(15) << right << abs(wynik1[i] - wynik2[i]) << " ";
        }
        cout << "\n";
        for (int i = 0; i < n; ++i)
        {
            if (abs(wynik1[i] - wynik2[i]) < 0.000001)
            {
                test = true;
            }
            else
            {
                test = false;
                break;
            }
        }
        if (test)
        {
            cout << setw(15) << right << "zakonczono, ilosc iteracji: " << licz;;
            break;
        }
    }
    //wypisz_macierz(wyniki);
    cout << "\nWyniki:\n";
    for (int i = 0; i < n; ++i)
    {
        cout << "x" << i << " = " << wynik2[i] << "\n";
    }
}
```

Za maksymalna ilość iteracji podano 1000.

Wynik dla $\varepsilon = 0,000001$:

```
Konsola debugowania programu Microsoft Visual Studio
Uklad Rownan:
    8     2     2     4     5
    2     5     1     1    -4
    0     3     4     1     2
   -1    -2     1     5     7
Macierz diagonalnie slabo dominujaca! Program kontunuuje dzialanie
Przyblizanie wartosci rozwiazan - wartosc bledu: 1e-06
Roznice miedzy kolejnymi iteracjami:
             nr              x1              x2              x3              x4
             1:           0.625             0.8             0.5             1.4
             2:           0.625            0.63            0.25           0.295
             3:          0.2425           0.259         0.54625           0.427
             4:       0.0121875         0.12085          0.0875         0.04285
             5:       0.0306625        0.004055        0.079925       0.0284025
             6:      0.00679375       0.0225695      0.00405937       0.0082305
             7:      0.00874278      0.00355173       0.0189847       0.0111984
             8:     3.49062e-05      0.00505438     0.000135813     0.000627704
             9:      0.00154349     0.000112341      0.00394771      0.00205589
            10:     1.29349e-05      0.00099576     0.000429718     0.000435907
            11:     0.000359464     3.93622e-06     0.000855797     0.000481661
            12:     2.78652e-05     0.000218613     0.000123367     0.000100841
            13:     7.42319e-05     1.56513e-05      0.00018917     0.000106546
            14:     9.89314e-06     4.62176e-05     3.83749e-05     2.92481e-05
            15:     1.65847e-05     5.78261e-06     4.19752e-05     2.41834e-05
            16:     3.04354e-06     1.01923e-05     1.03828e-05     7.39114e-06
            17:     3.64794e-06     1.81575e-06     9.49199e-06     5.54476e-06
            18:     8.53321e-07     2.24862e-06       2.748e-06     1.89511e-06
            19:     8.22709e-07     5.11907e-07     2.16024e-06     1.27838e-06
            20:     2.27108e-07     5.05455e-07     7.03526e-07     4.72269e-07
zakonczono, ilosc iteracji: 20
Wyniki:
x1 = 0.289317
x2 = -1.31306
x3 = 1.31751
x4 = 0.66914

C:\Users\Zuza\source\repos\MetodyNumeryczne_lab5\x64\Debug\MetodyNumeryczne_lab5.exe (proces 5064) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Wynik dla $\varepsilon = 0,001$:

```
Konsola debugowania programu Microsoft Visual Studio
Uklad Rownan:
    8     2     2     4     5
    2     5     1     1    -4
    0     3     4     1     2
   -1    -2     1     5     7
Macierz diagonalnie slabo dominujaca! Program kontunuuje dzialanie
Przyblizanie wartosci rozwiazan - wartosc bledu: 0.001
Roznice miedzy kolejnymi iteracjami:
             nr              x1              x2              x3              x4
             1:           0.625             0.8             0.5             1.4
             2:           0.625            0.63            0.25           0.295
             3:          0.2425           0.259         0.54625           0.427
             4:       0.0121875         0.12085          0.0875         0.04285
             5:       0.0306625        0.004055        0.079925       0.0284025
             6:      0.00679375       0.0225695      0.00405937       0.0082305
             7:      0.00874278      0.00355173       0.0189847       0.0111984
             8:     3.49062e-05      0.00505438     0.000135813     0.000627704
             9:      0.00154349     0.000112341      0.00394771      0.00205589
            10:     1.29349e-05      0.00099576     0.000429718     0.000435907
zakonczono, ilosc iteracji: 10
Wyniki:
x1 = 0.288821
x2 = -1.31275
x3 = 1.31623
x4 = 0.669899

C:\Users\Zuza\source\repos\MetodyNumeryczne_lab5\x64\Debug\MetodyNumeryczne_lab5.exe (proces 18192) zakończono z kodem 0
.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```