

Wydział WIMiIP	Imię i nazwisko Mateusz Witkowski	Rok II	Grupa 4
Temat: Rozwiązywanie układów równań metodą eliminacji Gaussa.			Prowadzący dr hab. inż. Hojny Marcin, prof. AGH
Data ćwiczenia 23.04.2020	Data oddania 29.04.2020	Data zaliczenia	OCENA

1. Cel ćwiczenia

Celem ćwiczenia było zapoznanie się oraz implementacja metody rozwiązywania układów równań – eliminacji Gaussa.

2. Wprowadzenie teoretyczne

Metoda eliminacji Gaussa służy rozwiązywaniu układów równań nieliniowych, typu $Ax = b$, gdzie A musi być macierzą kwadratową $n \times n$, a „ x ” i „ b ” wektorami o rozmiarze n . Można przedstawić to w następujący sposób:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Rozwiązujemy układ sprowadzając go do układu trójkątnego przy pomocy elementarnych operacji, takich jak:

- Pomnożenie równania przez stałą różną od zera.
- Dodanie bądź odjęcie równań.
- Zamianę równań miejscami.

Docelowy, przekształcony układ powinien wyglądać następująco:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Obliczanie rozwiązań rozpoczynamy od ostatniego równania, dzięki temu w łatwy sposób jesteśmy w stanie uzyskać jedną z szukanych wartości:

$$x_n = \frac{b_n}{a_{nn}}$$

Znając wartość x_n możemy bez problemu obliczyć wartość równania rzędu o jeden mniejszego:

$$x_{n-1} = \frac{b_{n-1} - a_{n-1} \cdot x_n}{a_{n-1n-1}}$$

Wszystkie następne równania obliczamy w analogiczny sposób, przy założeniu, że współczynniki leżące na przekątnej macierzy są niezerowe. Układ może być oznaczony (mający jedno rozwiązanie), sprzeczny (brak rozwiązań) lub nieoznaczony (nieskończenie wiele rozwiązań) zależnie o ilości niewiadomych i rzędu macierzy rozszerzonej.

3. Kod programu

Zdefiniowano globalnie wektor przechowujący dane, nazwę pliku z którego dane zostaną pobrane oraz funkcje odpowiedzialną z proces pobrania danych i zapisu ich do przekazanego wektora.

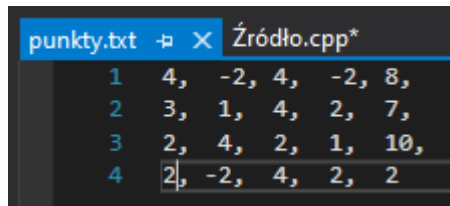
```
vector < double > dane;
string nazwaPliku = "punkty.txt";

//Funkcja pobierająca dane z pliku tekstowego i zapisujące je do wektora.
int pobranie_z_pliku(vector <double>* data, string fileName) {
    string linia;
    fstream plik;

    plik.open(fileName, ios::in);          //otwarcie pliku
    if (plik.good() == true)              //sprawdzenie poprawnosci pliku
    {
        while (!plik.eof())
        {
            getline(plik, linia, ',');      //zapisuje slowa odzielane przecinkami
            double d = atof(linia.c_str()); //konwersja stringa na double
            data->push_back(d);              //zapis slowa parsowanego na typ double do wektora
        }
        plik.close();                      //zamknięcie pliku
    }
    if (data->size() % 5 != 0) {             //zabezpieczenie w razie zle wypelnionego pliku tekstowego
        cout << "bledne dane w pliku" << endl;
        return -1;
    }
    return data->size();                    //funkcja zwraca wielkosc wektora potrzebna do stworzenia tablic.
}
```

Rysunek 1. Funkcja pobierająca dane z pliku txt

Utworzono plik txt i wypełniono go danymi.



	1	2	3	4	5	6
1	4,	-2,	4,	-2,	8,	
2	3,	1,	4,	2,	7,	
3	2,	4,	2,	1,	10,	
4	2,	-2,	4,	2,	2	

Rysunek 2. Plik przechowujący parametry układu równań

W funkcji main następuje sprawdzenie poprawności pliku, przygotowanie oraz wypełnienie dynamicznej tablicy dwuwymiarowej potrzebnej do naszej macierzy rozszerzonej, przygotowanie tablicy przeznaczonej na rozwiązanie układu.

```
int main() {
    //////////////SPRAWDZANIE POPRAWNOSCI PLIKU////////////////
    int wielkosc_wektora = pobranie_z_pliku(&dane, nazwaPliku);
    if (wielkosc_wektora == -1) //program zostanie przerwany jesli plik tekstowy by³ b³ędny
    {
        return -1;
    }

    ////////////////WYPEŁNIANIE TABLIC////////////////
    double** uk lad_rownan;
    int liczba_weirsz y = wielkosc_wektora / 5; //wielkosc wektora podzielona przez maksymalna ilość zmiennych w danym równaniu,
                                                //powinna nam dac ilość rownan zapisanych w pliku
    int dlugosc_wiersza = liczba_weirsz y + 1;
    double* Wyniki = new double[liczba_weirsz y]; //tablica rozwiązan x

    uk lad_rownan = new double* [liczba_weirsz y]; //utworzenie macierzy
    for (int i = 0; i < liczba_weirsz y; i++) {
        uk lad_rownan[i] = new double[dlugosc_wiersza]; //tworzenie wiersza
    }
    int x = 0;
    for (size_t i = 0; i < liczba_weirsz y; i++) //wypełnienie tablic
    {
        for (size_t j = 0; j < dlugosc_wiersza; j++)
        {
            uk lad_rownan[i][j] = dane[x];
            x++;
        }
    }
}
```

Rysunek 3. Utworzenie tablic, wypełnienie macierzy układu równa, sprawdzenie pliku.

Wyświetlono stan początkowy macierzy rozszerzonej oraz wywołano funkcję Gauss.

```
for (size_t i = 0; i < liczba_wierszy; i++) //wypisanie tablicy
{
    for (size_t j = 0; j < dlugosc_wiersza; j++)
    {
        if (j == liczba_wierszy) {
            cout << "| " << uklad_rownan[i][j];
        }
        else {
            cout << uklad_rownan[i][j] << setw(6);
        }
    }
    cout << endl;
}
cout << endl;

int result = Gauss(uklad_rownan, Wyniki, liczba_wierszy, dlugosc_wiersza);
```

Rysunek 4. Wyświetlenie macierzy, wywołanie funkcji.

Funkcja Gauss na początku sprowadza układ do postaci „schodkowej” (trójkątnej) poprzez obliczanie ilorazów współczynnika jednego równania przez drugie, a następnie odjęcia równań i wyzerowania współczynnika. Po przekształceniu układu sprawdzane jest czy ma on rozwiązanie, jeśli nie funkcja zwraca -1, jeśli ma nieskończenie wiele rozwiązań funkcja zwraca 0. Jeśli układ jest oznaczony obliczane są kolejne wartości **X** i zapisywane w przekazanej dynamicznej tablicy wyników. Po obliczeniu wartości funkcja zwraca 1.

```
int Gauss(double** uklad_rownan, double* Wyniki, int liczba_wierszy, int dlugosc_wiersza) {
    for (size_t i = 0; i < liczba_wierszy; i++) //sprowadzenie układu to postaci trojkatnej
    {
        for (size_t j = i+1; j < liczba_wierszy; j++)
        {
            double iloraz = uklad_rownan[j][i] / uklad_rownan[i][i]; //Obliczamy iloraz wartosci jednego wiersza przez drugi
            uklad_rownan[j][i] = 0; //zerujemy tworząc "schodki"
            for (size_t k = i+1; k < dlugosc_wiersza; k++)
            {
                uklad_rownan[j][k] = uklad_rownan[j][k] - uklad_rownan[i][k] * iloraz; //Obliczamy wartosci kolejnych wspolczynnika w wierszu
            }
        }
    }

    if (uklad_rownan[liczba_wierszy - 1][liczba_wierszy - 1] == 0 && uklad_rownan[liczba_wierszy - 1][liczba_wierszy] != 0) {
        return -1; //Funkcja zakonczy sie z wynikiem -1 jesli układ jest sprzeczny
    }
    if (uklad_rownan[liczba_wierszy - 1][liczba_wierszy - 1] == 0 && uklad_rownan[liczba_wierszy - 1][liczba_wierszy] == 0) {
        return 0; //Funkcja zakonczy sie z wynikiem 0 jesli układ jest nieoznaczony
    }

    for (int w = liczba_wierszy - 1; w > -1; w--) { //Obliczenie wartosci kolejnych X
        Wyniki[w] = uklad_rownan[w][liczba_wierszy];
        for (size_t i = w+1; i < liczba_wierszy; i++)
        {
            Wyniki[w] = Wyniki[w] - uklad_rownan[w][i] * Wyniki[i];
        }
        Wyniki[w] = Wyniki[w] / uklad_rownan[w][w];
    }
    return 1;
}
```

Rysunek 5. Funkcja realizująca eliminację Gaussa

Po wykonaniu funkcji Gauss wyświetlana jest przekształcona macierz oraz wyniki bądź odpowiedni komunikat.

```
cout << "Macierz po eliminacji Gaussa" << endl;
cout << endl;

for (size_t i = 0; i < liczba_weirzy; i++) //wypisanie tablicy
{
    for (size_t j = 0; j < dlugosc_wiersza; j++)
    {
        if (j == liczba_weirzy) {
            cout << "|" << uklad_rownan[i][j];
        }
        else {
            cout << uklad_rownan[i][j] << setw(6);
        }
    }
    cout << endl;
}

cout << endl;
if (result == 1) {
    cout << "Wyniki:" << endl;
    for (size_t i = 0; i < liczba_weirzy; i++)
    {
        cout << "X" << i + 1 << ": " << Wyniki[i] << endl;
    }
}
else if (result == 0) {
    cout << "Układ nieoznaczny" << endl;
}
else if (result == -1) {
    cout << "Układ sprzeczny" << endl;
}

getchar(); getchar();

return 0;
```

Rysunek 6. Wypisanie macierzy i wyników.

Cały kod

```
using namespace std;
vector < double > dane;
string nazwaPliku = "punkty.txt";

//Funkcja pobierająca dane z pliku tekstowego i zapisujące je do wektora.
int pobranie_z_pliku(vector <double>* data, string fileName) {
    string linia;
    fstream plik;

    plik.open(fileName, ios::in);           //otwarcie pliku
    if (plik.good() == true)               //sprawdzenie poprawności pliku
    {
        while (!plik.eof())
        {
            getline(plik, linia, ',');      //zapisuje słowa oddzielane przecinkami
            double d = atof(linia.c_str()); //konwersja stringa na double
            data->push_back(d);              //zapis słowa parsowanego na typ double do wektora
        }
        plik.close();                      //zamknięcie pliku
    }
    if (data->size() % 5 != 0) {             //zabezpieczenie w razie złe wypełnionego pliku tekstowego
        cout << "błędne dane w pliku" << endl;
        return -1;
    }
    return data->size();                    //funkcja zwraca wielkość wektora potrzebna do stworzenia tablic.
}

int Gauss(double** uk lad_rownan, double* Wyniki, int liczba_wierszy, int dlugosc_wiersza) {
    for (size_t i = 0; i < liczba_wierszy; i++) //sprowadzenie układu to postaci trójkątnej
    {
        for (size_t j = i+1; j < liczba_wierszy; j++)
        {
            double iloraz = uk lad_rownan[j][i] / uk lad_rownan[i][i]; //Obliczamy iloraz wartości jednego wiersza przez drugi
            uk lad_rownan[j][i] = 0; //zerujemy tworząc "schodki"
            for (size_t k = i+1; k < dlugosc_wiersza; k++)
            {
                uk lad_rownan[j][k] = uk lad_rownan[j][k] - uk lad_rownan[i][k] * iloraz; //Obliczamy wartości kolejnych współczynników w wierszu
            }
        }
    }

    if (uk lad_rownan[liczba_wierszy - 1][liczba_wierszy - 1] == 0 && uk lad_rownan[liczba_wierszy - 1][liczba_wierszy] != 0) {
        return -1; //Funkcja zakończy się z wynikiem -1 jeśli układ jest sprzeczny
    }

    if (uk lad_rownan[liczba_wierszy - 1][liczba_wierszy - 1] == 0 && uk lad_rownan[liczba_wierszy - 1][liczba_wierszy] == 0) {
        return 0; //Funkcja zakończy się z wynikiem 0 jeśli układ jest nieoznaczony
    }

    for (int w = liczba_wierszy - 1; w > -1; w--) { //Obliczenie wartości kolejnych X
        Wyniki[w] = uk lad_rownan[w][liczba_wierszy];
        for (size_t i = w+1; i < liczba_wierszy; i++)
        {
            Wyniki[w] = Wyniki[w] - uk lad_rownan[w][i] * Wyniki[i];
        }
        Wyniki[w] = Wyniki[w] / uk lad_rownan[w][w];
    }

    return 1;
}
```

```

int main() {
    ////////////SPRAWDZANIE POPRAWNOSCI PLIKU//////////
    int wielkosc_wektora = pobranie_z_pliku(&dane, nazwaPliku);
    if (wielkosc_wektora == -1) //program zostanie przerwany jesli plik tekstowy by³ b³edny
    {
        return -1;
    }
    ////////////WYPEŁNIANIE TABLIC//////////
    double** uk lad_rownan;
    int liczba_weirzy = wielkosc_wektora / 5; //wielkosc wektora podzielona przez maksymalna iloœc zmiennych w danym rownaniu,
                                                //powinna nam dac iloœc rownan zapisanych w pliku
    int dlugosc_wiersza = liczba_weirzy + 1;
    double* Wyniki = new double[liczba_weirzy]; //tablica rozwiazañ x

    uk lad_rownan = new double* [liczba_weirzy]; //utworzenie macierzy
    for (int i = 0; i < liczba_weirzy; i++) {
        uk lad_rownan[i] = new double[dlugosc_wiersza]; //tworzenie wiersza
    }
    int x = 0;
    for (size_t i = 0; i < liczba_weirzy; i++) //wypelnienie tablic
    {
        for (size_t j = 0; j < dlugosc_wiersza; j++)
        {
            uk lad_rownan[i][j] = dane[x];
            x++;
        }
    }
    for (size_t i = 0; i < liczba_weirzy; i++) //wypisanie tablicy
    {
        for (size_t j = 0; j < dlugosc_wiersza; j++)
        {
            if (j == liczba_weirzy) {
                cout << "| " << uk lad_rownan[i][j];
            }
            else {
                cout << uk lad_rownan[i][j] << setw(6);
            }
        }
        cout << endl;
    }
    cout << endl;

    int result = Gauss(uk lad_rownan, Wyniki, liczba_weirzy, dlugosc_wiersza);

    cout << "Macierz po eliminacji Gaussa" << endl;
    cout << endl;

    for (size_t i = 0; i < liczba_weirzy; i++) //wypisanie tablicy
    {
        for (size_t j = 0; j < dlugosc_wiersza; j++)
        {
            if (j == liczba_weirzy) {
                cout << "| " << uk lad_rownan[i][j];
            }
            else {
                cout << uk lad_rownan[i][j] << setw(6);
            }
        }
        cout << endl;
    }
    cout << endl;
}

```

```

    if (result == 1) {
        cout << "Wyniki:" << endl;
        for (size_t i = 0; i < liczba_weirszzy; i++)
        {
            cout << "X" << i + 1 << ": " << Wyniki[i] << endl;
        }
    }

    else if (result == 0) {
        cout << "Układ nieoznaczny" << endl;
    }

    else if (result == -1) {
        cout << "Układ sprzeczny" << endl;
    }

    getchar(); getchar();
    return 0;
}

```

4. Testy

W celu zweryfikowania wyników programu dokonano czterech testów z wykorzystaniem kalkulatora znajdującego się na stronie https://calcoolator.pl/metoda_gaussa.html#solve-using-Gaussian-elimination%28%7B%7B4,-2,4,-2,8%7D,%7B3,1,4,2,7%7D,%7B2,4,2,1,10%7D,%7B2,-2,4,2,2%7D%7D%29.

Przykład z instrukcji:

```

C:\> D:\STUDIA\IV_Semestr\Metody\zajecia8\eliminacjaGaussa\Debug\eliminacjaGaussa.exe
4   -2   4   -2   |   8
3    1   4    2   |   7
2    4   2    1  |  10
2   -2   4    2   |   2

Macierz po eliminacji Gaussa
4   -2   4   -2   |   8
0   2.5  1   3.5  |   1
0    0  -2   -5   |   4
0    0   0  -1.6  |  3.2

Wyniki:
X1: -1
X2: 2
X3: 3
X4: -2

```

Rysunek 7. Wynik działania programu dla przykładu z instrukcji.

Wynik ze strony:

$$\begin{cases} 4 \cdot x_1 - 2 \cdot x_2 + 4 \cdot x_3 - 2 \cdot x_4 = 8 \\ \frac{5}{2} \cdot x_2 + x_3 + \frac{7}{2} \cdot x_4 = 1 \\ -2 \cdot x_3 - 5 \cdot x_4 = 4 \\ -\frac{8}{5} \cdot x_4 = \frac{16}{5} \end{cases} \quad \begin{matrix} \text{Wynik:} \\ x_1 = -1, \\ x_2 = 2, \\ x_3 = 3, \\ x_4 = -2 \end{matrix}$$

Test 1:

```

c:\ D:\STUDIA\IV_Semestr\Metody\zajecia8\eliminacjaGausa\Debug\eliminacjaGausa.exe
1      2      -1      |      5
3      4      1      |      9
2     -2      3      |     -1

Macierz po eliminacji Gaussa

1      2      -1      |      5
0     -2      4      |     -6
0      0     -7      |      7

Wyniki:
X1: 2
X2: 1
X3: -1

```

Rysunek 8. Wynik programu dla macierz 3x3

Wynik ze strony:

$$\begin{cases} x_1 + 2 \cdot x_2 - x_3 = 5 \\ -2 \cdot x_2 + 4 \cdot x_3 = -6 \\ -7 \cdot x_3 = 7 \end{cases} \quad \begin{matrix} \text{Wynik:} \\ x_1 = 2, \\ x_2 = 1, \\ x_3 = -1 \end{matrix}$$

Test 2:

```

D:\STUDIA\IV_Semestr\Metody\zajecia8\eliminacjaGausa\Debug\eliminacjaGausa.exe

2    1    1    1    |  0
-1   -2   -1    1    |  1
1    1   -2   -3    |  2
4   -2    0    4    |  2

Macierz po eliminacji Gaussa

2    1    1    1    |  0
0  -1.5  -0.5   1.5   |  1
0    0 -2.66667   -3    |  2.33333
0    0    0  -1.25   | -1.25

Wyniki:
X1: 0
X2: 1
X3: -2
X4: 1

```

Rysunek 9. Wynik programu dla macierz 4x4

Wynik ze strony:

$$\begin{cases} 2 \bullet x_1 + x_2 + x_3 + x_4 = 0 \\ -\frac{3}{2} \bullet x_2 - \frac{1}{2} \bullet x_3 + \frac{3}{2} \bullet x_4 = 1 \\ -\frac{8}{3} \bullet x_3 - 3 \bullet x_4 = \frac{7}{3} \text{ (1)} \\ -\frac{5}{4} \bullet x_4 = -\frac{5}{4} \end{cases}$$

▼

Wynik:

$x_1 = 0$,
 $x_2 = 1$,
 $x_3 = -2$,
 $x_4 = 1$

Test 3:

```

D:\STUDIA\IV_Semestr\Metody\zajecia8\eliminacjaGausa\Debug\eliminacjaGausa.exe

5  2  6  6 -7 | 8
4 -4  7  2  1 | 16
5  9 -3  1  5 | 3
-3  4  7  9 -3 | 5
6  1  6 -4  2 | 7

Macierz po eliminacji Gaussa

5  2  6  6 -7 | 8
0 -5.6 2.2 -2.8 6.6 | 9.6
0  0 -6.25 -8.5 20.25 | 7
0  0  0 -7.1942939.8914 | 32.8743
0  0  0  0 -41.9444 | -44.0643

Wyniki:
X1: 1.22374
X2: -0.877632
X3: 0.576106
X4: 1.25562
X5: 1.05054

```

Rysunek 10. Wynik programu dla macierz 5x5

Wynik ze strony:

$$\begin{cases}
 5 \cdot x_1 + 2 \cdot x_2 + 6 \cdot x_3 + 6 \cdot x_4 - 7 \cdot x_5 = 8 \\
 -\frac{28}{5} \cdot x_2 + \frac{11}{5} \cdot x_3 - \frac{14}{5} \cdot x_4 + \frac{33}{5} \cdot x_5 = \frac{48}{5} \\
 -\frac{25}{4} \cdot x_3 - \frac{17}{2} \cdot x_4 + \frac{81}{4} \cdot x_5 = 7 \quad (1) \\
 -\frac{1259}{175} \cdot x_4 + \frac{6981}{175} \cdot x_5 = \frac{5753}{175} \\
 -\frac{52808}{1259} \cdot x_5 = \frac{-55477}{1259}
 \end{cases}$$

Wynik:

$$\begin{aligned}
 x_1 &= \frac{64623}{52808}, \\
 x_2 &= \frac{-23173}{26404}, \\
 x_3 &= \frac{30423}{52808}, \\
 x_4 &= \frac{66307}{52808}, \\
 x_5 &= \frac{55477}{52808}
 \end{aligned}$$

Tabela 1. Wyniki kalkulatora ze strony przeliczone na ułamki dziesiętne

X1	X2	X3	X4	X5
1,223735	-0,877632	0,5761058	1,25562	1,05054

Test 4:

```
C:\ D:\STUDIA\IV_Semestr\Metody\zajecia8\eliminacjaGaussa\Debug\eliminacjaGaussa.exe
5   -2   -1   |   1
-2    2   -2   |   2
-1   -2    5   |   1

Macierz po eliminacji Gaussa

5   -2   -1   |   1
0   1.2 -2.4   |  2.4
0    0    0   |   6

Układ sprzeczny
```

Rysunek 11. Wynik programu dla układu sprzecznego

Wynik ze strony:

$$\begin{cases} 5 \cdot x_1 - 2 \cdot x_2 - x_3 = 1 \\ \frac{6}{5} \cdot x_2 - \frac{12}{5} \cdot x_3 = \frac{12}{5} \quad (1) \\ 0 = 6 \end{cases}$$

brak rozwiązania!

5. Wnioski

W metodzie eliminacji Gaussa nie trzeba liczyć wyznaczników macierzy tak jak miało to miejsce w przypadku metody Cramera, dlatego ta metoda sprawdza się znacznie lepiej w przypadku układów równań o znacznej liczbie niewiadomych. Jest efektywna i bardzo dokładna co pokazały testy. Każdy test potwierdził prawidłowość wyników oraz pokazał czy program jest w stanie wykryć czy układ jest sprzeczny bądź nieoznaczony. Program bez problemu radzi sobie z układami 3x3, 4x4 i większymi zachowując wysoki stopień dokładności.