

Grupa lab.3	Data wykonania 15.03.2024	Inżynieria Obliczeniowa 2023/2024
Temat ćwiczenia Metoda eliminacji Gaussa		
Imię i nazwisko Karolina Kurowska		Ocena i uwagi

Wstęp

Metoda eliminacji Gaussa to technika rozwiązywania układów równań liniowych poprzez stopniowe eliminowanie niewiadomych. Proces ten polega na przekształcaniu układu równań tak, aby w każdym kroku jedna z niewiadomych została wyeliminowana. Metoda ta wykorzystuje operacje elementarne na wierszach macierzy, takie jak dodawanie, odejmowanie i mnożenie przez stałą, aby sprowadzić macierz do postaci trójkątnej górnej, co ułatwia znalezienie rozwiązania.

Przebieg ćwiczenia

Na początku wprowadzamy dane z pliku tekstowego do dwuwymiarowej tablicy o nazwie `el_gausso`. Następnie przypisujemy zmiennej `quantity` ilość równań, która jest umieszczona na początku tego pliku.

```
int quantity;
ifstream plik;
//plik.open ("RURL_dane1.txt", ios::in); //opcja 1
plik.open ("RURL_dane2.txt", ios::in); //opcja 2
if (plik.good () == false) {
    cout << "Plik nie istnieje\n";
    exit (0);
}
plik >> quantity;
cout << "Ilosc rownan: " << quantity << endl;

//stworzenie macierzy
double** el_gausso = new double* [quantity];
for (int i = 0; i < quantity; i++) {
    el_gausso[i] = new double[quantity + 1];
}

for (int i = 0; i < quantity; i++) {
    for (int j = 0; j < quantity + 1; j++) {
        plik >> el_gausso[i][j];
    }
}
plik.close ();
```

Metoda składa się z dwóch głównych etapów: postępowania prostego, nazywanego również etapem eliminacji, oraz postępowania odwrotnego. Pierwszy etap polega na przekształceniu układu równań do postaci górnie trójkątnej poprzez odejmowanie od każdego kolejnego wiersza odpowiednio przekształconego wiersza zerowego.

Aby to osiągnąć, podzieliłam mój kod na kilka mniejszych funkcji, z których każda odpowiada odpowiednim wzorom. Na początku działa funkcja *gaussZero*, która musi zostać wykonana o jeden raz mniej niż liczba równań, które posiadamy.

```
wypisz_macierz (quantity, el_gausso);
//postępowanie proste
for (int i = 0; i < (quantity - 1); i++) { //tyle co kolumn, i -> kolumny
    gaussZero (el_gausso, quantity, i);
}
wypisz_macierz (quantity, el_gausso);
```

Wewnątrz tej funkcji należy sprawdzić, czy podczas przekształcania macierzy na macierz trójkątną górną nie ma już zer, do których dążymy, a których ponowne obliczanie spowodowałoby dzielenie przez zero. Następnie obliczamy mnożnik ze wzoru:

$$m_{i0} = \frac{a_{i0}}{a_{00}}.$$

Kiedy już znamy nasz mnożnik, wewnątrz funkcji *odejmij_wiersz* odejmujemy pierwszy wiersz pomnożony przez obliczoną liczbę od kolejnego wiersza. W ten sposób stopniowo uzyskujemy macierz schodkową.

```
void odejmij_wiersz(double ** table, int quantity, int i, int j, double mnoz){
    for (int k = 0; k <= (quantity+1); k++) { //do odejmowania w kazdej kolumnie
        table[j][i+k] = table[j][i+k] - (table[i][i+k] * mnoz);
    }
    // wypisz_macierz (quantity, table);
}

void gaussZero(double** table,int quantity, int i) {
    for (int j = i + 1; j < quantity; j++) { //tyle co wierszy, j -> wiersze
        if (table[i][i] == 0) {
            cout << "W wierszu " << j+1 << ". kolumnie " << i << " jest 0, nie trzeba nic zmieniac\n\n";
            return;
        }
        double mnoz = (table[j][i] / table[i][i]);
        // cout << "\nMnoznik wiersz " << j << ". kolumna " << i << ". -> " << mnoz << endl;
        odejmij_wiersz(table ,quantity, i, j, mnoz);
    }
}
```

Wyniki obliczeń:

	Przed	Po
P r z y k ł a d 1	<p>Ilość równań: 4</p> <pre>2 4 2 1 10 2 2 3 3 6 4 2 2 1 6 0 2 1 1 4</pre>	<pre>2 4 2 1 10 0 -2 1 2 -4 0 0 -5 -7 -2 0 0 0 0.2 -0.8</pre>

Przykład 2	<pre> Ilosc rownan: 6 1 1 -2 1 -2 -5 8 2 -4 -1 2 3 3 1 2 -2 6 -1 6 5 5 0 2 1 1 4 5 5 -5 0 4 -1 9 4 10 7 -2 -4 5 3 -1 -5 </pre>	<pre> 1 1 -2 1 -2 -5 8 0 -6 3 0 7 13 -15 0 0 8 -3 5.33333 6.33333 -1 0 0 0 1.75 5 7.75 0.25 0 0 0 0 -0.511905 -19.2976 36.6786 0 0 0 0 0 -90.2093 152.395 </pre>

Następnie możemy przejść do drugiego etapu, czyli postępowania odwrotnego. Do funkcji `policz_rozwiazania` przekazujemy naszą macierz, jej rozmiar oraz nowo utworzoną tablicę wyników, która będzie przechowywać rozwiązania naszego układu równań. Funkcja działa na zasadzie wzoru:

$$x_i = \frac{b_i}{a_{ii}} - \frac{\sum_{k=i+1}^n a_{ik} x_k}{a_{ii}}, \text{ dla } i = n-1, \dots, 0$$

```

void policz_rozwiazania(int quantity, double** table, double* wyniki) {
    for (int l = quantity - 1; l >= 0; l--) {
        wyniki[l] = table[l][quantity] / table[l][l]; //b_n/a_nn
        for (int k = l + 1; k < quantity; k++) {
            wyniki[l] -= (table[l][k] * wyniki[k]) / table[l][l]; //reszta wzoru
        }
    }
    for (int i = 0; i < quantity; i++) { //wypisanie rozwiazań
        cout << "Wynik rownania " << i + 1 << ". to " << wyniki[i] << endl;
    }
}

```

Rozwiązania układów równań:

Przykład I	Przykład II
<pre> Wynik rownania 1. to -1 Wynik rownania 2. to 1 Wynik rownania 3. to 6 Wynik rownania 4. to -4 </pre>	<pre> Wynik rownania 1. to -9.43387 Wynik rownania 2. to -1.49549 Wynik rownania 3. to 17.918 Wynik rownania 4. to 30.3857 Wynik rownania 5. to -7.96649 Wynik rownania 6. to -1.68935 </pre>

Wnioski

Metoda eliminacji Gaussa jest potężnym narzędziem, które umożliwia przekształcenie skomplikowanych równań na bardziej zrozumiałe poprzez zastosowanie prostych operacji,

takich jak dodawanie jednego równania do innego lub mnożenie równań przez stałą. Jej celem jest sprowadzenie macierzy równań do postaci, w której łatwo można znaleźć rozwiązanie. Ta technika jest powszechnie stosowana w różnych dziedzinach, takich jak elektryka, finanse, statystyka i inżynieria. Znajduje zastosowanie m.in. przy analizie obwodów elektrycznych, prognozowaniu zachowań finansowych, analizie danych oraz rozwiązywaniu problemów optymalizacyjnych. Dzięki niej jesteśmy w stanie skutecznie radzić sobie z problemami opartymi na równaniach, które odgrywają kluczową rolę w wielu praktycznych zastosowaniach.