Grupa lab.3	Data wykonania 22.03.2024	Inżynieria Obliczeniowa 2023/2024
Temat ćwiczenia **Metoda eliminacji Gaussa - pivoting**		
Imię i nazwisko Karolina Kurowska		Ocena i uwagi

Wstęp

Metoda eliminacji Gaussa jest wykorzystywana do rozwiązywania układów równań liniowych poprzez eliminację zmiennych. W przypadku wyboru częściowego, znanej również jako partial pivoting, polega ona na przestawieniu kolejności wierszy w macierzy tak, aby element znajdujący się na diagonali, który jest dzielony podczas eliminacji, był największym elementem w danej kolumnie. Natomiast metoda Crouta stanowi ulepszenie metody eliminacji Gaussa. W tej metodzie, poszukujemy elementu o największym module w danym wierszu macierzy AB i zamieniamy go miejscami z kolumną zawierającą element głównej przekątnej. Dzięki temu unikamy sytuacji, w której dzielnik może przyjąć wartość równą zero. Algorytm Crouta wymaga zapamiętania kolumn, które zostały zamienione miejscami.

Przebieg ćwiczenia

Na początku wprowadzamy dane z pliku tekstowego do dwuwymiarowej tablicy o nazwie *el_gausso*. Przypisujemy zmiennej *quantity* ilość równań, która jest umieszczona na początku tego pliku. Dodatkowo w tych laboratoriach uzupełniamy domyślnymi wartościami wektor *kolejność_kolumn*, którego używać będziemy do zadania drugiego.

```
int quantity;
ifstream plik;
//plik.open ("RURL_dane1.txt", ios::in);//opcja 1
plik.open ("RURL_dane2.txt", ios::in); //opcja 2
//plik.open ("RURL_dane3.txt", ios::in); //opcja 3
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];
    kolejnosc_kolumn.push_back (i);
}

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

Zadanie 1

Jako pierwsze zadanie mieliśmy do napisania funkcję do rozwiązywania układu równań z wyborem częściowym (zmiana po wierszach).

Jako pierwsze włączamy funkcję *ustaw_najwiekszy_wiersze*. Jej zadaniem, jest znalezienie największej wartości w kolumnie i ustawienie jej na przekątnej macierzy. Jeśli największa wartość nie znajduje się na macierzy, przestawi ona wszystkie wartości wiersza, w której maximum zostało znalezione. Na koniec ustawienia każdej kolumny włącza się funkcja *gaussZero* która zeruje wartości pod przekątną.

```
//ZADANIE I
ustaw_najwiekszy_wiersze (el_gausso, quantity);

cout << "Po zamianie: " << endl;
wypisz_macierz (quantity, el_gausso);

double* wyniki = new double[quantity];
policz_rozwiazania (quantity, el_gausso, wyniki);
```

```
void ustaw_najwiekszy_wiersze (double** table, int quantity){ //i to indeks kolumny
    for (int i = 0; i < (quantity - 1); i++) {
        int line = i;
        double value = table[i][i];

        for (int j = i + 1; j < quantity; j++) {
            if (abs (value) < abs (table[j][i])) {
                value = table[j][i];
                line = j;
            }
        }
        //cout << "Najwieksza wartosc w kolumnie " << i << " i wierszu " << line << " to: " << value << endl;

        for (int k = 0; k <= quantity + 1; k++) {
            double temp = table[i][k];
            table[i][k] = table[line][k];
            table[line][k] = temp;
        }
        //zerownie kolumn
        gaussZero (table, quantity, i);
    }
}
```

```
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\n\n";
            return;
        }

        double mnoz = (table[j][i] / table[i][i]);
        // cout << "\nMnoznik wiersz " << j << ". kolmna "<< i << ".  -> " << mnoz << endl;
        odejmij_wiersz (table, quantity, i, j, mnoz);
    }
}
```

Po zamianie całej macierzy na schodkową program włącza funkcję policz rozwiązania, która działa jak w poprzednich laboratoriach korzystając z odpowiednio zaimplementowanych

wzorów. Na koniec wypisuje w konsoli odpowiednią kolejność rozwiązań według wektora kolejność_kolumn. Mimo że w tym zadaniu będzie to kolejność domyślna, to rozbudowanie funkcji o dodatkowe pętle i warunek przyda się przy wypisywaniu odpowiednich rozwiązań w zadaniu 2.

```
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 roziwazań
        for (int j = 0; j < quantity; j++){
            if (kolejnosc_kolumn[j] == i) {
                cout << "Wynik rownan to x" << i << " = " << wyniki[j] << endl;
            }
        }
    }
}
```

Wyniki dla pliku RURL_dane1	*Wyniki dla pliku RURL_dane3*
Ilosc rownan: 4 Przed zamiana: 2 4 2 1 10 2 2 3 3 6 4 2 2 1 6 0 2 1 1 4 Po zamianie: 4 2 2 1 6 0 3 1 0.5 7 0 0 1.66667 2.33333 0.666667 0 0 0 0.2 -0.8 Wynik rownan to x0 = -1 Wynik rownan to x1 = 1 Wynik rownan to x2 = 6 Wynik rownan to x3 = -4	Ilosc rownan: 4 Przed zamiana: 2 4 2 1 10 1 2 3 3 6 4 5 2 1 6 0 1 2 9 1 Po zamianie: 4 5 2 1 6 0 1.5 1 0.5 7 0 0 2 2.5 1 0 0 0 7 -4.33333 Wynik rownan to x0 = -4.0119 Wynik rownan to x1 = 4.02381 Wynik rownan to x2 = 1.27381 Wynik rownan to x3 = -0.619048

Zadanie 2

Następnym zadaniem było napisanie funkcji do rozwiązywania układu równań liniowych z wyborem częściowym (zmiana po kolumnach - Eliminacja Gaussa-Crouta).

W tym celu zaimplementowałam funkcję *ustaw_najwiekszy_kolumny*, która w przeciwieństwie do wcześniejszego zadania ustawia największą wartość zamieniając ze sobą kolumny, nie wiersze. Nie zmienia to jednak faktu, że działa analogicznie. Kluczową różnicą miedzy, nimi jest zapisanie zmian przeniesienia kolumn w wektorze kolejnosc_kolumn.

Na końcu włączamy tą samą funkcję policz_rozwiazania, która, jak wyżej wyjaśniłam, została zmodyfikowana do odczytywania kolejności rozwiązań przy zmienie kolejności kolumn.

```cpp
//ZADANIE II
ustaw_najwiekszy_kolumny (el_gausso, quantity);

cout << "Po zamianie: " << endl;
wypisz_macierz (quantity, el_gausso);

double* wyniki = new double[quantity];
policz_rozwiazania (quantity, el_gausso, wyniki);
```

```cpp
void ustaw_najwiekszy_kolumny (double** table, int quantity) {
    for (int i = 0; i < quantity - 1; i++) { //to do kazdego wiersza
        double value = abs(table[i][i]);
        int column = i;

        for (int j = i + 1; j < quantity; j++) { //to do kazdej kolumny
            if (abs(table[i][j]) > value) {
                value = table[i][j];
                column = j;
            }
        }
       // cout << "Najwieksza wartosc w kolumnie " << column << " i wierszu " << i << " to: " << value << endl;

        if (column != i) {
            for (int k = 0; k < quantity; k++) { // zamień całą kolumnę
                double temp = table[k][i];
                table[k][i] = table[k][column];
                table[k][column] = temp;
            }

            int temp2 = kolejnosc_kolumn[i];
            kolejnosc_kolumn[i] = kolejnosc_kolumn[column];
            kolejnosc_kolumn[column] = temp2;
        }
        gaussZero (table, quantity, i);
    }
}
```

Wyniki dla RURL_dane3

```
Ilosc rownan: 6
Przed zamiana:
 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

Po zamianie:
 -5 1 -2 -2 1 1 8
 0 -3.4 -2.2 1.8 2.6 2.6 5.8
 0 0 4.64706 3.47059 2.23529 -0.764706 11.2941
 0 0 0 5.78481 4.70886 3.81013 25.2658
 0 0 2.22045e-16 0 -9.71772 -3.50547 -14.8403
 0 0 0 0 0 -0.873452 -26.5404

Wynik rownan to x0 = -9.43387
Wynik rownan to x1 = -1.49549
Wynik rownan to x2 = 17.918
Wynik rownan to x3 = 30.3857
Wynik rownan to x4 = -7.96649
Wynik rownan to x5 = -1.68935
```

Wnioski

Metody eliminacji Gaussa i Gaussa-Crouta są ważnymi narzędziami w rozwiązywaniu równań liniowych, które oferują równowagę pomiędzy prostotą implementacji, efektywnością obliczeniową i stabilnością numeryczną. Są one stosowane w różnorodnych dziedzinach,

takich jak inżynieria, fizyka, ekonomia czy informatyka, do rozwiązywania problemów związanych z analizą danych, symulacjami numerycznymi, optymalizacją i wieloma innymi aplikacjami, gdzie występują równania liniowe.