Grupa lab.3	Data wykonania 19.04.2024	Inżynieria Obliczeniowa 2023/2024
Temat ćwiczenia **Rozwiązywanie układów równań liniowych - Metoda Jacobiego**		
Imię i nazwisko Karolina Kurowska		Ocena i uwagi

Wstęp

Metoda Jacobiego jest jedną z iteracyjnych metod numerycznych służących do rozwiązywania układów równań liniowych. Znajduje zastosowanie w przypadkach, gdy układ równań jest zbyt złożony, aby rozwiązać go metodami dokładnymi, takimi jak metoda eliminacji Gaussa. Jest to szczególnie przydatne w przypadku dużych i rzadkich układów równań, często spotykanych w praktycznych problemach inżynierskich i naukowych.

Istotą metody Jacobiego jest iteracyjne przybliżanie rozwiązania układu równań poprzez wielokrotne podstawianie przybliżonych wartości niewiadomych do równań. Proces ten jest powtarzany aż do osiągnięcia zadowalającej zbieżności, czyli gdy różnica między kolejnymi przybliżeniami staje się mniejsza od ustalonej tolerancji.

Przebieg ćwiczenia

Zadanie 1

➔ Wczytywanie danych z pliku:

```
int quantity;
ifstream plik;
plik.open ("RURL_Jacobi_dane1.txt", ios::in);
if (plik.good () == false) {
    cout << "Plik nie istnieje\n";
    exit (0);
}
plik >> quantity;
cout << "Ilosc rownan: " << quantity << endl;

//stworzenie macierzy
double** matrix_A = new double* [quantity];
double** matrix_D = new double* [quantity];
double** matrix_LU = new double* [quantity];
double* matrix_B = new double[quantity];
double* matrix_X = new double[quantity];

for (int i = 0; i < quantity; i++) {
    matrix_A[i] = new double[quantity + 1];
    matrix_D[i] = new double[quantity];
    matrix_LU[i] = new double[quantity];
    matrix_X[i] = 0;
}

for (int i = 0; i < quantity; i++) {
    for (int j = 0; j < quantity + 1; j++) {
        plik >> matrix_A[i][j];
        if (j == quantity) {
            matrix_B[i] = matrix_A[i][j];
        }
    }
}
for (int i = 0; i < quantity; i++) {
    for (int j = 0; j < quantity; j++) {
        matrix_D[i][j] = 0;
        matrix_LU[i][j] = 0;
    }
}
plik.close ();
```

➔ Funkcja sprawdzająca, czy macierz jest diagonalnie słabo dominująca implementuje warunki w postaci wzorów:

$$|a_{ii}| \geq \sum_{j=0,\, j\neq i}^{n} |a_{ij}|,$$

oraz spełnienie warunku dla conajmniej jednego i:

$$|a_{ii}| > \sum_{j=0,\, j\neq i}^{n} |a_{ij}|$$

Warunkiem zbieżności metody Jacobiego jest diagonalna słaba dominacja macierzy współczynników układu równań. Oznacza to, że wartość bezwzględna elementu na głównej przekątnej musi być większa lub równa sumie wartości bezwzględnych pozostałych elementów w danym wierszu, oraz większa dla co najmniej jednego wiersza.

```
bool macierz_slabodominujaca (double** A, int quantity) {
    int counter = 0;
    for (int i = 0; i < quantity; i++) {
        double non_diagonal_sum = 0;

        for (int j = 0; j < quantity; j++) {
            if (j != i) {
                non_diagonal_sum += A[i][j];
            }
        }

        if (A[i][i] = non_diagonal_sum) {
            counter++;
        }
        if (A[i][i] < non_diagonal_sum) {
            cout << "Macierz NIE JEST diagonalnie slabo dominujaca\n" << endl;
            return false;
        }
    }

    if (counter > 0) {
        cout << "Macierz JEST diagonalnie slabo dominujaca\n" << endl;
        return true;
    }
    return false;
}
```

```
bool verification = macierz_slabodominujaca (matrix_A, quantity);
if (verification = false) {
    return 0;
}
```

➔ Implementacja metody Jacobiego:
Do tego należało skorzystać z wzoru:

$$x^{(i+1)} = -D^{-1}(L + U)x^{(i)} + D^{-1}b$$

```
else {
    //zadanie 1
    int iteracje = 0;
    cout << "Wybierz ilosc iteracji: ";
    cin >> iteracje;
    cout << endl;
    policz_ciag_przyblizen (quantity, iteracje, matrix_LU, matrix_D, matrix_X, matrix_B);
```

```
void policz_ciag_przyblizen (int quantity, int iteration, double ** LU, double** D, double* X, double* B) {
    for (int k = 0; k < iteration; k++) {
        double* tempX = new double[quantity];
        for (int i = 0; i < quantity; i++) {
            double LUX = 0;
            for (int j = 0; j < quantity; j++) {
                if (i != j) {
                    LUX += LU[i][j] * X[j]; //LU * X to macierz 4x1
                }
            }
            tempX[i] = (B[i] - LUX) * D[i][i];
        }

        for (int i = 0; i < quantity; i++) {
            X[i] = tempX[i];
        }
    }

    cout << "Rozwiazanie po " << iteration << " iteracjach: " << endl;
    for (int i = 0; i < quantity; i++) {
        cout << "x[" << i << "] = " << X[i] << endl;
    }
}
```

➔ Końcowe rozwiązanie układu równań po zadanej liczbie iteracji:

```
Ilosc rownan: 4
Macierz glowna A:
 8 2 2 4 5
 2 5 1 1 -4
 0 3 4 1 2
 -1 -2 1 5 7

Macierz L + U:
 0 2 2 4
 2 0 1 1
 0 3 0 1
 -1 -2 1 0

Macierz Diagonalna * -1:
 0.125 0 0 0
 0 0.2 0 0
 0 0 0.25 0
 0 0 0 0.2

Macierz JEST diagonalnie slabo dominujaca

Wybierz ilosc iteracji: 5

Rozwiazanie po 5 iteracjach:
x[0] = 0.28535
x[1] = -1.2878
x[2] = 1.28868
x[3] = 0.692447

C:\Users\kkuro\source\repos\lab06_met_num\x64\Debug\lab06_met_num.exe (proces 29348) zakończono z kodem 0.
```

➔ Porównanie wyników uzyskanych metodą Jacobiego i Gaussa.

◆ Wyniki dla tych samych danych uzyskane za pomocą funckji z laboratoriów 4

```
Ilosc rownan: 4
Przed zamiana:
 8 2 2 4 5
 2 5 1 1 -4
 0 3 4 1 2
 -1 -2 1 5 7

Po zamianie:
 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

Wynik rownan to x0 = 0.289318
Wynik rownan to x1 = -1.31306
Wynik rownan to x2 = 1.31751
Wynik rownan to x3 = 0.669139

C:\Users\kkuro\source\repos\lab04_met_num\x64\Debug\lab04_met_num.exe (proces 25504) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Zasadnicze różnice w wynikach to:

- Metoda Jacobiego da przybliżone rozwiązanie, którego dokładność zależy od liczby wykonanych iteracji. Metoda Gaussa da dokładne rozwiązanie.
- Wyniki metody Jacobiego mogą się nieznacznie różnić w zależności od wyboru przybliżenia początkowego. Metoda Gaussa da zawsze ten sam wynik.
- W przypadku źle uwarunkowanych układów, metoda Jacobiego może mieć problemy ze zbieżnością. Metoda Gaussa jest pod tym względem stabilniejsza.
- Jeśli układ ma nieskończenie wiele rozwiązań lub jest sprzeczny, metoda Jacobiego może nie wykryć tego. Metoda Gaussa pozwoli to stwierdzić.

Podsumowując, wyniki uzyskane metodą eliminacji Gaussa będą dokładniejsze i pewniejsze. Metoda Jacobiego może być przydatna gdy wystarczy przybliżone rozwiązanie, ale należy mieć świadomość jej ograniczeń.

Zadanie 2

➔ Zadaniem był zaimplementowanie waruneku stopu w postaci:

$$|x^{(i+1)} - x^{(i)}| < \varepsilon$$

Ustalam maksymalną liczbę iteracji, aby zapobiec nieskończonej pętli w przypadku braku zbieżności. Jeśli warunek stopu nie zostanie spełniony w zadanej liczbie iteracji, program zakończy działanie.

```cpp
void policz_ciag_przyblizen_epsilon (int quantity, double epsilon, double** LU, double** D, double* X, double* B) {
    double* tempX = new double[quantity];
    bool converged = false; //zbieżny
    int iteration = 0;
    int max_iterations = 10000;
    double temp_epsilon = 0;

    for (int k = 0; !converged && k < max_iterations; k++) { //dpouki wynik nie będzie zbiezny
        converged = true;

        for (int i = 0; i < quantity; i++) {
            double LUX = 0;
            for (int j = 0; j < quantity; j++) {
                if (i != j) {
                    LUX += LU[i][j] * X[j];
                }
            }
            tempX[i] = (B[i] - LUX) * D[i][i];
            //sprawdzanie warunku epsilona
            temp_epsilon = fabs (tempX[i] - X[i]);
            if (temp_epsilon >= epsilon) {
                converged = false;
            }
        }
        for (int i = 0; i < quantity; i++) {
            X[i] = tempX[i];
        }

        cout << "Iteracja " << iteration << ": Epsilon " << temp_epsilon << endl;
        iteration++;
    }

    cout << "Rozwiazanie po " << iteration << " iteracjach: " << endl;
    for (int i = 0; i < quantity; i++) {
        cout << "x[" << i << "] = " << X[i] << endl;
    }

    delete[] tempX;
}
```

```cpp
else {
    ////zadanie 1
    //int iteracje = 0;
    //cout << "Wybierz ilosc iteracji: ";
    //cin >> iteracje;
    //cout << endl;
    //policz_ciag_przyblizen (quantity, iteracje, matrix_LU, matrix_D, matrix_X, matrix_B);

    //zadanie 2
    //double epsilon = 0.001;
    double epsilon = 0.000001;
    cout << "Obliczenia dla epsilon = " << epsilon << endl;
    policz_ciag_przyblizen_epsilon (quantity, epsilon, matrix_LU, matrix_D, matrix_X, matrix_B);
}
```

➔ Obliczenia dla różnych wartości ε:

ε = 0.001	ε = 0.000001
Macierz JEST diagonalnie slabo dominujaca Obliczenia dla epsilon = 0.001 Iteracja 0: Epsilon 1.4 Iteracja 1: Epsilon 0.295 Iteracja 2: Epsilon 0.427 Iteracja 3: Epsilon 0.04285 Iteracja 4: Epsilon 0.0284025 Iteracja 5: Epsilon 0.0082305 Iteracja 6: Epsilon 0.0111984 Iteracja 7: Epsilon 0.000627704 Iteracja 8: Epsilon 0.00205589 Iteracja 9: Epsilon 0.000435907 Rozwiazanie po 10 iteracjach: x[0] = 0.288821 x[1] = -1.31275 x[2] = 1.31623 x[3] = 0.669899	Macierz JEST diagonalnie slabo dominujaca Obliczenia dla epsilon = 1e-06 Iteracja 0: Epsilon 1.4 Iteracja 1: Epsilon 0.295 Iteracja 2: Epsilon 0.427 Iteracja 3: Epsilon 0.04285 Iteracja 4: Epsilon 0.0284025 Iteracja 5: Epsilon 0.0082305 Iteracja 6: Epsilon 0.0111984 Iteracja 7: Epsilon 0.000627704 Iteracja 8: Epsilon 0.00205589 Iteracja 9: Epsilon 0.000435907 Iteracja 10: Epsilon 0.000481661 Iteracja 11: Epsilon 0.000100841 Iteracja 12: Epsilon 0.000106546 Iteracja 13: Epsilon 2.92481e-05 Iteracja 14: Epsilon 2.41834e-05 Iteracja 15: Epsilon 7.39114e-06 Iteracja 16: Epsilon 5.54476e-06 Iteracja 17: Epsilon 1.89511e-06 Iteracja 18: Epsilon 1.27838e-06 Iteracja 19: Epsilon 4.72269e-07 Rozwiazanie po 20 iteracjach: x[0] = 0.289317 x[1] = -1.31306 x[2] = 1.31751 x[3] = 0.66914

Wnioski

Metoda Jacobiego to skuteczna iteracyjna metoda numeryczna służąca do rozwiązywania układów równań liniowych. Szczególnie przydatna jest w przypadku dużych i rzadkich układów równań, które często spotykane są w praktycznych problemach inżynierskich i naukowych.

Istotą metody Jacobiego jest iteracyjne przybliżanie rozwiązania poprzez wielokrotne podstawianie przybliżonych wartości niewiadomych do równań. Proces ten jest powtarzany aż do osiągnięcia zadowalającej zbieżności, co pozwala na uzyskanie przybliżonego rozwiązania układu równań.

W przeciwieństwie do dokładnych metod, takich jak metoda eliminacji Gaussa, metoda Jacobiego dostarcza przybliżone rozwiązanie. Dokładność uzyskanego wyniku zależy od liczby wykonanych iteracji, co daje możliwość dostosowania precyzji do potrzeb danego problemu.

Podsumowując, metoda Jacobiego stanowi użyteczne narzędzie do rozwiązywania układów równań liniowych, zwłaszcza w przypadku dużych i rzadkich układów. Należy jednak mieć na uwadze jej ograniczenia, takie jak wrażliwość na źle uwarunkowane układy oraz fakt, że dostarcza ona jedynie przybliżonych rozwiązań. Mimo to, dzięki możliwości kontroli dokładności i efektywności w przypadku specyficznych układów równań, metoda Jacobiego pozostaje cennym narzędziem w arsenale metod numerycznych.