| Wydział    | lmię i nazwisko:                                       | Rok: | Grupa: | Data:      |  |
|------------|--|------|--------|------------|--|
| WIMiIP     | Zuzanna Będkowska                                      | 2    | 1      | 10.04.2022 |  |
| Metody     | Temat:   |      |        |            |  |
| Numeryczne | 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:

```
for (int i = 0; i < n; ++i)
      cout << setw(5) << right << wolne[i] << "\n";
fool test1 = true; //jesli chociaz 1 test sie mywali to zmiana na false
bool test2 = false; //jesli chociaz 1 test jest ok to zmiana na true
for (int i = 0; i < n; ++i)</pre>
      if (test2 = true)
      double suma = 0.0;
for (int j = 0; j < n; ++i)
                suma += wspolczynniki[i][j];
      if (suma > wspolczynniki[i][i])
            test1 = false;
cout << "Warunek nie jest spelniony, przerwanie programu!\n";</pre>
            return 0;
      if (suma < wspolczynniki[i][i])
if (test1 == true && test2 == true)
else
     cout << "Warunek nie jest spelniony, przerwanie programu!\n"; return \theta;
                  D[i][i] = wspolczynniki[i][i];
D0[i][i] = wspolczynniki[i][i];
continue;
                  L[i][j] = wspolczynniki[i][j];
L_U[i][j] = wspolczynniki[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";</pre>
wypisz_macierz(U);
cout << "Macierz dolna + gorna:\n";
wypisz_macierz(L_U);
cout << "Macierz diagonalna:\n";
wypisz_macierz(D);</pre>
//wyznaczanie odwrotnej dopelnieniami algebraicznymi for (int i = 0; i < n; ++i)
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;
```

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:  $\frac{wyznacznik \, macierzy}{a_{ii}} \times \left(-1\right)^{i^2}$ . Z własności macierzy dopełnień algebraicznych wiadomo, że otrzymana w ten sposób macierz również jest macierza diagonalna.
- 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):

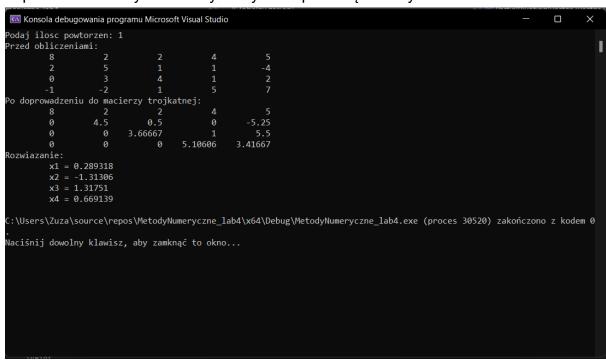
$$wyniki[i + 1] = DO \times (wolne - L_U \times wyniki[i])$$

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:
   0
Macierz diagonalnie slabo dominujaca! Program kontunuuje dzialanie
Macierz dolna:
                    0
                                           0
Macierz gorna:
                    0
Macierz dolna + gorna:
        0
                    0
        0
Macierz diagonalna:
Macierz diagonalna odwrotna:
    0.125
        0
                  0.2
        0
                    0
                            0.25
                                           0
        0
Podaj ilosc iteracji: 5
Rozwiazanie po 5iteracjach:
x0 = 0.28535
x1 = -1.2878
x2 = 1.28868
x3 = 0.692447
 :\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:



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

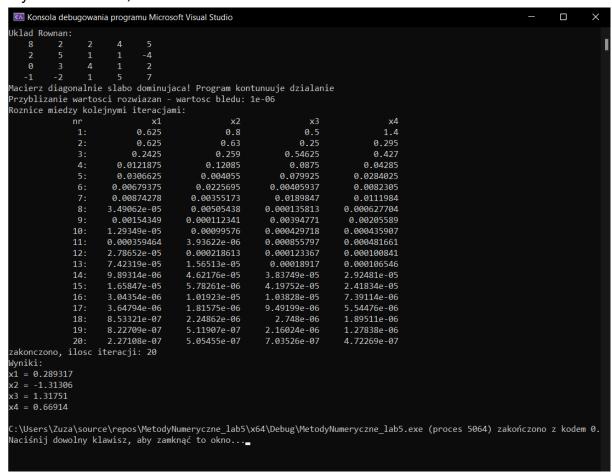
| Lp | $x_{Gauss}$ | x<br>Jacobi | $\Delta x = x_{Gauss} - x_{Jacobi}$ |
|----|-------------|-------------|-------------------------------------|
| 1  | 0.289318    | 0.28535     | 0,003968                            |
| 2  | -1.31306    | -1.2878     | 0,02526                             |
| 3  | 1.31751     | 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:

Za maksymalna ilość iteracji podano 1000.

#### Wynik dla $\varepsilon = 0,000001$ :



### Wynik dla $\varepsilon = 0,001$ :

