

Wydział WIMiP	Imię i nazwisko: Zuzanna Będkowska	Rok: 2	Grupa: 1	Data: 04.04.2022
Metody Numeryczne	Temat: Usprawnienie rozwiązywania układu równań liniowych metodą Gaussa			

Zadanie 1: wyznaczenie rozwiązania układu równań liniowych za pomocą metody Gaussa usprawnionej o wybór częściowy

Zgodnie z instrukcją, kod z poprzednich zajęć rozszerzono o następującą funkcję:

```

11 int PartialPivoting(vector<vector<double>> &macierz, vector<double> &x, int wiersz)
12 {
13     double maks = -1.0; //tu będzie element o największym module
14     int p = 0; //indeksy wiersza zamienianego z rozpatrywanym
15     for (int j = wiersz; j < macierz[wiersz].size(); ++j)
16     {
17         if (abs(macierz[j][wiersz]) > maks)
18         {
19             maks = abs(macierz[j][wiersz]);
20             p = j; //zapisujemy nr wiersza
21         }
22     }
23     if (maks == 0)
24     {
25         return -1; //układ osobliwy, ABORT THE MISSION
26     }
27     else if (wiersz == p)
28     {
29         return 1; //układ rozwiązywalny, ale zamiana nie jest potrzebna
30     }
31     else if (maks > 0) //zamiana jest potrzebna
32     {
33         double pomoc = 0.0;
34         for (int i = wiersz; i < macierz.size(); ++i) //nie trzeba zamieniac przed elementem o nr wiersz, bo te elementy sa = 0
35         {
36             pomoc = macierz[wiersz][i];
37             macierz[wiersz][i] = macierz[p][i];
38             macierz[p][i] = pomoc;
39         }
40         pomoc = x[wiersz];
41         x[wiersz] = x[p];
42         x[p] = pomoc;
43         return 1;
44     }
45 }

```

Funkcję tę wywoływano przy każdym rozpatrywanym wierszu przy postępowaniu prostym:

```

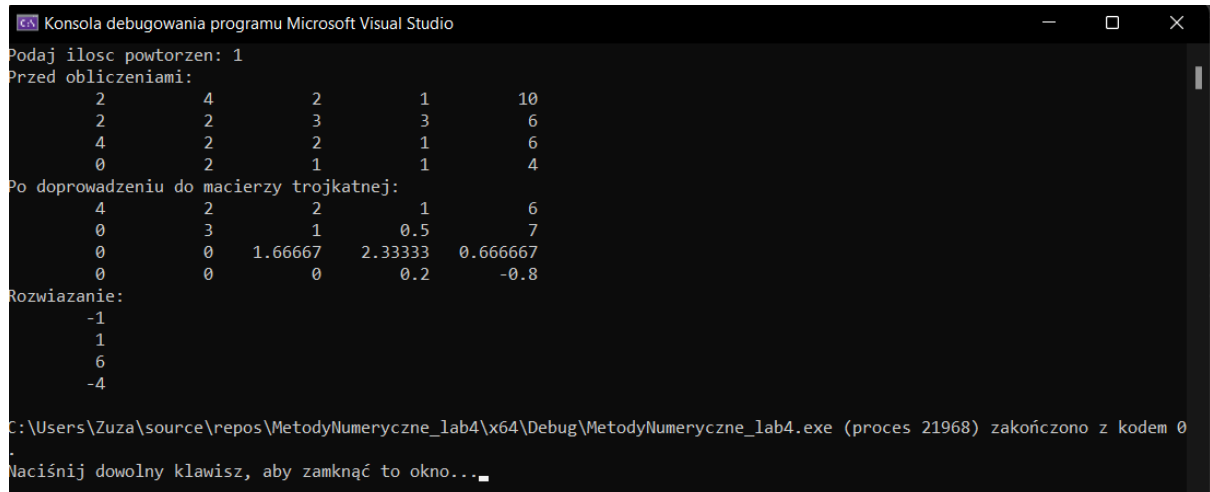
87 //część 1: postępowanie proste
88 double mnoznik = 0.0;
89 for (int i = 0; i < n - 1; ++i) //petla po wierszach
90 {
91     int kontrol = PartialPivoting(wspolczynniki, wolne, i);
92     for (int j = i + 1; j < n; ++j) //petla po elementach wiersza != 0
93     {
94         if (kontrol == -1)
95         {
96             cout << "UKŁAD OSOBLIWY, PRZERWANIE DZIAŁANIA PROGRAMU!\n";
97             return 0;
98         }
99         mnoznik = wspolczynniki[j][i] / wspolczynniki[i][i];
100         for (int k = i; k < n; ++k)
101         {
102             wspolczynniki[j][k] -= mnoznik * wspolczynniki[i][k];
103         }
104         wolne[j] -= wolne[i] * mnoznik;
105     }
106 }

```

Zastosowanie tej funkcji pozwala na eliminację zer występujących na przekątnej - korzystając z własności macierzy, wiersze zamieniano tak, aby na przekątnej znalazły się elementy co do modułu największe spośród elementów znajdujących się w tych samych kolumnach i jednocześnie będących poniżej przekątnej. Gdy

znalezionym maksimum jest zero, zamiana wierszy jest niemożliwa - wtedy zwracane jest -1.

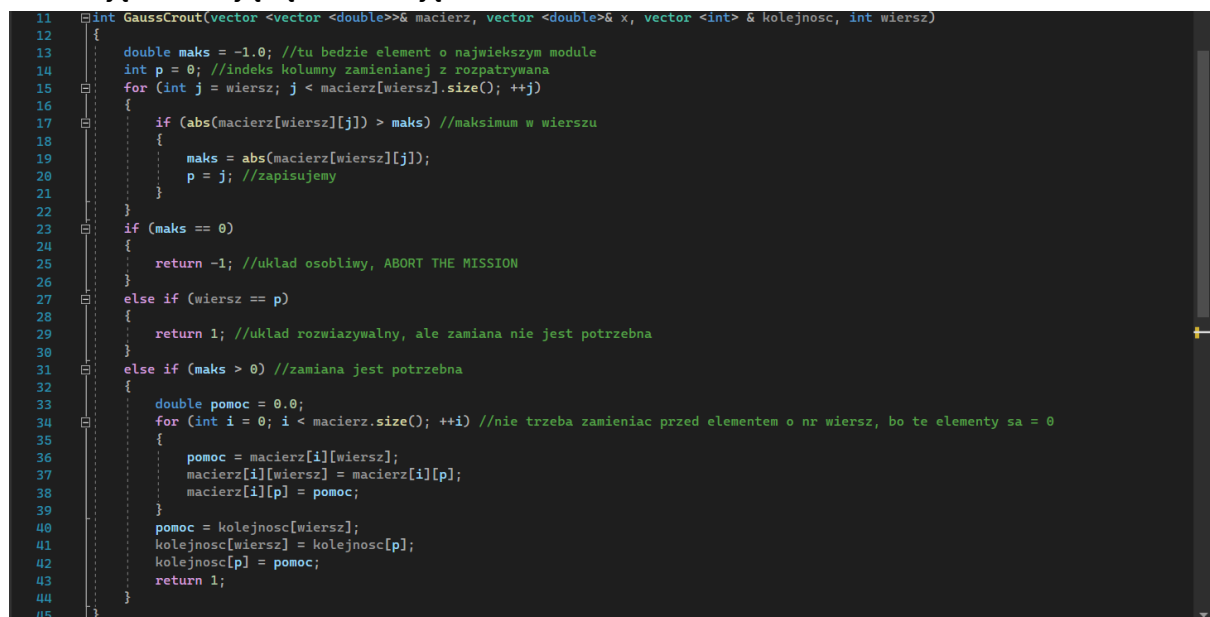
Efekt jest następujący:



```
Konsola debugowania programu Microsoft Visual Studio
Podaj ilość powtórzeń: 1
Przed obliczeniami:
    2      4      2      1      10
    2      2      3      3      6
    4      2      2      1      6
    0      2      1      1      4
Po doprowadzeniu do macierzy trojkatnej:
    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
Rozwiązanie:
    -1
     1
     6
    -4
C:\Users\Zuza\source\repos\MetodyNumeryczne_lab4\x64\Debug\MetodyNumeryczne_lab4.exe (proces 21968) zakończono z kodem 0
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Zadanie 2: wyznaczenie rozwiązania układu równań liniowych za pomocą metody Gaussa usprawnionej o eliminację Crouta:

Podobnie jak w zadaniu 1, program napisany na poprzednich zajęciach usprawniono o funkcję realizującą eliminację Crouta:



```
11 int GaussCrout(vector<vector<double>>& macierz, vector<double>& x, vector<int> & kolejnosc, int wiersz)
12 {
13     double maks = -1.0; //tu będzie element o największym module
14     int p = 0; //indeks kolumny zamienianej z rozpatrywaną
15     for (int j = wiersz; j < macierz[wiersz].size(); ++j)
16     {
17         if (abs(macierz[wiersz][j]) > maks) //maksimum w wierszu
18         {
19             maks = abs(macierz[wiersz][j]);
20             p = j; //zapisujemy
21         }
22     }
23     if (maks == 0)
24     {
25         return -1; //układ osobliwy, ABORT THE MISSION
26     }
27     else if (wiersz == p)
28     {
29         return 1; //układ rozwiązywalny, ale zamiana nie jest potrzebna
30     }
31     else if (maks > 0) //zamiana jest potrzebna
32     {
33         double pomoc = 0.0;
34         for (int i = 0; i < macierz.size(); ++i) //nie trzeba zamieniac przed elementem o nr wiersz, bo te elementy sa = 0
35         {
36             pomoc = macierz[i][wiersz];
37             macierz[i][wiersz] = macierz[i][p];
38             macierz[i][p] = pomoc;
39         }
40         pomoc = kolejnosc[wiersz];
41         kolejnosc[wiersz] = kolejnosc[p];
42         kolejnosc[p] = pomoc;
43         return 1;
44     }
45 }
```

Funkcja ta wywoływana jest w tym samym miejscu co w zadaniu 1. Analogicznie do zadania 1, funkcja zamienia ze sobą kolumny, w celu uzyskania na przekątnej maksimumów co do modułu dla wszystkich elementów danego wiersza będących poniżej przekątnej. Jeżeli maksimum jest równe zero - zamiana kolumn jest niemożliwa i funkcja zwraca -1. Dodatkowo w celu zapewnienia czytelności rozwiązania, dodano vector przechowujący indeksy zamienianych wierszy - dzięki temu można odtworzyć kolejność elementów w rozwiązaniu.

Efekt jest następujący:

```
Konsola debugowania programu Microsoft Visual Studio
Podaj ilosc powtorzen: 1
Przed obliczeniami:
  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 doprowadzeniu do macierzy trojkatnej:
 -5      1     -2      1      1      8
  0    -3.4    -2.2      1.8      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
Rozwiazanie:
  x6 = -1.68935
  x2 = -1.49549
  x3 = 17.918
  x5 = -7.96649
  x1 = -9.43387
  x4 = 30.3857
C:\Users\Zuza\source\repos\MetodyNumeryczne_lab4\x64\Debug\MetodyNumeryczne_lab4.exe (proces 12560) zakończono z kodem 0
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Wnioski:

Dzięki rozszerzeniu metody Gaussa możliwe jest wyznaczenie istniejących jednoznacznych rozwiązań równań liniowych.