

<p>Wydział WIMiP</p>	<p>Imię i nazwisko: Zuzanna Będkowska</p>	<p>Rok: 2</p>	<p>Grupa: 1</p>	<p>Data: 01.05.2022</p>
<p>Metody Numeryczne</p>	<p>Temat: Aproksymacja</p>			

Zadanie: wyznaczenie wielomianu aproksymacyjnego dowolnego stopnia

Zgodnie z instrukcją, w trakcie zajęć stworzono następujący kod:

```

1  #include <iostream>
2  #include <fstream>
3  #include <cmath>
4  #include <iomanip>
5  #include <vector>
6
7  using namespace std;
8
9  double fi(double& x, double j) // obliczanie wartosci funkcji fi_j i fi_k
10 {
11     return pow(x, j);
12 }
13
14 double wielomian(double x, vector<double>& a)
15 {
16     double wynik = 0;
17     for (int i = 0; i < a.size(); ++i)
18     {
19         wynik += fi(x, i) * a[i];
20     }
21     return wynik;
22 }
23
24 int PartialPivoting(vector<vector<double>>& macierz, vector<double>& x, int wiersz)
25 {
26     double maks = -1.0; //tu bedzie element o najwiekszym module
27     int p = 0; //indeksy wiersza zamienianego z rozpatrywanym
28     for (int j = wiersz; j < macierz[wiersz].size(); ++j)
29     {
30         if (abs(macierz[j][wiersz]) > maks)
31         {
32             maks = abs(macierz[j][wiersz]);
33             p = j; //zapisujemy nr wiersza
34         }
35     }
36     if (maks == 0)
37     {
38         return -1; //uklad osobliwy, ABORT THE MISSION
39     }
40     else if (wiersz == p)
41     {
42         return 1; //uklad rozlaczalny, ale zamiana nie jest potrzebna
43     }
44     else if (maks > 0) //zamiana jest potrzebna
45     {
46         double pomoc = 0.0;
47         for (int i = wiersz; i < macierz.size(); ++i) //nie trzeba zamieniac przed elementem o nr wiersz, bo te elementy sa = 0
48         {
49             pomoc = macierz[wiersz][i];
50             macierz[wiersz][i] = macierz[p][i];
51             macierz[p][i] = pomoc;
52         }
53         pomoc = x[wiersz];
54         x[wiersz] = x[p];
55         x[p] = pomoc;
56         return 1;
57     }
58 }
59
60 vector<double> Gauss(int n, vector<vector<double>>& wspolczynniki, vector<double>& wolne)
61 {
62     vector<double> wyniki(n, 0);
63     double mnoznik = 0.0;
64     for (int i = 0; i < n - 1; ++i) //petla po wierszach
65     {
66         for (int j = i + 1; j < n; ++j) //petla po elementach wiersza != 0
67         {
68             int kontrol = PartialPivoting(wspolczynniki, wolne, i);
69             if (kontrol == -1)
70             {
71                 cout << "UKLAD OSOBLIWE, PRZERWANIE DZIALANIA PROGRAMU!\n";
72             }
73             mnoznik = wspolczynniki[j][i] / wspolczynniki[i][i];
74             for (int k = i; k < n; ++k)

```

```

75         {
76             wspolczynniki[j][k] -= mnoznik * wspolczynniki[i][k];
77         }
78         wolne[j] -= wolne[i] * mnoznik;
79     }
80 }
81 //czesc 2: postepowanie odwrotne
82 for (int i = n - 1; i >= 0; --i)
83 {
84     double skladnik = 0;
85     for (int k = i; k < n; k++)
86     {
87         skladnik += wyniki[k] * wspolczynniki[i][k];
88     }
89     wyniki[i] = (wolne[i] - skladnik) / wspolczynniki[i][i];
90 }
91 for (int i = 0; i < n; ++i)
92 {
93     cout << setw(10) << right << "a" << i << " = " << wyniki[i] << "\n";
94 }
95 return wyniki;
96 }
97
98 int main()
99 {
100     int n = 1; //stopien wielomianu
101     int wezly = 8;
102     vector<double> wagi(wezly, 1.0); //wagi
103     vector<double> F(n + 1, 0);
104     vector<double> a(n + 1, 0);
105     vector<vector<double>> macierz_g(n + 1, F); //macierz
106     vector<pair<double, double>> punkty;
107     punkty.push_back(make_pair(1.0, 2.0));
108     punkty.push_back(make_pair(2.0, 4.0));
109     punkty.push_back(make_pair(3.0, 3.0));
110     punkty.push_back(make_pair(4.0, 5.0));
111     punkty.push_back(make_pair(5.0, 6.0));
112     punkty.push_back(make_pair(6.0, 9.0));
113     punkty.push_back(make_pair(7.0, 11.0));
114     punkty.push_back(make_pair(8.0, 11.0));
115     //uzupelnianie macierzy
116
117     for (int k = 0; k < n + 1; ++k)
118     {
119         for (int j = 0; j < n + 1; ++j)
120         {
121             for (int i = 0; i < wezly; ++i)
122             {
123                 macierz_g[k][j] += fi(punkty[i].first, k) * fi(punkty[i].first, j) * wagi[i];
124             }
125         }
126     }
127
128     for (int k = 0; k < n + 1; ++k)
129     {
130         for (int i = 0; i < wezly; ++i)
131         {
132             F[k] += fi(punkty[i].first, k) * (punkty[i].second) * wagi[i];
133         }
134     }
135     cout << "Rozwiazanie:\n\t- Ilosc wezlow: " << wezly << "\n\t- Wspolczynniki wielomianu aproksymujacego:\n";
136     a = Gauss(n + 1, macierz_g, F);
137     cout << "\n\t- Wezly aproksymacji i ich wartosci: \n";
138     for (int i = 0; i < wezly; ++i)
139     {
140         cout << "\t\t" << i + 1 << ": " << punkty[i].first << ", " << punkty[i].second << "\n";
141     }
142     cout << "obliczone wartosci funkcji aproksymujacej w wezlach aproksymacji:\n";
143     for (int i = 0; i < wezly; ++i)
144     {
145         cout << "\t\t" << i + 1 << ": " << punkty[i].first << ", " << wielomian(punkty[i].first, a) << "\n";
146     }
147 }

```

Za pomocą wzorów umieszczonych w instrukcji wyznaczono *macierz_g* i wektor *F* będące częściami układu równań liniowych, którego niewiadomymi są współczynniki wielomianu aproksymującego. Układ ten rozwiązano za pomocą metody Gaussa rozszerzonej o Partial Pivoting zaimplementowanej w trakcie laboratorium 4.

Efekt działania programu jest następujący:

```
Konsola debugowania programu Microsoft Visual Studio
Rozwiązanie:
- Ilość węzłów: 8
- Współczynniki wielomianu aproksymującego:
  a0 = 0.107143
  a1 = 1.39286
- Wzły aproksymacji i ich wartości:
  1:1; 2
  2:2; 4
  3:3; 3
  4:4; 5
  5:5; 6
  6:6; 9
  7:7; 11
  8:8; 11
obliczone wartości funkcji aproksymującej w węzłach aproksymacji:
  1:1; 1.5
  2:2; 2.89286
  3:3; 4.28571
  4:4; 5.67857
  5:5; 7.07143
  6:6; 8.46429
  7:7; 9.85714
  8:8; 11.25
C:\Users\Zuza\source\repos\MetodyNumeryczne_lab8\x64\Debug\MetodyNumeryczne_lab8.exe (proces 28268) zakończono z kodem 0
Naciśnij dowolny klawisz, aby zamknąć to okno...
```