Laboratorium Metod Numerycznych						
Rok akademicki	Termin	Rodzaj studiów	Kierunek	Prowadzący	Grupa	Sekcja
2017/2018	Czwartek 16:45 - 19:45	Dzienne	INF	Dr inż. Ewa Starzewska-Karwan	6	11

Sprawozdanie z ćwiczenia numer 1

Data wykonania ćwiczenia: 2018-03-15

Temat ćwiczenia:
Układy równań liniowych
Metoda Seidela

Skład podsekcji:
Michał Miciak
Bartłomiej Krasoń

1.Cel ćwiczenia

Celem naszego ćwiczenia było utworzenie programu w języku C++, który rozwiązuje metodą Seidela podane układy równań liniowych w postaci macierzy. Dane wejściowe przekazywane są do programu przez pliki tekstowe. Program generuje raport z wykonanych obliczeń. W ramach analizy zadania należało wyznaczyć błąd bezwzględny rozwiązania oraz sprawdzić warunki zbieżności dla wszystkich zadanych zestawów danych.

2.Kod programu

```
#include <iostream>
#include <math.h>
#include <fstream>
#include <string>
using namespace std;

void funA(double **&A, double *&B, double **&Alpha, double *&Beta, double *&New,
double *&Old, int &n, double &epsilon, int &MLI)
{
	//Przygotowanie pliku
	fstream input;
	input.open("plik.txt", ios::in);
	if (!input.good())
	{
		throw exception();
	}

	//Pobranie danych od użytkownika
	cout << "Podaj rozmiar n macierzy A i  wektorów B i X: ";
	int _n;
	cin >> _n;
	n = _n;
	cout << endl;

	cout << "Podaj dokładnosc epsilon: ";
	double _epsilon;
	cin >> _epsilon;
	epsilon = _epsilon;
	cout << endl;

	cout << "Podaj maksymalna liczbe iteracji MLI: ";
	int _MLI;
	cin >> _MLI;
	MLI = _MLI;
	cout << endl;

	//Zaalokowanie miejsca dla macierzy i wektorów
	A = new double*[n];
	for (int i = 0; i < n; i++)
	{
		A[i] = new double[n];
	}

	B = new double[n];

	Alpha = new double*[n];
	for (int i = 0; i < n; i++)
	{
```

```
			Alpha[i] = new double[n];
		}

		Beta = new double[n];

		New = new double[n];

		Old = new double[n];

		//Wypełnienie macierzy A i wektora B danymi z pliku
		for (int i = 0; i < n; i++)
		{
			for (int j = 0; j < n; j++)
			{
				input >> A[i][j];
			}
		}

		for (int i = 0; i < n; i++)
		{
			input >> B[i];
		}
}

bool funB(double **&A, double *&B, double **&Alpha, double *&Beta, double *&New,
double *&Old, int &n, double &epsilon, int &MLI)
{
		//Wypełnienie macierzy alpha i wektora beta
		for (int i = 0; i < n; i++)
		{
			//Wyjście z funkcji i zwrócenie false gdy na przekątnej napotkamy zero
			if (A[i][i] == 0)
			{
				return false;
			}
			//Obliczanie wartości zgodnie z wzorami z instrukcji
			Beta[i] = B[i] / A[i][i];
			for (int j = 0; j < n; j++)
			{
				if (i == j)
				{
					Alpha[i][j] = 0;
				}
				else
				{
					Alpha[i][j] = -(A[i][j] / A[i][i]);
				}
			}
		}
		return true;

}
void funC(double **&A, double *&B, double **&Alpha, double *&Beta, double *&New,
double *&Old, int &n, double &epsilon, int &MLI, int &LI)
{
		int it = 0; //liczba iteracji
		double norm; //norma stanowiąca warunek zakończenia algorytmu

		//W pierwszej iteracji wektor rozwiązań to wektor beta
		for (int i = 0; i < n; i++)
		{
			New[i] = B[i];
```

```
	}

	//Algorytm wyznaczania kolejnych wektorów rozwiązań
	do
	{
		//Zaktualizowanie wektora rozwiązań przedostatniej iteracji
		for (int i = 0; i < n; i++)
		{
			Old[i] = New[i];
		}
		//Zwiększenie licznika iteracji
		it++;
		//Zerowanie wektora ostatniej iteracji (aktualnej)
		for (int i = 0; i < n; i++)
		{
			New[i] = 0;
		}
		//Aktualizacja pierwszego elementu wektora rozwiązań
		for (int j = 1; j < n; j++)
		{
			New[0] += Alpha[0][j] * Old[j];
		}
		New[0] += Beta[0];
		//Aktualizacja pozostałych elementów wektora rozwiązań
		for (int i = 1; i < n; i++)
		{
			for (int j = 0; j <= i - 1; j++)
			{
				New[i] += Alpha[i][j] * New[j];
			}
			for (int j = i + 1; j < n; j++)
			{
				New[i] += Alpha[i][j] * Old[j];
			}
			New[i] += Beta[i];
		}
		//Obliczenie normy dla aktualnej iteracji
		norm = 0;
		for (int i = 0; i < n; i++)
		{
			norm += pow((New[i] - Old[i]), 2);
		}
		norm = sqrt((norm / n));
	//Wykonanie algorytmu kończy się gdy norma będzie mniejsza od epsilonu lub
	liczba iteracji wyniesie MLI
	} while (it<MLI&&norm>epsilon);
	//Zapisanie liczby iteracji
	LI = it;
}

void funD(double n, double **A, double *B, double epsilon, int MLI, double **alfa,
double *beta, double *Old, double *New, double LI, int nr_zestawu, bool goodCondition)
{
	//Utworzenie pliku raportu
	std::fstream file;
	std::string filename = "Raport_Zestaw";
	filename += to_string(nr_zestawu);
	filename += ".txt";
	file.open(filename, std::ios_base::out);
	if (!file.is_open())
	{
		std::cout << "Blad pliku";
```

```
			return;
		}
		//Wypisanie danych wejściowych
		file << "-----DANE WEJSCIOWE-----\n";
		file << "Epsilon: " << epsilon;
		file << ", Maks. liczba iteracji: " << MLI << std::endl;
		file << "Macierz A:";
		for (int i = 0; i < (n + 8); i++)
			file << " ";
		file << "Wektor B:\n";
		for (int i = 0; i < n; i++)
		{
			for (int j = 0; j < n; j++)
			{
				if (j == 0)   file << "|";
				file.width(3);
				file << std::internal << A[i][j];

				if (j == (n - 1)) file << "|    |" << B[i] << "|";
				else file << " ";
			}
			file << std::endl;
		}
		//Sprawdzenie czy dane wejściowe spełniły warunki metody
		if (!goodCondition)
		{
			file << "Dana macierz nie spełnia założeń metody Seidela\n";
			file.close();
			return;
		}
		//Wypisanie danych pośrednich
		file << "-----DANE POSREDNIE-----\n";
		file << "Macierz alfa:";
		for (int i = 0; i < (n + 23); i++)
			file << " ";
		file << "Wektor beta:\n";
		for (int i = 0; i < n; i++)
		{
			for (int j = 0; j < n; j++)
			{
				if (j == 0)   file << "|";
				file.precision(3);
				file.width(6);
				file << std::right << alfa[i][j];

				if (j == (n - 1)) file << "|    |" << beta[i] << "|";
				else file << "  ";
			}
			file << std::endl;
		}
		//Wypisanie wyników obliczeń
		file << "-----WYNIKI OBLICZEN-----\n";
		file << "Ilosc wykonanych iteracji: " << LI << std::endl;
		file << "Wektor ostatniej iteracji:         Wektor przedostatniej iteracji:\n";
		for (int i = 0; i < n; i++)
		{
			file.precision(10);
			file.setf(std::ios::scientific);
			file << "    |" << New[i] << "|                    |" << Old[i] <<
"|\n";
		}
		file.close();
```

```
}

int main()
{
	double **A = nullptr; //macierz A
	double *B = nullptr; //wektor B
	double **Alpha = nullptr; //macierz alfa
	double *Beta = nullptr; //macierz beta
	double *New = nullptr; //wektor rozwiązań ostatniej iteracji
	double *Old = nullptr; //wektor rozwiązań przedostatniej iteracji
	int n; //rozmiar macierzy i wektorów
	double epsilon; //dokładność epsilon
	int MLI; //maksymalna liczba iteracji
	int LI; //liczba iteracji

	funA(A, B, Alpha, Beta, New, Old, n, epsilon, MLI);
	//funkcja funB() zwraca 'true', gdy warunki metody zostały spełnione
	bool notZeroOnDiagonal = funB(A, B, Alpha, Beta, New, Old, n, epsilon, MLI);
	funC(A, B, Alpha, Beta, New, Old, n, epsilon, MLI, LI);
	funD(n, A, B, epsilon, MLI, Alpha, Beta, Old, New, LI, nr, notZeroOnDiagonal);

	getchar();
}
```

3.Format pliku wejściowego

Dane wejściowe znajdujące się w pliku tekstowym powinny być ułożone w następujący sposób (Dany przykład tyczy się macierzy o rozmiarze n=5, w przypadku innego rozmiaru należy postępować analogicznie):

$$
\begin{matrix}
a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\
a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\
a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\
a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\
a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \\
b_1 \\
b_2 \\
b_3 \\
b_4 \\
b_5
\end{matrix}
\qquad \text{lub} \qquad
\begin{matrix}
a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\
a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\
a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\
a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\
a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \\
b_1 & b_2 & b_3 & b_4 & b_5
\end{matrix}
$$

Plik z danymi należy nazwać „plik.txt”.
Po uruchomieniu programu należy podać:

a) rozmiar macierzy A oraz wektora B
b) epsilon
c) maksymalną liczbę iteracji

4. Wyniki działania programu

a) Zestaw 1

```
-----DANE WEJSCIOWE-----
Epsilon: 0.0001, Maks. liczba iteracji: 30
Macierz A:              Wektor B:
| 10   1   1   1   2|    |15|
|  3  20   4   2   1|    |30|
|  5   1  40   9   5|    |60|
|  3   0   1  10   1|    |15|
|  6   2   1   1  20|    |30|
-----DANE POSREDNIE-----
Macierz alfa:                              Wektor beta:
|     0  -  0.1  -  0.1  -  0.1  -  0.2|    |1.5|
|- 0.15       0  -  0.2  -  0.1  - 0.05|    |1.5|
|-0.125  -0.025       0  -0.225  -0.125|    |1.5|
|-  0.3  -    0  -  0.1       0  -  0.1|    |1.5|
|-  0.3  -  0.1  - 0.05  - 0.05       0|    |1.5|
-----WYNIKI OBLICZEN-----
Ilosc wykonanych iteracji: 8
Wektor ostatniej iteracji:          Wektor przedostatniej iteracji:
    |9.9999148786e-01|                  |1.0000670814e+00|
    |9.9999376210e-01|                  |1.0001139984e+00|
    |1.0000089802e+00|                  |1.0000538902e+00|
    |1.0000049979e+00|                  |9.9998407823e-01|
    |1.0000024785e+00|                  |9.9996657732e-01|
```

b) Zestaw 2

```
-----DANE WEJSCIOWE-----
Epsilon: 0.0001, Maks. liczba iteracji: 30
Macierz A:              Wektor B:
| 10 - 1   1   1   2|    |13|
|- 3  20   4   2   1|    |24|
|  5 - 1  40   9   5|    |58|
|  3   0 - 1  10   1|    |13|
|  6 - 2   1   1  20|    |26|
-----DANE POSREDNIE-----
Macierz alfa:                              Wektor beta:
|     0     0.1  -  0.1  -  0.1  -  0.2|    |1.3|
|  0.15       0  -  0.2  -  0.1  - 0.05|    |1.2|
|-0.125   0.025       0  -0.225  -0.125|    |1.45|
|-  0.3  -    0     0.1       0  -  0.1|    |1.3|
|-  0.3     0.1  - 0.05  - 0.05       0|    |1.3|
-----WYNIKI OBLICZEN-----
Ilosc wykonanych iteracji: 8
Wektor ostatniej iteracji:          Wektor przedostatniej iteracji:
    |9.9999515046e-01|                  |9.9998270295e-01|
    |1.0000006972e+00|                  |9.9994834033e-01|
    |9.9999897565e-01|                  |9.9998922032e-01|
    |1.0000013323e+00|                  |1.0000072124e+00|
    |1.0000015092e+00|                  |1.0000002015e+00|
```

c) Zestaw 3

```
-----DANE WEJSCIOWE-----
Epsilon: 0.0001, Maks. liczba iteracji: 30
Macierz A:                Wektor B:
| 10   2   3   1   4|     |20|
|  3  20   7   2   8|     |40|
| 15   5  40   9  11|     |80|
|  3   0   6  10   1|     |20|
|  6   4   5   5  20|     |40|
-----DANE POSREDNIE-----
Macierz alfa:                                Wektor beta:
|     0  -  0.2  -  0.3  -  0.1  -  0.4|     |2|
|- 0.15       0  - 0.35  -  0.1  -  0.4|     |2|
|-0.375  -0.125       0  -0.225  -0.275|     |2|
|-  0.3  -    0  -  0.6       0  -  0.1|     |2|
|-  0.3  -  0.2  - 0.25  - 0.25       0|     |2|
-----WYNIKI OBLICZEN-----
Ilosc wykonanych iteracji: 11
Wektor ostatniej iteracji:          Wektor przedostatniej iteracji:
    |1.0000306843e+00|                    |1.0000973798e+00|
    |1.0000267831e+00|                    |9.9998239921e-01|
    |9.9998553165e-01|                    |9.9991556702e-01|
    |1.0000000743e+00|                    |1.0000055992e+00|
    |9.9998903661e-01|                    |9.9999401464e-01|
```

d) Zestaw 4

```
-----DANE WEJSCIOWE-----
Epsilon: 0.0001, Maks. liczba iteracji: 30
Macierz A:                Wektor B:
| 10   3   3   1   4|     |21|
|  4  20   7   2   8|     |41|
| 15   5  40  10  11|     |81|
|  4   0   6  10   1|     |21|
|  6   4   5   7  20|     |42|
-----DANE POSREDNIE-----
Macierz alfa:                                Wektor beta:
|     0  -  0.3  -  0.3  -  0.1  -  0.4|     |2.1|
|-  0.2       0  - 0.35  -  0.1  -  0.4|     |2.05|
|-0.375  -0.125       0  - 0.25  -0.275|     |2.02|
|-  0.4  -    0  -  0.6       0  -  0.1|     |2.1|
|-  0.3  -  0.2  - 0.25  - 0.35       0|     |2.1|
-----WYNIKI OBLICZEN-----
Ilosc wykonanych iteracji: 12
Wektor ostatniej iteracji:          Wektor przedostatniej iteracji:
    |9.9996026502e-01|                    |9.9999833839e-01|
    |1.0000050977e+00|                    |1.0001313678e+00|
    |1.0000268836e+00|                    |1.0000504928e+00|
    |1.0000031345e+00|                    |9.9998659704e-01|
    |1.0000030830e+00|                    |9.9996629277e-01|
```

e) Zestaw 5

```
|-----DANE WEJSCIOWE-----
Epsilon: 0.0001, Maks. liczba iteracji: 30
Macierz A:              Wektor B:
| 10   2   3   1   4|     |20|
|  3  20   7   2   8|     |40|
| 15   5  40   9  11|     |80|
|  3   0   6   0   1|     |10|
|  6   4   5   5  20|     |40|
Dana macierz nie spełnia założeń metody Seidela
```

f) Zestaw 6

```
|-----DANE WEJSCIOWE-----
Epsilon: 0.0001, Maks. liczba iteracji: 5
Macierz A:              Wektor B:
| 10   3   3 100   4|     |120|
|  4  20   7   2  80|     |113|
|150   5  40  10  11|     |216|
|  4   0  60  10   1|     |75|
|  6   4   5  70  20|     |105|
-----DANE POSREDNIE-----
Macierz alfa:                              Wektor beta:
|     0    -0.3    -0.3     -10    -0.4|     |12|
|  -0.2       0   -0.35    -0.1      -4|     |5.65|
| -3.75  -0.125       0   -0.25  -0.275|     |5.4|
|  -0.4      -0      -6       0    -0.1|     |7.5|
|  -0.3    -0.2   -0.25    -3.5       0|     |5.25|
-----WYNIKI OBLICZEN-----
Ilosc wykonanych iteracji: 5
Wektor ostatniej iteracji:         Wektor przedostatniej iteracji:
    |-9.8582526248e+11|                    |5.4463381732e+09|
    |1.7873215112e+12|                    |-9.8743234744e+09|
    |3.5542197793e+12|                    |-1.9635815378e+10|
    |-2.0891117932e+13|                    |1.1541608573e+11|
    |7.2168641095e+13|                    |-3.9870638298e+11|
```

5. Sprawdzenie warunków zbieżności

Obliczenia zostały przeprowadzone według następujących wzorów norm w arkuszu kalkulacyjnym Microsoft Excel:

$$\| \alpha \|_{I} = \max_{i} \sum_{j=1}^{n} | \alpha_{ij} | < 1$$

$$\| \alpha \|_{II} = \max_{j} \sum_{i=1}^{n} | \alpha_{ij} | < 1$$

$$\| \alpha \|_{III} = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_{ij}^{2}} < 1$$

ZESTAW 1			ZESTAW 2		
$\|\|\alpha\|\|_I$	$\|\|\alpha\|\|_{II}$	$\|\|\alpha\|\|_{III}$	$\|\|\alpha\|\|_I$	$\|\|\alpha\|\|_{II}$	$\|\|\alpha\|\|_{III}$
0,875	0,5	0,66521	0,875	0,5	0,66521
Warunek zbieżności został spełniony.			Warunek zbieżności został spełniony.		
ZESTAW 3			**ZESTAW 4**		
$\|\|\alpha\|\|_I$	$\|\|\alpha\|\|_{II}$	$\|\|\alpha\|\|_{III}$	$\|\|\alpha\|\|_I$	$\|\|\alpha\|\|_{II}$	$\|\|\alpha\|\|_{III}$
1,175	1	1,26984	1,275	1,1	1,34977
Warunek zbieżności został nie spełniony.			Warunek zbieżności nie został spełniony.		
ZESTAW 5			**ZESTAW 6**		
$\|\|\alpha\|\|_I$	$\|\|\alpha\|\|_{II}$	$\|\|\alpha\|\|_{III}$	$\|\|\alpha\|\|_I$	$\|\|\alpha\|\|_{II}$	$\|\|\alpha\|\|_{III}$
Macierz alfa nie została utworzona. Nie da się wyznaczyć warunku wyst. zbieżności.			4,775	11	13,39184
			Warunek zbieżności nie został spełniony.		

6. Wyznaczanie błędu bezwzględnego dla zestawów

Błąd bezwzględny wyznaczyliśmy za pomocą wzoru:

$$\Delta(x_i) = |X_i - x_i|, \qquad gdzie: X_i = 1 - wartość\ dokładna$$
$$x_i - wartość\ przybliżona$$

ZESTAW 1	ZESTAW 2
$\Delta(x_1)$= 8.512139e-06 $\Delta(x_2)$= 6.237901e-06 $\Delta(x_3)$= 8.980199e-06 $\Delta(x_4)$ =4.997890e-06 $\Delta(x_5)$ = 2.478527e-06	$\Delta(x_1)$ =4.849540e-06 $\Delta(x_2)$ =6.971914e-07 $\Delta(x_3)$ =1.024354e-06 $\Delta(x_4)$ =1.332275e-06 $\Delta(x_5)$ =1.509185e-06
ZESTAW 3	**ZESTAW 4**
$\Delta(x_1)$ = 3.068427e-05 $\Delta(x_2)$ = 2.678312e-05 $\Delta(x_3)$ = 1.446835e-05 $\Delta(x_4)$ = 7.426279e-08 $\Delta(x_5)$ = 1.096339e-05	$\Delta(x_1)$ = 3.973498e-05 $\Delta(x_2)$ = 5.097706e-06 $\Delta(x_3)$ = 2.688363e-05 $\Delta(x_4)$ = 3.134535e-06 $\Delta(x_5)$ = 3.082957e-06
ZESTAW 5	**ZESTAW 6**
Wektor rozwiązań nie został utworzony. Nie da się wyznaczyć błędu bezwzględnego.	$\Delta(x_1)$ =9.858253e+11 $\Delta(x_2)$ =1.787322e+12 $\Delta(x_3)$ =3.554220e+12 $\Delta(x_4)$ =2.089112e+13 $\Delta(x_5)$ =7.216864e+13

7. Wnioski

1. Analiza uzyskanych wyników potwierdza tezę, iż Metoda Seidela jest niedokładną metodą znajdowania rozwiązań układów równań liniowych. Uzyskiwane wyniki zawsze są opatrzone tak zwanym błędem metody. Wartości tych błędów wyznaczyliśmy za pomocą błędów bezwzględnych obliczonych dla rozwiązań każdego z zestawów przyjmując za dokładne rozwiązanie wektor $X = [1\ \ 1\ \ 1\ \ 1\ \ 1]^T$

2. Jako że jest to metoda iteracyjna, trzeba w niej założyć jakieś warunki stopu obliczeń. My przyjęliśmy je jako wyznaczenie normy różnicy wektorów przedostatniej i ostatniej iteracji oraz porównanie jej z przyjętą dokładnością epsilon. W naszym programie wykorzystaliśmy normę trzeciego rzędu:

$$\left\|X^{(k+1)} - X^{(k)}\right\|_{III} = \sqrt{\frac{\sum_{i=1}^{n}\left|x_i^{(k+1)} - x_i^{(k)}\right|^2}{n}} \leq \epsilon$$

   Ten warunek zakończył działanie algorytmu dla zestawów 1-5. Jedynie dla zestawu 6. działanie algorytmu zakończył dodatkowy warunek stopu jakim była maksymalna liczba iteracji (w tym przypadku równa 5).
3. Zadowalające rozwiązania uzyskaliśmy dla zestawów 1. i 2., gdyż wszystkie elementy wektora rozwiązania spełniły oczekiwaną dokładność do czterech pozycji po przecinku ($\epsilon = 0.0001$). Wynika to z tego, że dla tych zestawów warunki zbieżności zostały spełnione.
4. W przypadku zestawów 3. i 4. warunki zbieżności nie zostały spełnione, toteż nie wszystkie elementy wektorów rozwiązań uzyskały oczekiwaną dokładność. Natomiast wyniki te nie spełniają oczekiwanej dokładności zaledwie na pojedynczych pozycjach. Wynika to z tego że metoda Seidela jest ulepszeniem metody Jacobiego, przez co jest ona szybciej zbieżna. Często nawet jest zbieżna gdy proces iteracji prostej Jacobiego jest rozbieżny.
5. Metoda Seidela nie jest metodą uniwersalną, aby za jej pomocą wyznaczyć rozwiązanie układu musi zostać spełniony warunek tej metody, który jest następujący:

$$a_{ii} \neq 0 \quad dla\ i = 1,2,\dots,n$$

   Czyli na przekątnej macierzy A nie może wystąpić „0”. Dla zestawu 5. warunek ten nie został spełniony, toteż nie mogliśmy uzyskać dla niego rozwiązania.
6. Zdecydowanie nie udało nam się uzyskać oczekiwanego wyniku w przypadku zestawu 6. Wynika to z silnej rozbieżności procesu iteracyjnego dla tego przykładu, co obrazują uzyskane dla niego normy macierzy alfa. Tak wysokie wartości norm są skutkiem braku silnej dominacji elementów macierzy A leżących na głównej przekątnej nad pozostałymi elementami. Przykład ten obrazuje wadę tej metody, która w takich warunkach prowadzi do coraz większych błędów rozwiązań w kolejnych iteracjach procesu.