Міністерство освіти і науки України
Національний університет “Львівська політехніка”
Інститут комп’ютерних наук та інформаційних технологій
Кафедра програмного забезпечення

Звіт
Про виконання лабораторної роботи №5

На тему:

«Наближені методи розв’язування СЛАР»
з дисципліни «Чисельні методи»

Лектор:
доцент каф. ПЗ
Мельник Н. Б.
Виконав:
ст. гр. ПЗ-11
Морозов О. Р.
Прийняла:
доцент каф. ПЗ
Мельник Н. Б.

« __ » _____ 2022 р.

∑ = _____ _____ .

Львів – 2022

Тема: Наближені методи розв'язування СЛАР
Мета: ознайомлення на практиці з методами Якобі та Зейделя розв'язування систем лінійних алгебраїчних рівнянь.

Теоретичні відомості

6.1. Метод простої ітерації (Якобі)

Розглянемо систему лінійних алгебраїчних рівнянь. Припустивши, що коефіцієнти aii ≠ 0 (i =1,n) , розв'яжемо і -те рівняння системи відносно x . Таку систему називають зведеною. У матричному вигляді запишемо її так: X = β+αX

За нульове наближення розв'язку системи виберемо стовпець вільних членів, тобто β.

Перше наближення розв'язку системи знаходимо у вигляді X = β +α · X . (6.8) Аналогічно робимо довільне наближення розв'язку системи для наступних ітерацій.

6.2. Збіжність ітераційного процесу

Збіжність ітераційного процесу залежить від величини коефіцієнтів матриці α . Тому не обов'язково за нульове наближення розв'язку вибирати стовпець вільних членів. Теорема (про збіжність ітераційного процесу). Якщо елементи матриці α системи рівнянь задовольняють одну з умов:

$$\sum_{j=1}^{n}\left|\frac{a_{ij}}{a_{ii}}\right|<1,\quad i=\overline{1,n},\qquad \sum_{i,j=1}^{n}\left(\frac{a_{ij}}{a_{ii}}\right)^2<1,\qquad \sum_{i=1}^{n}\left|\frac{a_{ij}}{a_{ii}}\right|<1,\quad j=\overline{1,n},$$

6.4. Метод Зейделя

Даний метод є модифікацією методу простої ітерації. Основна його ідея полягає в тому, що при обчисленні чергового k -го наближення розв'язку використовують вже знайдені значення k -го наближеного розв'язку X. Умова збіжності і критерій припинення ітераційного процесу за методом Зейделя є такими ж, як і для методу простої ітерації. За методом Зейделя отримують кращу збіжність, ніж за методом Якобі, але доводиться проводити громіздкіші обчислення. Крім того, ітераційний процес, проведений за методом Зейделя іноді буває збіжним у тому випадку, коли він є розбіжним за методом простої ітерації і навпаки.

$$x_i^{(k)} = \beta_i + \sum_{j=1}^{i-1} \alpha_{ij} x_j^{(k)} + \sum_{j=i+1}^{n} \alpha_{ij} x_j^{(k-1)}, \quad i = \overline{1,n}, \quad k = 1,2,...$$

Індивідуальне завдання

Варіант 2

Написати програму розв'язку матриці
0.81 + 0.12 - 0.34 - 0.16 = 0,64
0.34 - 1.08 + 0.17 - 0.18 = -1.42
0.16 + 0.34 + 0.75 - 0.31 = 0.42
0.12 - 0.26 - 0.08 + 0.68 = -0.83

Хід роботи

Код програми

```
#include <iostream>
#include <iomanip>
#include <cmath>
```

```
#define SIZE 4

void printMatrix(double **A, double *B, int n);

bool convergence(double **A, double **Alpha, int size);

void methodJacobi(double **a, double *b, int n, double eps);

void methodZeydelya(double** A, double* B, int size, double eps);

int main() {

	std::cout << "Custom matrix - U\nStandard 3x3 matrix - S\nClose - E"
<< std::endl;//menu

	for (char choise = '\n'; choise != 'E'; std::cin >> choise) {
		switch (choise) {
		case 'U': {		//custom matrix
			int n, m;
			std::cout << "Enter the amount of x: ";
			std::cin >> n;

			double** a = new double* [n];
			double* b = new double[n];

			for (int i = 0; i < n; i++) {

				a[i] = new double[n];

				for (int j = 0; j < n + 1; j++) {

					if (j < n) {

						std::cout << "Enter " << i + 1 << "
row " << j + 1 << " column A-matrix element" << std::endl;
						std::cin >> (a[i][j]);
					}
					else {
						std::cout << "Enter " << i + 1 << "
row B-matrix element" << std::endl;
						std::cin >> (b[i]);
					}
				}
			}

			double eps;
```

```
			std::cout << "Enter eps:" << std::endl;
			std::cin >> eps;

			printMatrix(a, b, n);

			break;
		}
		case 'S': { //standart matrix
			double solveMatrix[SIZE] = { 0.64, -1.42, 0.42, -0.83 };
			double mainMatrixMemory[SIZE][SIZE] = {
				{0.81, 0.12, -0.34, -0.16},
				{0.34, -1.08, 0.17, -0.18},
				{0.16, 0.34, 0.75, -0.31},
				{0.12, -0.26, -0.08, 0.68} };

			double** a = new double* [SIZE];
			double* b = new double[SIZE];

			for (int i = 0; i < SIZE; i++) {
				a[i] = new double[SIZE];
				b[i] = solveMatrix[i];
				for (int j = 0; j < SIZE; j++) {
						a[i][j] = mainMatrixMemory[i][j];
				}
			}

			double eps;
			std::cout << "Enter eps: ";
			std::cin >> eps;

			std::cout << "Standart Matrix" << std::endl;
			printMatrix(a, b, SIZE);

			methodJacobi(a, b, SIZE, eps);
			methodZeydelya(a, b, SIZE, eps);

			break;
		}
		default: {  //try again
			if (choise != '\n') {
				std::cout << "Not correct sumbol" << std::endl;
			}
			break;
		}
		}
}
```

```
	return 0;
}

void printMatrix(double **A, double *B, int n) {
	std::cout << "-----------------------------------------------------" <<
std::endl;
	for (int i = 0; i < n; i++) {
		for (int j = 0; j <= n; j++) {

			char znak = '\0';

			if (j == n) {
				znak = '=';
				std::cout << znak << " ";
				std::cout << std::setw(5) << std::setprecision(4)
<< B[i];
			}
			else {
				if (j > 0) {
					((A[i][j]) >= 0) ? znak = '+' : znak = '\0';
				}
				std::cout << znak << " ";
				std::cout << std::setw(5) << std::setprecision(4)
<< A[i][j] << " ";
			}

		}
		std::cout << "\n";
	}
	std::cout << "-----------------------------------------------------" <<
std::endl;
}

bool convergence(double **A, double **Alpha, int size) {


	bool  bCheck = false;

	std::cout << "Norma 1" << std::endl;
	double dCheck = 0;
	for (int i = 0; i < size; i++) {
		dCheck = 0;
		for (int j = 0; j < size; j++) {
			dCheck += fabs(Alpha[i][j]);
		}
		bCheck = (dCheck <= 1) ? true : false;
		std::cout << dCheck - 0.09 << " < 1" << std::endl;
```

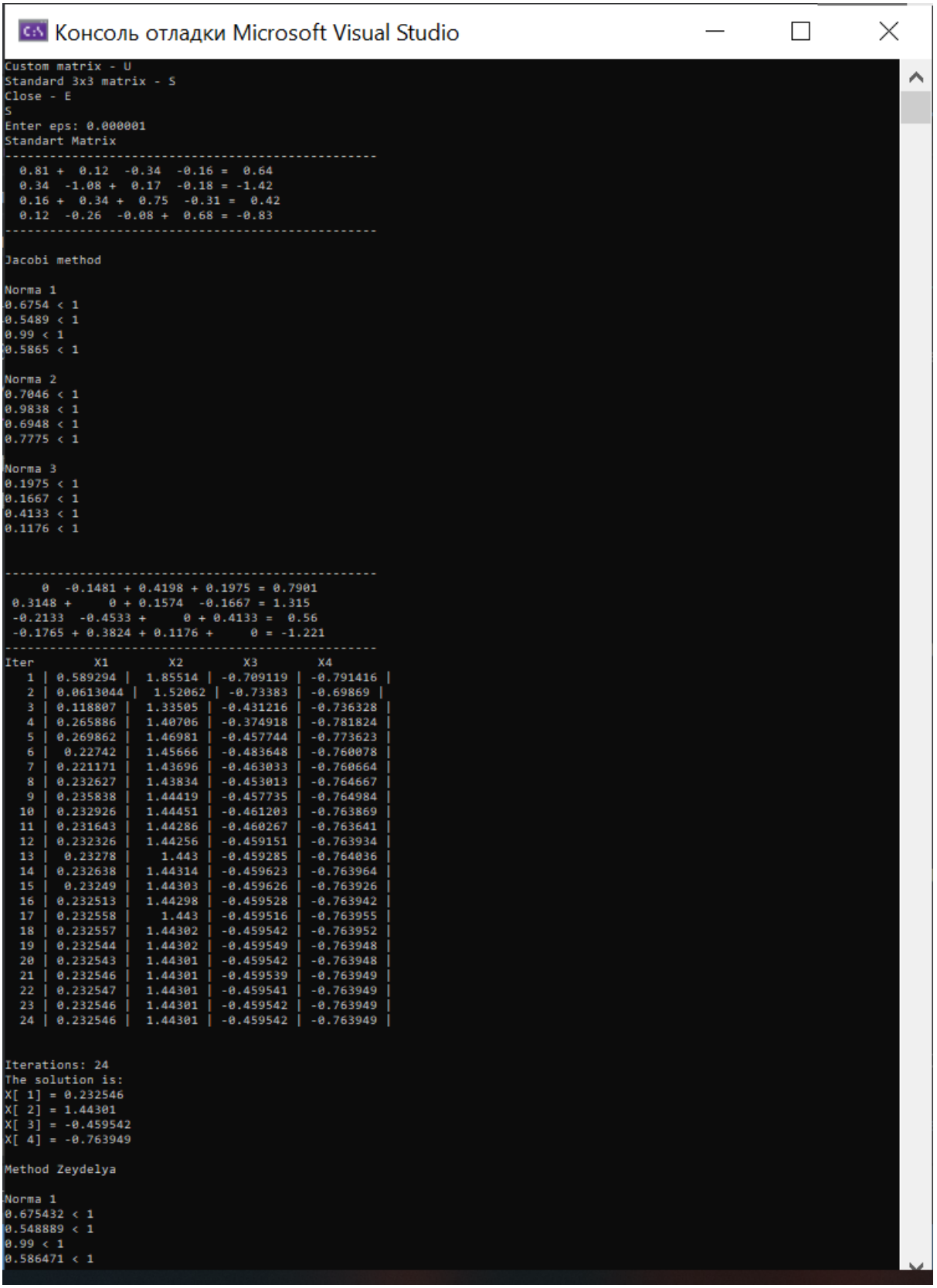
```
	}

	std::cout << "\nNorma 2" << std::endl;
	for (int i = 0; i < size; i++) {
		dCheck = 0;
		for (int j = 0; j < size; j++) {
				dCheck += fabs(Alpha[j][i]);
		}
		bCheck = (dCheck <= 1) ? true : false;
		std::cout << dCheck << " < 1" << std::endl;
	}

	std::cout << "\nNorma 3" << std::endl;
	for (int i = 0; i < size; i++) {
		dCheck = 0;
		for (int j = 0; j < size; j++) {
			if (i != j) {
				dCheck = (Alpha[i][j] * Alpha[i][j]);
			}
		}
		bCheck = (sqrt(dCheck) <= 1) ? true : false;
		std::cout << sqrt(dCheck) << " < 1" << std::endl;
	}

	std::cout << "\n" << std::endl;
	return bCheck;
}

void methodJacobi(double **A, double *B, int size, double eps) {
	std::cout << "\nJacobi method\n" << std::endl;
	double** Alpha = new double* [size];
	//---------------------------------------------------------
	for (int i = 0; i < size; i++) {
		Alpha[i] = new double[size];
		for (int j = 0; j < size; j++) {
			Alpha[i][j] = (i != j) ? (-(A[i][j] / A[i][i])) : 0;
		}
	}
	//---------------------------------------------------------

	double* Beta = new double [size];
	//---------------------------------------------------------
	for (int i = 0; i < size; i++) {
		Beta[i] = (B[i] / A[i][i]);
	}
	//---------------------------------------------------------
```

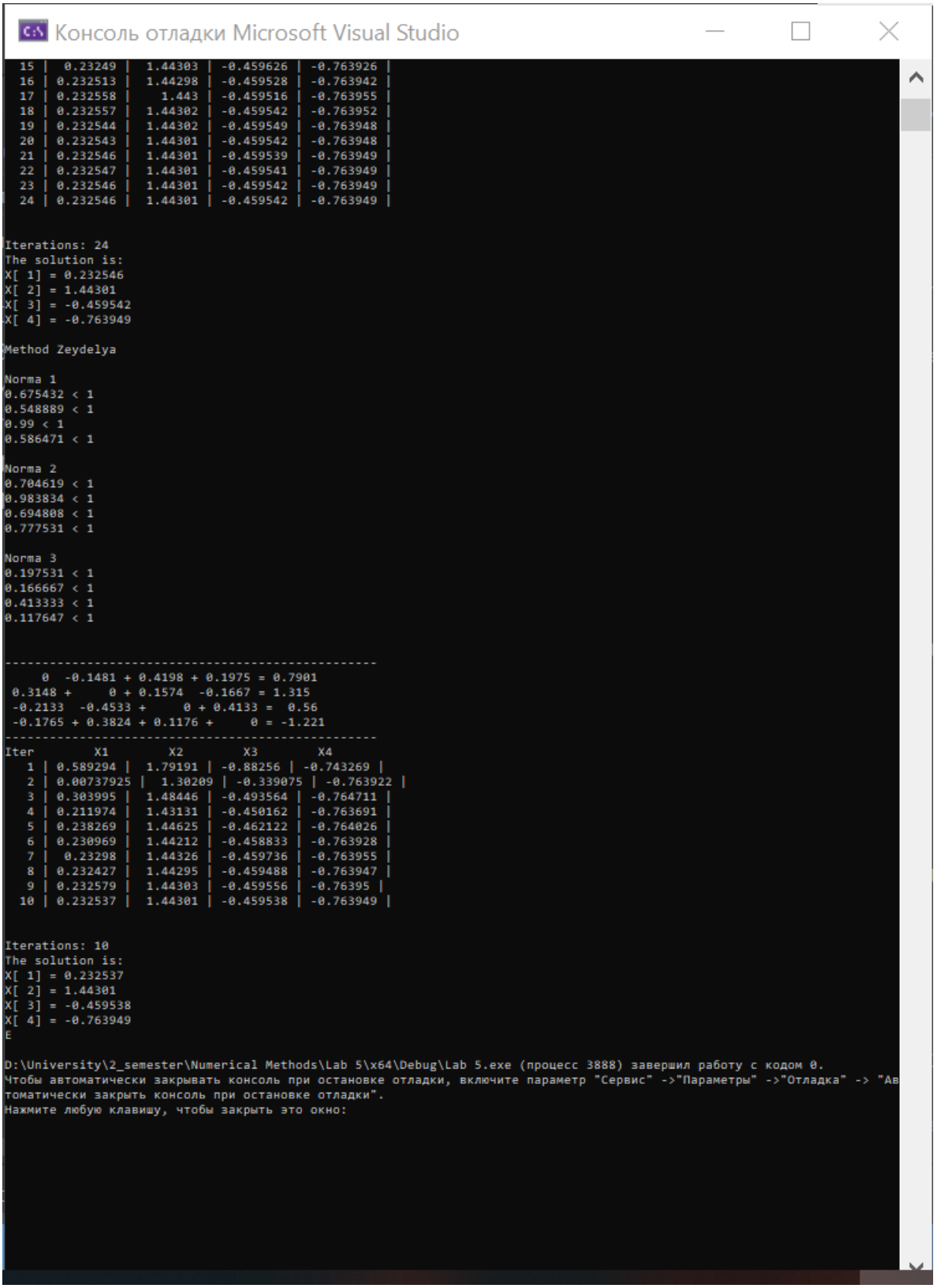
```
	if (!convergence(A, Alpha, size)) {
		std::cout << "Matrix not convergence" << std::endl;
		return;
	}

	printMatrix(Alpha, Beta, size);

	double* X1 = new double[size];
	double* X0 = new double[size];
	for (int i = 0; i < size; i++) {
		X1[i] = Beta[i];
	}

	int itr = 0;
	double max;
	double sum;
	std::cout << "Iter" << std::setw(10) << "X1" << std::setw(10) <<
"X2" << std::setw(10) << "X3" << std::setw(10) << "X4" << std::endl;

		do {
			for (int i = 0; i < size; i++) {
				sum = 0;
				for (int j = 0; j < size; j++) {
					sum += Alpha[i][j] * X1[j];
				}
				X0[i] = Beta[i] + sum;
			}

			max = fabs(X0[0] - X1[0]);
			for (int i = 1; i < size; i++)
			{
				if (fabs(X0[i] - X1[i]) > max)
					max = fabs(X0[i] - X1[i]);
			}
			for (int i = 0; i < size; i++) {
				X1[i] = X0[i];
			}

			std::cout << std::setw(4) << itr + 1 << " | ";
			for (int i = 0; i < size; i++) {
				std::cout << std::setw(8) << std::setprecision(-4)
<< X1[i] << " | ";
			}
			std::cout << "\n";

			itr++;
		} while (max >= eps);
```

```
	std::cout << "\n\nIterations: " << itr << std::endl;
	std::cout << "The solution is:\n";
	for (int i = 0; i < size; i++)
		std::cout << "X[" << std::setw(2) << i + 1 << "] = " <<
std::setw(7) << std::setprecision(-4) << X1[i] << std::endl;
}

void methodZeydelya(double** A, double* B, int size, double eps) {
	std::cout << "\nMethod Zeydelya\n" << std::endl;
	double** Alpha = new double* [size];
	//-----------------------------------------------------------
	for (int i = 0; i < size; i++) {
		Alpha[i] = new double[size];
		for (int j = 0; j < size; j++) {
			Alpha[i][j] = (i != j) ? (-(A[i][j] / A[i][i])) : 0;
		}
	}
	//-----------------------------------------------------------

	double* Beta = new double[size];
	//-----------------------------------------------------------
	for (int i = 0; i < size; i++) {
		Beta[i] = (B[i] / A[i][i]);
	}
	//-----------------------------------------------------------

	if (!convergence(A, Alpha, size)) {
		std::cout << "Matrix not convergence" << std::endl;
		return;
	}

	printMatrix(Alpha, Beta, size);

	double* X0 = new double[size];
	double* X1 = new double[size];
	for (int i = 0; i < size; i++) {
		X1[i] = Beta[i];
	}

	int itr = 0;
	double max;
	double sum;
	std::cout << "Iter" << std::setw(10) << "X1" << std::setw(10) <<
"X2" << std::setw(10) << "X3" << std::setw(10) << "X4" << std::endl;
	do {
		for (int i = 0; i < size; i++) {
```

```
                for (int i = 0; i < size; i++) {
                      X0[i] = X1[i];
                }
                X1[i] = Beta[i];
                for (int j = i + 1; j < size; j++) {
                       X1[i] += Alpha[i][j] * X1[j];
                }
                for (int j = 0; j <= i-1 ; j++) {
                      X1[i] += Alpha[i][j] * X0[j];
                }

          }

          max = 0;
          for (int i = 0; i < size; i++)
          {
                if (fabs(X0[i] - X1[i]) > max)
                      max = fabs(X0[i] - X1[i]);
          }

          std::cout << std::setw(4) << itr + 1 << " | ";
          for (int i = 0; i < size; i++) {
                std::cout << std::setw(8) << std::setprecision(-4) <<
X1[i] << " | ";
          }
          std::cout << "\n";

          itr++;
     } while (max >= eps);

     std::cout << "\n\nIterations: " << itr << std::endl;
     std::cout << "The solution is:\n";
     for (int i = 0; i < size; i++)
          std::cout << "X[" << std::setw(2) << i + 1 << "] = " <<
std::setw(7) << std::setprecision(-4) << X1[i] << std::endl;
}
```

Результат роботи


Консоль отладки Microsoft Visual Studio
Custom matrix - U
Standard 3x3 matrix - S
Close - E
S
Enter eps: 0.000001
Standart Matrix
0.81 + 0.12 -0.34 -0.16 = 0.64
0.34 -1.08 + 0.17 -0.18 = -1.42
0.16 + 0.34 + 0.75 -0.31 = 0.42
0.12 -0.26 -0.08 + 0.68 = -0.83
Jacobi method
Norma 1
0.6754 < 1
0.5489 < 1
0.99 < 1
0.5865 < 1
Norma 2
0.7046 < 1
0.9838 < 1
0.6948 < 1
0.7775 < 1
Norma 3
0.1975 < 1
0.1667 < 1
0.4133 < 1
0.1176 < 1
0 -0.1481 + 0.4198 + 0.1975 = 0.7901
0.3148 + 0 + 0.1574 -0.1667 = 1.315
-0.2133 -0.4533 + 0 + 0.4133 = 0.56
-0.1765 + 0.3824 + 0.1176 + 0 = -1.221
Iter X1 X2 X3 X4
1 | 0.589294 | 1.85514 | -0.709119 | -0.791416 |
2 | 0.0613044 | 1.52062 | -0.73383 | -0.69869 |
3 | 0.118807 | 1.33505 | -0.431216 | -0.736328 |
4 | 0.265886 | 1.40706 | -0.374918 | -0.781824 |
5 | 0.269862 | 1.46981 | -0.457744 | -0.773623 |
6 | 0.22742 | 1.45666 | -0.483648 | -0.760078 |
7 | 0.221171 | 1.43696 | -0.463033 | -0.760664 |
8 | 0.232627 | 1.43834 | -0.453013 | -0.764667 |
9 | 0.235838 | 1.44419 | -0.457735 | -0.764984 |
10 | 0.232926 | 1.44451 | -0.461203 | -0.763869 |
11 | 0.231643 | 1.44286 | -0.460267 | -0.763641 |
12 | 0.232326 | 1.44256 | -0.459151 | -0.763934 |
13 | 0.23278 | 1.443 | -0.459285 | -0.764036 |
14 | 0.232638 | 1.44314 | -0.459623 | -0.763964 |
15 | 0.23249 | 1.44303 | -0.459626 | -0.763926 |
16 | 0.232513 | 1.44298 | -0.459528 | -0.763942 |
17 | 0.232558 | 1.443 | -0.459516 | -0.763955 |
18 | 0.232557 | 1.44302 | -0.459542 | -0.763952 |
19 | 0.232544 | 1.44302 | -0.459549 | -0.763948 |
20 | 0.232543 | 1.44301 | -0.459542 | -0.763948 |
21 | 0.232546 | 1.44301 | -0.459539 | -0.763949 |
22 | 0.232547 | 1.44301 | -0.459541 | -0.763949 |
23 | 0.232546 | 1.44301 | -0.459542 | -0.763949 |
24 | 0.232546 | 1.44301 | -0.459542 | -0.763949 |
Iterations: 24
The solution is:
X[1] = 0.232546
X[2] = 1.44301
X[3] = -0.459542
X[4] = -0.763949
Method Zeydelya
Norma 1
0.675432 < 1
0.548889 < 1
0.99 < 1
0.586471 < 1


Консоль отладки Microsoft Visual Studio
15 | 0.23249 | 1.44303 | -0.459626 | -0.763926 |
16 | 0.232513 | 1.44298 | -0.459528 | -0.763942 |
17 | 0.232558 | 1.443 | -0.459516 | -0.763955 |
18 | 0.232557 | 1.44302 | -0.459542 | -0.763952 |
19 | 0.232544 | 1.44302 | -0.459549 | -0.763948 |
20 | 0.232543 | 1.44301 | -0.459542 | -0.763948 |
21 | 0.232546 | 1.44301 | -0.459539 | -0.763949 |
22 | 0.232547 | 1.44301 | -0.459541 | -0.763949 |
23 | 0.232546 | 1.44301 | -0.459542 | -0.763949 |
24 | 0.232546 | 1.44301 | -0.459542 | -0.763949 |
Iterations: 24
The solution is:
X[1] = 0.232546
X[2] = 1.44301
X[3] = -0.459542
X[4] = -0.763949
Method Zeydelya
Norma 1
0.675432 < 1
0.548889 < 1
0.99 < 1
0.586471 < 1
Norma 2
0.704619 < 1
0.983834 < 1
0.694808 < 1
0.777531 < 1
Norma 3
0.197531 < 1
0.166667 < 1
0.413333 < 1
0.117647 < 1
0 -0.1481 + 0.4198 + 0.1975 = 0.7901
0.3148 + 0 + 0.1574 -0.1667 = 1.315
-0.2133 -0.4533 + 0 + 0.4133 = 0.56
-0.1765 + 0.3824 + 0.1176 + 0 = -1.221
Iter X1 X2 X3 X4
1 | 0.589294 | 1.79191 | -0.88256 | -0.743269 |
2 | 0.00737925 | 1.30209 | -0.339075 | -0.763922 |
3 | 0.303995 | 1.48446 | -0.493564 | -0.764711 |
4 | 0.211974 | 1.43131 | -0.450162 | -0.763691 |
5 | 0.238269 | 1.44625 | -0.462122 | -0.764026 |
6 | 0.230969 | 1.44212 | -0.458833 | -0.763928 |
7 | 0.23298 | 1.44326 | -0.459736 | -0.763955 |
8 | 0.232427 | 1.44295 | -0.459488 | -0.763947 |
9 | 0.232579 | 1.44303 | -0.459556 | -0.76395 |
10 | 0.232537 | 1.44301 | -0.459538 | -0.763949 |
Iterations: 10
The solution is:
X[1] = 0.232537
X[2] = 1.44301
X[3] = -0.459538
X[4] = -0.763949
E
D:\University\2_semester\Numerical Methods\Lab 5\x64\Debug\Lab 5.exe (процесс 3888) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Ав
томатически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно:

Висновок

Виконуючи лабораторну роботу №5, я ознайомився на практиці з методами Якобі та Зейделя при розв’язування систем лінійних алгебраїчних рівнянь.