Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра Автоматизованих Систем Обробки Інформації та Управління

Лабораторна робота №3

з дисципліни «Спеціальні розділи математики»

на тему

«Розв’язання систем лінійних алгебраїчних рівнянь (СЛАР) ітераційними

методами. Метод простої ітерації. Метод Зейделя»

Виконав:

студент гр. ІС-02

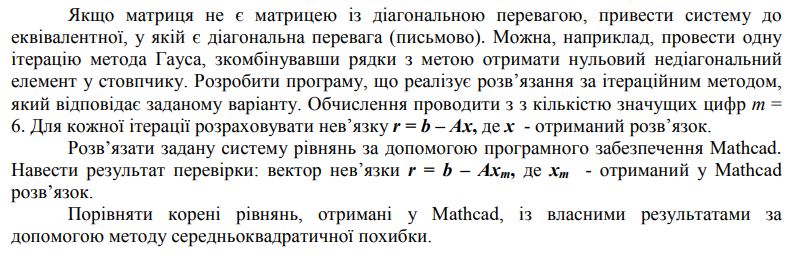
Плостак Ілля

Викладач:

доц. Рибачук Л.В.

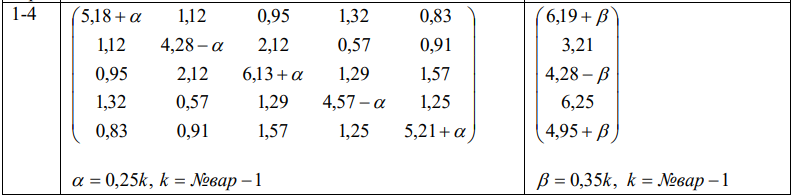
Київ – 2021

**1. Завдання**





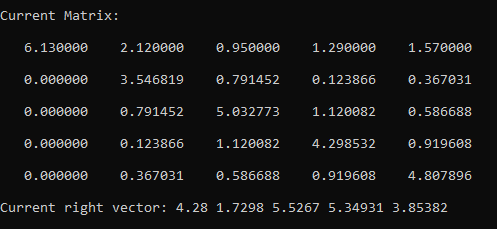
**2. Система рівнянь**



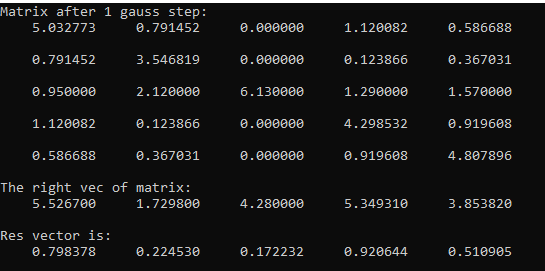
Варіант 1; k=0

**3. Результати виконання програми**

а) Приведення матриці до матриці за діагональною перевагою (один крок методу Гауса)



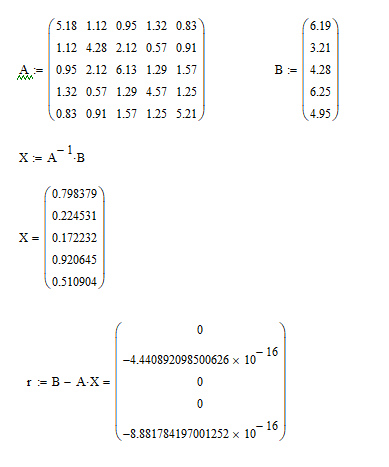
б) Рішення методом Зейделя:



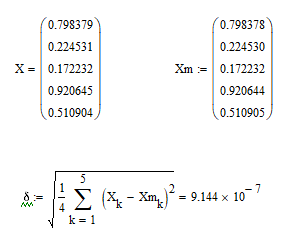
**4. Вектор нев’язки**



**5. Копія розв’язку задачі у Mathcad; вектор нев’язки для цього розв’язку**



**6. Порівняння власного розв’язку та розв’язку, отриманого у Mathcad**



**7. Лістинг програми**

[main.cpp](https://github.com/feedblackg44/kpilabs2/blob/master/CHM/Lab3/Lab3/main.cpp):

#include <iostream>

#include "functions.h"

using namespace std;

int main()

{

int size = 5;

int\* numbersX = new int[size];

fillInSeq(numbersX, size);

double\*\* A = createMatrix(size);

matrix5Init(A);

cout << "Matrix after 1 gauss step:\n";

print(A, size);

double\* B = createVec(size);

vector5Init(B);

cout << "The right vec of matrix:\n";

print(B, size);

double\* X = createVec(size);

vector5Init(X);

zeidelMethodSolve(A, B, X, size);

cout << "\nRes vector is:\n";

print(X, size);

cout << "\nr = b - Ax:\n";

print(vecDelta(A, B, X, size), size, 15);

system("pause");

return 0;

}

[functions.h](https://github.com/feedblackg44/kpilabs2/blob/master/CHM/Lab3/Lab3/functions.h):

#pragma once

double\*\* createMatrix(int size);

double\* createVec(int size);

void fillInSeq(int\* Vector, int size);

void matrix5Init(double\*\* M);

void vector5Init(double\* V);

void print(double\*\* Matrix, int size);

void print(double\* Vector, int size, int prec = 6);

void print(double number);

double sumOfMatrixElements(double\*\* matrix, double\* X, int firstNum, int secondNum, int i);

void zeidelMethodSolve(double\*\* matrix, double\* B, double\* X, int size);

double maxInArrays(double\* arr1, double\* arr2, int size);

void copyVec(double\* vecCopyFrom, double\* vecToCopy, int size);

double\* MatrixVecMult(double\*\* matrix1, double\* matrix2, int size);

double\* vecDelta(double\*\* Matrix, double\* rightVec, double\* X, int size);

[function.cpp](https://github.com/feedblackg44/kpilabs2/blob/master/CHM/Lab3/Lab3/functions.cpp):

#include <iostream>

#include <iomanip>

#include "functions.h"

using namespace std;

double\*\* createMatrix(int size)

{

double\*\* matrix = new double\* [size];

for (int i = 0; i < size; i++)

{

matrix[i] = new double[size];

}

return matrix;

}

double\* createVec(int size)

{

double\* vector = new double[size];

return vector;

}

void fillInSeq(int\* Vector, int size)

{

for (int i = 0; i < size; i++)

{

Vector[i] = i;

}

}

void matrix5Init(double\*\* M)

{

M[0][0] = 5.032773; M[0][1] = 0.791452; M[0][2] = 0.0; M[0][3] = 1.120082; M[0][4] = 0.586688;

M[1][0] = 0.791452; M[1][1] = 3.546819; M[1][2] = 0.0; M[1][3] = 0.123866; M[1][4] = 0.367031;

M[2][0] = 0.95; M[2][1] = 2.12; M[2][2] = 6.13; M[2][3] = 1.29; M[2][4] = 1.57;

M[3][0] = 1.120082; M[3][1] = 0.123866; M[3][2] = 0.0; M[3][3] = 4.298532; M[3][4] = 0.919608;

M[4][0] = 0.586688; M[4][1] = 0.367031; M[4][2] = 0.0; M[4][3] = 0.919608; M[4][4] = 4.807896;

}

void vector5Init(double\* V)

{

V[0] = 5.5267;

V[1] = 1.7298;

V[2] = 4.28;

V[3] = 5.34931;

V[4] = 3.85382;

}

void print(double\*\* Matrix, int size)

{

for (int i = 0; i < size; i++)

{

for (int j = 0; j < size; j++)

{

cout << fixed << setw(12) << setprecision(6) << Matrix[i][j] << " ";

}

cout << "\n" << endl;

}

cout << defaultfloat;

}

void print(double\* Vector, int size, int prec)

{

for (int i = 0; i < size; i++)

{

cout << fixed << setw(prec + 6) << setprecision(prec) << Vector[i] << " ";

}

cout << endl;

cout << defaultfloat;

}

void print(double number)

{

cout << fixed << setw(21) << setprecision(15) << number << endl;

cout << defaultfloat;

}

double sumOfMatrixElements(double\*\* matrix, double\* X, int firstNum, int secondNum, int i)

{

double sum = 0.0;

for (int j = firstNum; j < secondNum; j++)

{

sum += matrix[i][j] / matrix[i][i] \* X[j];

}

return sum;

}

void zeidelMethodSolve(double\*\* matrix, double\* B, double\* X, int size)

{

double\* tempX = new double[size];

double e = 0.000001;

do

{

copyVec(X, tempX, size);

for (int i = 0; i < size; i++)

{

X[i] = -sumOfMatrixElements(matrix, X, 0, i, i) - sumOfMatrixElements(matrix, tempX, i + 1, size, i) + B[i] / matrix[i][i];

}

} while (maxInArrays(X, tempX, size) >= e);

}

double maxInArrays(double\* arr1, double\* arr2, int size)

{

double max = abs(arr1[0] - arr2[0]);

for (int i = 1; i < size; i++)

{

if (abs(arr1[i] - arr2[i]) > max)

max = abs(arr1[i] - arr2[i]);

}

return max;

}

void copyVec(double\* vecCopyFrom, double\* vecToCopy, int size)

{

for (int i = 0; i < size; i++)

{

vecToCopy[i] = vecCopyFrom[i];

}

}

double\* MatrixVecMult(double\*\* matrix1, double\* matrix2, int size)

{

double\* outMatrix = new double[size];

for (int i = 0; i < size; i++)

{

outMatrix[i] = 0;

for (int k = 0; k < size; k++)

{

outMatrix[i] += matrix1[i][k] \* matrix2[k];

}

}

return outMatrix;

}

double\* vecDelta(double\*\* Matrix, double\* rightVec, double\* X, int size)

{

double\* Ax = MatrixVecMult(Matrix, X, size);

double\* r = new double[size];

for (int i = 0; i < size; i++)

{

r[i] = rightVec[i] - Ax[i];

}

return r;

}