**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2**

**ПРИБЛИЖЕННЫЕ МЕТОДЫ РЕШЕНИЯ**

**НЕЛИНЕЙНЫХ УРАВНЕНИЙ**

**(Вариант 9)**

*Выполнил студент 3 курса МОиАИС*

*Ходосевич Данила*

*Задания к работе*.

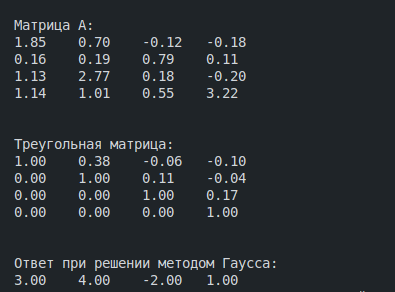
Написать, отладить и выполнить программы решения систем линейных алгебраических уравнений, записанных в векторно-матричной форме и приведенных в таблице. В колонке х\* приведено точное решение. Решить систему методом Гаусса с выбором главного элемента и методом Зейделя.



Оценить погрешности методов.

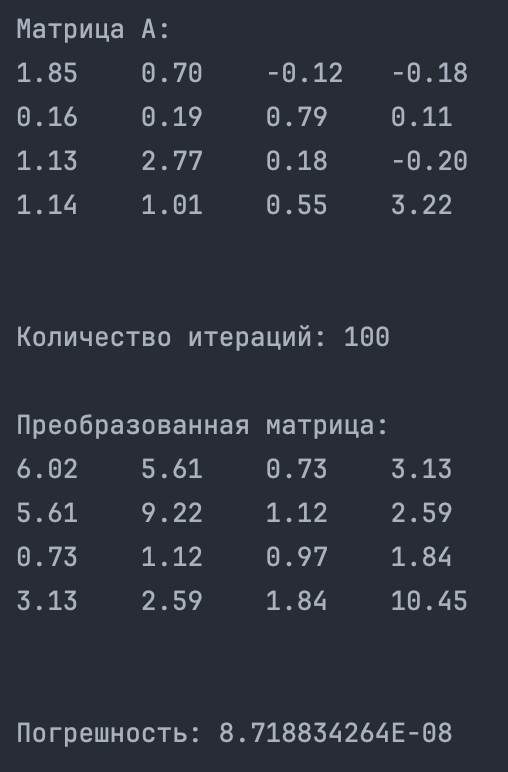
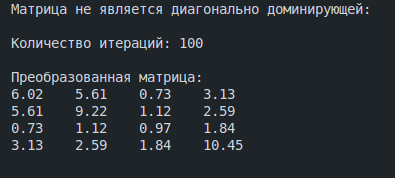
**Для метода Гаусса привести матрицу, приведенную к треугольному виду. Для метода Зейделя - преобразованную матрицу и количество итераций. Показать, что условия сходимости выполнены.**

Метод Гаусса:



Погрешность метода Гаусса соответсвенно равна 0, поскольку метод точный.

Метод Зейделя:



Погрешность высчитана как разница погрешности последней итерации и предпоследней итерации

Погрешность в методе Зейделя оценивается, как разность последней и предпоследней итерации.

**Приложение**

#include <iostream>

#include <vector>

#include <cmath>

#include <iomanip>

const int N = 4;

const double EPS = 1e-7;

const int MAX\_ITERATIONS = 100;

typedef std::vector<std::vector<double>> Matrix;

typedef std::vector<double> Vector;

void printMatrix(const Matrix& matrix) {

for (const auto& row : matrix) {

for (double val : row) {

std::cout << std::fixed << std::setprecision(2) << val << "\t";

}

std::cout << "\n";

}

std::cout << "\n";

}

void printVector(const Vector& vector) {

for (double val : vector) {

std::cout << std::fixed << std::setprecision(2) << val << "\t";

}

std::cout << "\n";

}

void swapRows(Matrix& matrix, Vector& vector, int row1, int row2) {

if (row1 == row2) return;

std::swap(matrix[row1], matrix[row2]);

std::swap(vector[row1], vector[row2]);

}

Vector gauss(Matrix A, Vector b) {

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

int maxRow = i;

for (int k = i + 1; k < N; ++k) {

if (std::fabs(A[k][i]) > std::fabs(A[maxRow][i])) {

maxRow = k;

}

}

swapRows(A, b, i, maxRow);

double leader = A[i][i];

for (int j = 0; j < N; ++j) {

A[i][j] /= leader;

}

b[i] /= leader;

for (int j = i + 1; j < N; ++j) {

double factor = A[j][i];

for (int k = i; k < N; ++k) {

A[j][k] -= factor \* A[i][k];

}

b[j] -= factor \* b[i];

}

}

Vector solution(N, 0.0);

for (int i = N - 1; i >= 0; --i) {

double sum = b[i];

for (int j = i + 1; j < N; ++j) {

sum -= A[i][j] \* solution[j];

}

solution[i] = sum;

}

std::cout << "\nТреугольная матрица:\n";

printMatrix(A);

return solution;

}

bool isDiagonallyDominant(const Matrix& A) {

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

double sumRow = 0.0;

for (int j = 0; j < N; ++j) {

if (j != i) {

sumRow += std::fabs(A[i][j]);

}

}

if (std::fabs(A[i][i]) <= sumRow) {

return false;

}

}

return true;

}

void makeDiagonallyDominant(Matrix& A, Vector& b) {

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

int maxIndex = i;

for (int k = i + 1; k < N; ++k) {

if (std::fabs(A[k][i]) > std::fabs(A[maxIndex][i])) {

maxIndex = k;

}

}

swapRows(A, b, i, maxIndex);

}

}

Vector zeidel(Matrix A, Vector b, Vector x) {

if (!isDiagonallyDominant(A)) {

std::cout << "Матрица не является диагонально доминирующей, перестановка строк...\n";

makeDiagonallyDominant(A, b);

if (!isDiagonallyDominant(A)) {

std::cout << "Не удалось сделать матрицу диагонально доминирующей.\n";

return {};

}

}

int iterations = 0;

double maxInaccuracy = 0.00;

for (int iter = 0; iter < MAX\_ITERATIONS; ++iter) {

Vector x\_new = x;

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

double sum1 = 0.0, sum2 = 0.0;

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

sum1 += A[i][j] \* x\_new[j];

}

for (int j = i + 1; j < N; ++j) {

sum2 += A[i][j] \* x[j];

}

x\_new[i] = (b[i] - sum1 - sum2) / A[i][i];

}

maxInaccuracy = 0.0000000000000000000;

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

maxInaccuracy = std::max(maxInaccuracy, std::fabs(x\_new[i] - x[i]));

}

x = x\_new;

++iterations;

if (maxInaccuracy < EPS) {

break;

}

}

std::cout << "\nКоличество итераций: " << iterations << "\n";

std::cout << "\nПогрешность: " << maxInaccuracy << "\n";

return x;

}

int main() {

Matrix A = {

{1.85, 0.70, -0.12, -0.18},

{0.16, 0.19, 0.79, 0.11},

{1.13, 2.77, 0.18, -0.20},

{1.14, 1.01, 0.55, 3.22}

};

Vector b = {8.41, -0.23, 13.91, 9.58};

std::cout << "Матрица A:\n";

printMatrix(A);

Vector x\_gauss = gauss(A, b);

std::cout << "\nОтвет при решении методом Гаусса:\n";

printVector(x\_gauss);

Vector x\_zeidel(N, 0.0);

Vector x\_result = zeidel(A, b, x\_zeidel);

if (!x\_result.empty()) {

std::cout << "\nОтвет при решении методом Зейделя:\n";

printVector(x\_result);

}

return 0;

}