**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2**

**ПРИБЛИЖЕННЫЕ МЕТОДЫ РЕШЕНИЯ**

**НЕЛИНЕЙНЫХ УРАВНЕНИЙ**

**(Вариант 10)**

*Выполнил студент 3 курса МОиАИС*

*Соколов Арсений*

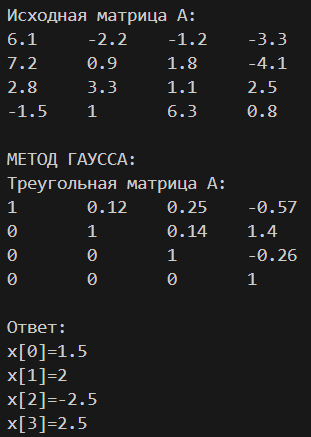
**Задание:** написать, отладить и выполнить программы решения систем линейных алгебраических уравнений, записанных в векторно-матричной форме и приведенных в таблице. В колонке х\* приведено точное решение. Решить систему методом Гаусса с выбором главного элемента и методом Зейделя.

Оценить погрешности методов.

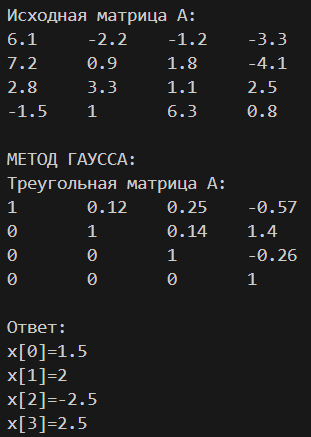
Для метода Гаусса привести матрицу, приведенную к треугольному виду. Для метода Зейделя - преобразованную матрицу и количество итераций. Показать, что условия сходимости выполнены.

***Метод Гаусса***

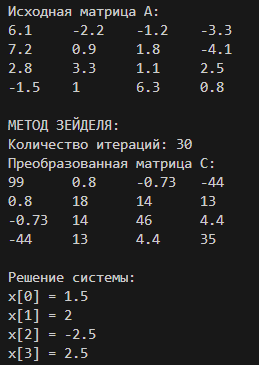
Матрица, приведенная к треугольному виду (код программы в приложении 1)



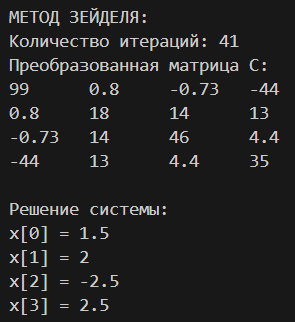
Решение СЛАУ



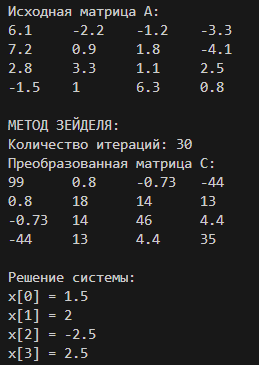
***Метод Зейделя***

Преобразованная матрица (код программы в приложении 2)****

Количество итераций

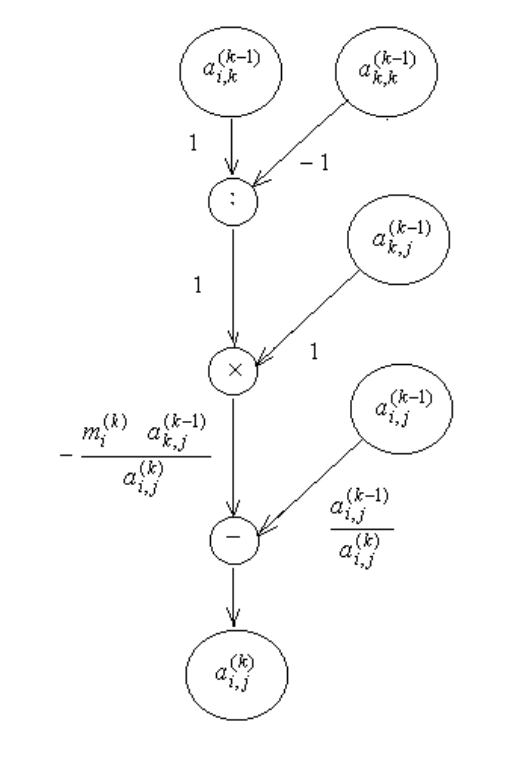
****

Решение СЛАУ

****

**Погрешность:**

**Метод Гаусса** является точным, поэтому **погрешности метода не существует**. Граф накопления погрешности вычисления в методе Гаусса:

****

Погрешность решения системы уравнений в **методе Зейделя** можно оценить путем сравнения приближения к решению на последней итерации и предыдущей. В нашем случае **погрешность равна 0,00096:**

****

**ПРИЛОЖЕНИЕ 1**

***Программа решения СЛАУ методом Гаусса***

#include <iostream>

#include <iomanip>

const int row = 4, column = 4;

void printMatrix(double A[row][column]) {

    for (int i = 0; i < row; i++) {

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

            std::cout << std::setprecision(2) << A[i][j] << "\t";

        }

        std::cout << std::endl;

    }

    std::cout << std::endl;

}

void Gauss(double A[row][column], double b[column], double x[column]){

    int k = 0, index = 0;

    double max;

    while (k < column){

        // Поиск строки с максимальным коэффициентом A[i][k]

        max = std::abs(A[k][k]);

        index = k;

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

            if (std::abs(A[i][k]) > max){

                max = std::abs(A[i][k]);

                index = i;

            }

        }

        // Перестановка строк

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

            std::swap(A[k][j], A[index][j]);

        }

        std::swap(b[k], b[index]);

        // Нормализация уравнений

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

            double temp = A[i][k];

            if (temp == 0){ // нулевой коэффициент

                continue;

            }

            for (int j = k; j < column; j++){

                A[i][j] /= temp;

            }

            b[i] /= temp;

            if (i == k){ // само же уравнение

                continue;

            }

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

                A[i][j] -= A[k][j];

            }

            b[i] -= b[k];

        }

        k++;

    }

    std::cout << "Треугольная матрица A: " << std::endl;

    printMatrix(A);

    // Обратная подстановка

    for (k = column - 1; k >= 0; k--) {

        x[k] = b[k];

        for (int i = 0; i < k; i++){

            b[i] -= A[i][k] \* x[k];

        }

    }

    // Вывод

    std::cout << "Ответ:" << std::endl;

    for (int i = 0; i < column; i++){

        std::cout << "x[" << i << "]=" << x[i] << std::endl;

    }

}

int main() {

    std::cout << std::endl;

    double A[row][column] = {

        {6.1, -2.2, -1.2, -3.3},

        {7.2, 0.9, 1.8, -4.1},

        {2.8, 3.3, 1.1, 2.5},

        {-1.5, 1.0, 6.3, 0.8}

    };

    double b[column] = {-0.50, -2.15, 14.30, -14.00};

    double x[column];

    std::cout << "Исходная матрица A: " << std::endl;

    printMatrix(A);

    std::cout << "МЕТОД ГАУССА: " << std::endl;

    Gauss(A, b, x);

    return 0;

}

**ПРИЛОЖЕНИЕ 2**

***Программа решения СЛАУ методом Зейделя***

include <iostream>

#include <iomanip>

const int row = 4, column = 4;

void printMatrix(double A[row][column]) {

    for (int i = 0; i < row; i++) {

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

            std::cout << std::setprecision(2) << A[i][j] << "\t";

        }

        std::cout << std::endl;

    }

    std::cout << std::endl;

}

void transposeMatrix(double A[row][column], double AT[column][row]) {

    for (int i = 0; i < row; i++) {

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

            AT[j][i] = A[i][j];

        }

    }

}

void Seidel(double A[row][column], double b[column], double x[column]) {

    double AT[column][row];

    double C[column][column];

    double d[column];

    transposeMatrix(A, AT); // транспонируем матрицу A

    // C = AT \* A

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

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

            C[i][j] = 0;

            for (int k = 0; k < row; k++) {

                C[i][j] += AT[i][k] \* A[k][j];

            }

        }

    }

    // d = AT \* b

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

        d[i] = 0;

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

            d[i] += AT[i][j] \* b[j];

        }

    }

    // Итерации

    double epsilon = 0.001; // точность

    int maxIterations = 100; // максимальное количество итераций

    int iterations = 0; // счетчик итераций

    double error, maxError; // погрешность

    for (int iter = 0; iter < maxIterations; iter++) {

        double x\_new[column];

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

            x\_new[i] = d[i];

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

                if (j != i) {

                    x\_new[i] -= C[i][j] \* x\_new[j];

                }

            }

            x\_new[i] /= C[i][i];

        }

        // Проверка на сходимость

        error, maxError = 0;

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

            error = std::abs(x[i] - x\_new[i]);

            maxError = std::max(error, maxError); // Обновляем значение погрешности

        }

        if (maxError < epsilon) {

            break;

        }

        // Обновляем значения вектора x

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

            x[i] = x\_new[i];

        }

        iterations++;

    }

    std::cout << "Количество итераций: " << iterations << std::endl;

    std::cout << "Преобразованная матрица C:" << std::endl;

    printMatrix(C);

    std::cout << "Решение системы:" << std::endl;

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

        std::cout << "x[" << i << "] = " << x[i] << std::endl;

    }

    std::cout << std::endl << "Максимальный модуль разности последней и предпоследней итерации:" << maxError << std::endl;

}

int main() {

    std::cout << std::endl;

    double A[row][column] = {

        {6.1, -2.2, -1.2, -3.3},

        {7.2, 0.9, 1.8, -4.1},

        {2.8, 3.3, 1.1, 2.5},

        {-1.5, 1.0, 6.3, 0.8}

    };

    double b[column] = {-0.50, -2.15, 14.30, -14.00};

    double x[column];

    std::cout << "Исходная матрица A: " << std::endl;

    printMatrix(A);

    std::cout << "МЕТОД ЗЕЙДЕЛЯ:" << std::endl;

    Seidel(A, b, x);

    return 0;

}