Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий Кафедра прикладной математики

Отчет защищен с оценкой

Преподаватель

(подпись)

«\_ » 2023 г.

Отчет

По лабораторной работе №2

по дисциплине «Вычислительные алгоритмы»

Студент группы

ПИ-02 Чередов Р.А.

Преподаватель

Проскурин А.В.

Барнаул 2023

# Задание

* Составить программу для решения системы линейных алгебраических уравнений методами Якоби и Зейделя, Исходные данные — матрица системы уравнений и столбец свободных членов, точность е должны читаться из файла, а результаты расчетов помещаться в файл. Предусмотреть вывод числа итераций, необходимых для получения решения с заданной точностью, нормы невязки. Предусмотреть возможность автоматической генерации матрицы системы уравнений и правой части для тестирования методов на системах большой размерности. Для тестирования программы можно использовать пример из предыдущей работы.
* Исследовать зависимость числа итераций от начального приближения, точности, выбора метода решения.
* Изучить влияние на сходимость величины диагонального преобладания матрицы, то есть величины отношения суммы модулей недиагональных элементов строки к модулю диагонального элемента.
* Подобрать примеры, показывающие, что диагональное преобладание не является необходимым условием сходимости.

# Теория

Итерационные методы

Его суть заключается в нахождении по приближенному значению величины следующего приближения, которое является более точным. Метод позволяет получить значения корней системы с заданной точностью в виде предела последовательности некоторых векторов (итерационный процесс). Характер сходимости и сам факт сходимости метода зависит от выбора начального приближения корня x0.

Решение системы является пределом n →∞ вычислимой последовательности x(n), где n- номер итерации(приближения). В зависимости от установленной точности e>0 достигается конечное приближение, позволяющее утверждать, что x – точное (искомое) решение и x(n) – полученное конечное приближение эквивалентны x~ x(n).

Для нахождения приближения к данному пределу, при применении итерационного метода, должно выполняться достаточное условие сходимости для построения СЛАУ матрицы  - строгое диагональное преобладание. При выполнении этого условия, можно считать, что СЛАУ имеет единственно существующее решение, а матрица не вырождена: det |A| ≠ 0.

Метод Якоби и Зейделя являются итерационными методами для решения СЛАУ.

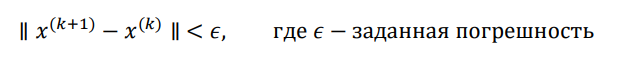
Представим систему в следующем виде:

X= Dx+c

Тогда решение системы мы можем найти как предел последовательности

X(k+1)=Dx(k)+c

Для нахождения решения системы необходимо использовать следующее условие:



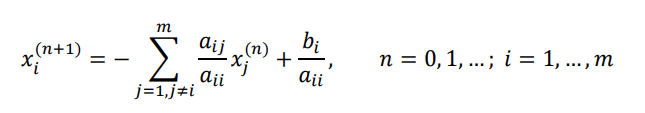
При переходе к условию для нахождения решения системы можно воспользоваться методами Якоби и Зейделя.

Метод Якоби

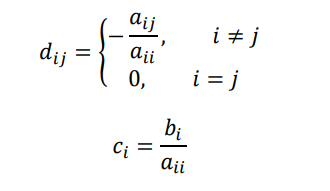
Метод Якоби – один из наиболее простых методов приведения системы матрицы к виду, удобному для итерации: из 1-го уравнения матрицы выражаем неизвестное x1, из 2-го выражаем неизвестное x2 и т.д.

Если все диагональные элементы не равны нулю, то систему можно получить.

Нужно разделить i-тое уравнение, при i= 1,..,m на aij, получаем итерационную формулу.



А элементы матрицы D и C можно представить в виде:

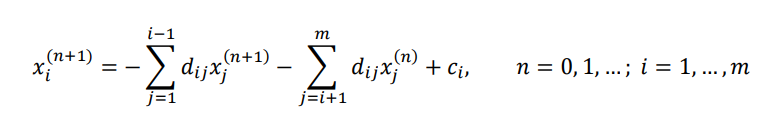


Метод Зейделя

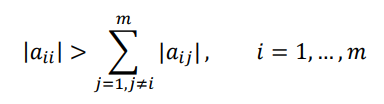
Метод Зейделя – это модификация метода Якоби

По сути, при вычислении очередного (n+1)−го приближения к неизвестному xi при i>1 используют уже найденные (n+1)−е приближения к неизвестным x1, x2, …, xi-1, а не n−ое приближение, как в методе Якоби

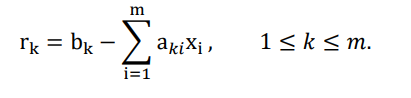
Данным способом мы на каждой итерации используем уже вычисленные значения. При этом и изменяется сама итерационная формула с использованием элементов матриц D и C:



При этом для сходимости обоих методов достаточно выполнения следующего условия (диагональное преобладание):



Невязку находим по следующей формуле:



Ошибка округления – с каждым округлением становиться больше, результат через какое-то количество итераций, не будет соответствовать реальности

# Решение

1. **Код программы**

#include <random>

#include <iostream>

#include <fstream>

#include <iomanip>

// Чтение матрицы из файла

void read\_matrix(double\*\* a, int n, int m, std::ifstream& inp)

{

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

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

inp >> a[i][j];

}

}

}

// Запись матрицы в файл

void write\_matrix(double\*\* matrix, int n, int m, std::ofstream& outp) {

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

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

outp << std::setw(15) << matrix[i][j] << ' ';

}

outp << std::endl;

}

}

double add\_form(double\*\* a, int n, double\* x, int i)

{

double result = 0;

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

if (j != i) result -= a[i][j] \* x[j];

}

return result;

}

// Подсчет ошибки

double get\_error(double\* x, double\* tmp\_x, int n)

{

double error = fabs(tmp\_x[0] - x[0]);

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

double tmp = fabs(tmp\_x[i] - x[i]);

if (error < tmp) error = tmp;

}

return error;

}

// Метод Якоби

bool jacobi(double\*\* a, int n, double\* x, int max\_k, double eps, int& k)

{

bool rashod = false;

double\* tmp\_x = new double[n];

double error = INFINITY;

k = 0;

while (error > eps && k < max\_k) {

k++;

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

tmp\_x[i] = (a[i][n] + add\_form(a, n, x, i)) / a[i][i];

}

error = get\_error(x, tmp\_x, n);

for (int i = 0; i < n; i++) x[i] = tmp\_x[i];

if (error > 1000000) {

rashod = true;

break;

}

}

delete[] tmp\_x;

if (k == max\_k || rashod)

return false;

return true;

}

// Метод Зейделя

bool zeidel(double\*\* a, int n, double\* x, int max\_k, double eps, int& k)

{

bool rashod = false;

double\* tmp\_x = new double[n];

double\* x\_copy = new double[n];

double error = INFINITY;

k = 0;

for (int i = 0; i < n; i++) x\_copy[i] = x[i];

while (error > eps && k < max\_k) {

k++;

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

tmp\_x[i] = (a[i][n] + add\_form(a, n, x\_copy, i)) / a[i][i];

x\_copy[i] = tmp\_x[i];

}

error = get\_error(x, tmp\_x, n);

for (int i = 0; i < n; i++) x[i] = tmp\_x[i];

if (error > 1000000) {

rashod = true;

break;

}

}

delete[] tmp\_x;

delete[] x\_copy;

if (k == max\_k || rashod)

return false;

return true;

}

// Метод Якоби с выводом промежуточных результатов и записью в файл

bool jacobi(double\*\* a, int n, double\* x, int max\_k, double eps, std::ostream& out)

{

double\* tmp\_x = new double[n];

double error = INFINITY;

int k = 0;

while (error > eps && k < max\_k) {

k++;

if (n < 20 && k < 50) {

std::cout << "Итерация = " << k << std::endl;

out << "Итерация = " << k << std::endl;

}

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

tmp\_x[i] = (a[i][n] + add\_form(a, n, x, i)) / a[i][i];

std::cout << "x" << i + 1 << " = " << std::setw(10) << tmp\_x[i] << " | ";

if (n < 20 && k < 50) {

out << "x" << i + 1 << " = " << std::setw(10) << tmp\_x[i] << " | ";

}

}

error = get\_error(x, tmp\_x, n);

if (n < 20 && k < 50) {

std::cout << "Ошибка = " << error;

out << "Ошибка = " << error;

std::cout << std::endl;

out << std::endl;

}

for (int i = 0; i < n; i++) x[i] = tmp\_x[i];

if (error > 1000000)

break;

}

std::cout << std::endl;

out << std::endl;

if (k == max\_k) {

std::cout << "Решение расходится. Количество итераций превысило максимальное значение: " << max\_k << std::endl;

out << "Решение расходится. Количество итераций превысило максимальное значение : " << max\_k << std::endl;

delete[] tmp\_x;

return false;

}

else if (error > 1000000) {

std::cout << "Решение расходится. Ошибка превысила значение 1000000" <<

std::endl;

out << "Решение расходится. Ошибка превысила значение 1000000" << std::endl;

delete[] tmp\_x;

return false;

}

else {

std::cout << "Решение сходится." << std::endl;

out << "Решение сходится." << std::endl;

std::cout << "Вектор X:" << std::endl;

out << "Вектор X:" << std::endl;

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

std::cout << "x" << i + 1 << " = " << x[i] << std::endl;

out << "x" << i + 1 << " = " << x[i] << std::endl;

}

}

std::cout << "Количество итераций = " << k << std::endl;

std::cout << "Точность = " << eps << std::endl;

out << "Количество итераций = " << k << std::endl;

out << "Точность = " << eps << std::endl;

delete[] tmp\_x;

return true;

}

// Метод Зейделя с выводом промежуточных результатов и записью в файл

bool zeidel(double\*\* a, int n, double\* x, int max\_k, double eps, std::ofstream& out)

{

double\* tmp\_x = new double[n];

double\* x\_copy = new double[n];

double error = INFINITY;

int k = 0;

for (int i = 0; i < n; i++) x\_copy[i] = x[i];

while (error > eps && k < max\_k) {

k++;

if (n < 20 && k < 50) {

std::cout << "Итерация = " << k << std::endl;

out << "Итерация = " << k << std::endl;

}

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

tmp\_x[i] = (a[i][n] + add\_form(a, n, x\_copy, i)) / a[i][i];

x\_copy[i] = tmp\_x[i];

if (n < 20 && k < 50) {

std::cout << "x" << i + 1 << " = " << std::setw(10) << tmp\_x[i] << "

| ";

out << "x" << i + 1 << " = " << std::setw(10) << tmp\_x[i] << " | ";

}

}

error = get\_error(x, tmp\_x, n);

if (n < 20 && k < 50) {

std::cout << "Ошибка = " << error;

out << "Ошибка = " << error;

std::cout << std::endl;

out << std::endl;

}

for (int i = 0; i < n; i++) x[i] = tmp\_x[i];

if (error > 1000000)

break;

}

std::cout << std::endl;

out << std::endl;

if (k == max\_k) {

std::cout << "Решение расходится. Количество итераций превысило максимальное значение: " << max\_k << std::endl;

out << "Решение расходится. Количество итераций превысило максимальное значение : " << max\_k << std::endl;

delete[] tmp\_x;

delete[] x\_copy;

return false;

}

else if (error > 1000000) {

std::cout << "Решение расходится. Ошибка превысила значение 1000000" <<

std::endl;

out << "Решение расходится. Ошибка превысила значение 1000000" << std::endl;

delete[] tmp\_x;

delete[] x\_copy;

return false;

}

else {

std::cout << "Решение сходится." << std::endl;

out << "Решение сходится." << std::endl;

std::cout << "Вектор X:" << std::endl;

out << "Вектор X:" << std::endl;

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

std::cout << "x" << i + 1 << " = " << x[i] << std::endl;

out << "x" << i + 1 << " = " << x[i] << std::endl;

}

}

std::cout << "Количество итераций = " << k << std::endl;

std::cout << "Точность = " << eps << std::endl;

out << "Количество итераций = " << k << std::endl;

out << "Точность = " << eps << std::endl;

delete[] tmp\_x;

delete[] x\_copy;

return true;

}

// Освобождение памяти

void free\_mem(double\*\* matrix, int n) {

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

delete[] matrix[i];

delete[] matrix;

}

// Рандомное заполнение матрицы

void fill\_matrix(double\*\* matrix, int n, int m)

{

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

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

matrix[i][j] = rand() % 200 - 100;

}

}

}

// Проверка, есть ли нули на главной диагонали

bool check\_zeros(double\*\* a, int n)

{

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

if (a[i][i] == 0) return true;

}

return false;

}

// Ввод входных значений n, eps, max\_k

void input\_values(int& n, double& eps, int& max\_k)

{

do {

std::cout << std::endl << "Введите размерность матрицы: ";

std::cin >> n;

} while (n < 2);

do {

std::cout << "Введите точность: ";

std::cin >> eps;

} while (eps < 0);

do {

std::cout << "Введите максимальное число итераций: ";

std::cin >> max\_k;

} while (max\_k < 1);

}

// Проверка диагонального преобладания

bool check\_diagonal\_dominance(double\*\* a, int n)

{

int sum = 0;

bool more = false;

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

sum = 0;

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

if (i != j)

sum += abs(a[i][j]);

}

if (abs(a[i][i]) > sum)

more = true;

if (sum > abs(a[i][i]))

return false;

}

return more;

}

// Перестановки строк (Попытка добиться диагонального преобладания)

void string\_permutation(double\*\* a, int n)

{

int max\_row;

double max;

double\* tmp\_row;

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

max\_row = i;

max = a[i][i];

// Перестановки строк

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

if (fabs(a[p][i]) > max) {

max\_row = p;

max = fabs(a[p][i]);

}

}

if (!(max\_row == i)) {

tmp\_row = a[max\_row];

a[max\_row] = a[i];

a[i] = tmp\_row;

}

}

}

// Вывод меню

void print\_menu()

{

std::cout << "Меню:" << std::endl;

std::cout << "1. Решение СЛАУ из файла методом Якоби" << std::endl;

std::cout << "2. Решение СЛАУ из файла методом Зейделя" << std::endl;

std::cout << "3. Решение автоматически сгенерированной СЛАУ методом Якоби" <<

std::endl;

std::cout << "4. Решение автоматически сгенерированной СЛАУ методом Зейделя" <<

std::endl;

std::cout << "5. Метод Якоби. Тест сходимости на 50000 случайных СЛАУ" <<

std::endl;

std::cout << "6. Метод Зейделя. Тест сходимости на 50000 случайных СЛАУ" <<

std::endl;

std::cout << "7. Выход из программы" << std::endl;

std::cout << std::endl;

std::cout << "Выберите пункт меню: ";

}

// Вывод подменю и ввод пункта подменю

bool submenu()

{

int choice = 0;

std::cout << std::endl << "Использовать перестановку строк, чтобы попытаться добиться диагонального преобладания ? " << std::endl;

std::cout << "1. Да" << std::endl;

std::cout << "2. Нет" << std::endl << std::endl;

do {

std::cout << "Ввод: ";

std::cin >> choice;

} while (choice < 1 || choice > 2);

if (choice == 1)

return true;

return false;

}

// Нахождение нормы невязки

double norma\_nevyazki(double\*\* a, double\* x, int n)

{

double tmp, nevyazka\_tmp, norma = 0;

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

tmp = 0;

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

tmp += a[i][j] \* x[j];

}

nevyazka\_tmp = a[i][n] - tmp;

if (nevyazka\_tmp > norma)

norma = nevyazka\_tmp;

}

return norma;

}

int main()

{

setlocale(LC\_ALL, "Russian");

int n, max\_k;

double eps;

int menu\_item;

while (true) {

srand(time(0));

system("cls");

print\_menu();

std::cin >> menu\_item;

switch (menu\_item)

{

case 1: // Метод Якоби с чтением матрицы из файла и записью результатов в файл

{

std::ifstream inp("Input.txt");

std::ofstream out("Result.txt");

if (inp.is\_open()) {

inp >> n;

inp >> eps;

inp >> max\_k;

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

double\* x = new double[n];

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

a[i] = new double[n + 1];

x[i] = 0;

}

read\_matrix(a, n, n + 1, inp);

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

out << "Исходная матрица:" << std::endl;

write\_matrix(a, n, n + 1, out);

print\_matrix(a, n, n + 1);

if (check\_diagonal\_dominance(a, n)) {

std::cout << "Матрица обладает диагональным преобладанием" <<

std::endl;

out << "Матрица обладает диагональным преобладанием" <<

std::endl;

}

std::cout << std::endl;

out << std::endl;

if (check\_zeros(a, n)) {

free\_mem(a, n);

delete[] x;

out << "Деление на ноль! Поменяйте местами уравнения.";

out.close();

inp.close();

std::cout << "Деление на ноль! Поменяйте местами уравнения.";

break;

}

if (jacobi(a, n, x, max\_k, eps, out)) {

std::cout << "Норма невязки = " << norma\_nevyazki(a, x, n) <<

std::endl;

out << "Норма невязки = " << norma\_nevyazki(a, x, n) <<

std::endl;

}

free\_mem(a, n);

delete[] x;

}

else std::cout << "Не удается открыть файл input.txt";

out.close();

inp.close();

system("pause");

break;

}

case 2: // Метод Зейделя с чтением матрицы из файла и записью результатов в файл

{

std::ifstream inp("Input.txt");

std::ofstream out("Result.txt");

if (inp.is\_open()) {

inp >> n;

inp >> eps;

inp >> max\_k;

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

double\* x = new double[n];

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

a[i] = new double[n + 1];

x[i] = 0;

}

read\_matrix(a, n, n + 1, inp);

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

out << "Исходная матрица:" << std::endl;

write\_matrix(a, n, n + 1, out);

print\_matrix(a, n, n + 1);

if (check\_diagonal\_dominance(a, n)) {

std::cout << "Матрица обладает диагональным преобладанием" <<

std::endl;

out << "Матрица обладает диагональным преобладанием" <<

std::endl;

}

std::cout << std::endl;

out << std::endl;

if (check\_zeros(a, n)) {

free\_mem(a, n);

delete[] x;

out << "Деление на ноль! Поменяйте местами уравнения.";

out.close();

inp.close();

std::cout << "Деление на ноль! Поменяйте местами уравнения.";

break;

}

if (zeidel(a, n, x, max\_k, eps, out)) {

std::cout << "Норма невязки = " << norma\_nevyazki(a, x, n) <<

std::endl;

out << "Норма невязки = " << norma\_nevyazki(a, x, n) <<

std::endl;

}

free\_mem(a, n);

delete[] x;

}

else std::cout << "Не удается открыть файл input.txt";

inp.close();

out.close();

system("pause");

break;

}

case 3: // Метод Якоби для случайной матрицы с записью результатов в файл

{

std::ofstream out("Result.txt");

input\_values(n, eps, max\_k);

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

double\* x = new double[n];

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

a[i] = new double[n + 1];

x[i] = 0;

}

fill\_matrix(a, n, n + 1);

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

out << "Исходная матрица:" << std::endl;

write\_matrix(a, n, n + 1, out);

print\_matrix(a, n, n + 1);

if (check\_diagonal\_dominance(a, n)) {

std::cout << "Матрица обладает диагональным преобладанием" <<

std::endl;

out << "Матрица обладает диагональным преобладанием" << std::endl;

}

std::cout << std::endl;

out << std::endl;

if (check\_zeros(a, n)) {

free\_mem(a, n);

delete[] x;

out << "Деление на ноль! Поменяйте местами уравнения.";

out.close();

std::cout << "Деление на ноль! Поменяйте местами уравнения.";

break;

}

if (jacobi(a, n, x, max\_k, eps, out)) {

std::cout << "Норма невязки = " << norma\_nevyazki(a, x, n) <<

std::endl;

out << "Норма невязки = " << norma\_nevyazki(a, x, n) << std::endl;

}

free\_mem(a, n);

delete[] x;

out.close();

system("pause");

break;

}

case 4: // Метод Зейделя для случайной матрицы с записью результатов в файл

{

std::ofstream out("Result.txt");

input\_values(n, eps, max\_k);

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

double\* x = new double[n];

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

a[i] = new double[n + 1];

x[i] = 0;

}

fill\_matrix(a, n, n + 1);

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

out << "Исходная матрица:" << std::endl;

write\_matrix(a, n, n + 1, out);

print\_matrix(a, n, n + 1);

if (check\_diagonal\_dominance(a, n)) {

std::cout << "Матрица обладает диагональным преобладанием" <<

std::endl;

out << "Матрица обладает диагональным преобладанием" << std::endl;

}

std::cout << std::endl;

out << std::endl;

if (check\_zeros(a, n)) {

free\_mem(a, n);

delete[] x;

out << "Деление на ноль! Поменяйте местами уравнения.";

out.close();

std::cout << "Деление на ноль! Поменяйте местами уравнения.";

break;

}

print\_matrix(a, n, n + 1);

if (zeidel(a, n, x, max\_k, eps, out)) {

std::cout << "Норма невязки = " << norma\_nevyazki(a, x, n) <<

std::endl;

out << "Норма невязки = " << norma\_nevyazki(a, x, n) << std::endl;

}

free\_mem(a, n);

delete[] x;

out.close();

system("pause");

break;

}

case 5: // Решение 50000 СЛАУ методом Якоби

{

bool permutations;

permutations = submenu();

input\_values(n, eps, max\_k);

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

double\* x = new double[n];

int k, k\_overall = 0;

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

a[i] = new double[n + 1];

x[i] = 0;

}

int count = 0, no\_divide\_by\_zero = 50000;

for (int i = 0; i < 50000; i++) {

fill\_matrix(a, n, n + 1);

if (permutations)

string\_permutation(a, n);

for (int i = 0; i < n; i++) x[i] = 0;

if (check\_zeros(a, n)) {

no\_divide\_by\_zero--;

continue;

}

if (jacobi(a, n, x, max\_k, eps, k)) {

count++;

k\_overall += k;

}

}

std::cout << "Процент сходимости = " << static\_cast<float>(count) /

no\_divide\_by\_zero \* 100 << "%" << std::endl;

std::cout << "Среднее число итераций = " << k\_overall / count <<

std::endl;

free\_mem(a, n);

delete[] x;

system("pause");

break;

}

case 6: // Решение 50000 СЛАУ методом Зейделя

{

bool permutations;

permutations = submenu();

input\_values(n, eps, max\_k);

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

double\* x = new double[n];

int k, k\_overall = 0;

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

a[i] = new double[n + 1];

x[i] = 0;

}

int count = 0, no\_divide\_by\_zero = 50000;

for (int i = 0; i < 50000; i++) {

fill\_matrix(a, n, n + 1);

if (permutations)

string\_permutation(a, n);

for (int i = 0; i < n; i++) x[i] = 0;

if (check\_zeros(a, n)) {

no\_divide\_by\_zero--;

continue;

}

if (zeidel(a, n, x, max\_k, eps, k)) {

count++;

k\_overall += k;

}

}

std::cout << "Процент сходимости = " << static\_cast<float>(count) /

no\_divide\_by\_zero \* 100 << "%" << std::endl;

std::cout << "Среднее число итераций = " << k\_overall / count <<

std::endl;

free\_mem(a, n);

delete[] x;

system("pause");

break;

}

case 7:

return 0;

default:

break;

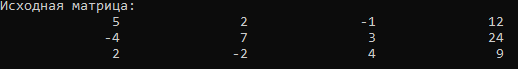
}

}

}

# Исследование зависимости числа итераций от начального приближения, точности, выбора метода решения.

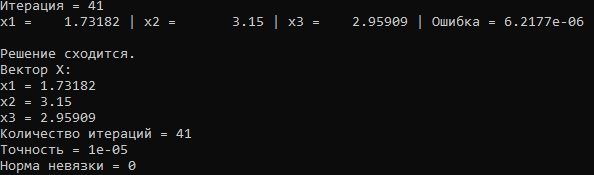
* 1. Зависимость числа итераций от начального приближения. Исходная расширенная матрица:



Начальное приближение 1:

X1 = 0; X2 = 0; X3 = 0

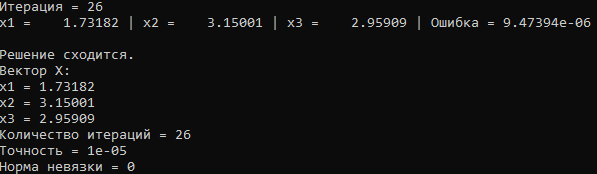
Решение:



Начальное приближение 2:

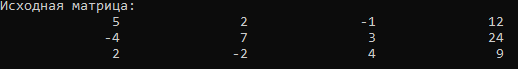
X1 = 1.7; X2 = 3.1; X3 = 2.9

Решение:

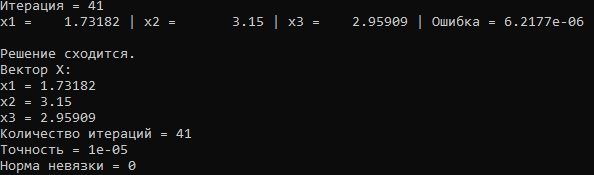


Вывод: Чем ближе значения начального приближения к корням СЛАУ, тем меньше итераций потребуется для решения этой СЛАУ.

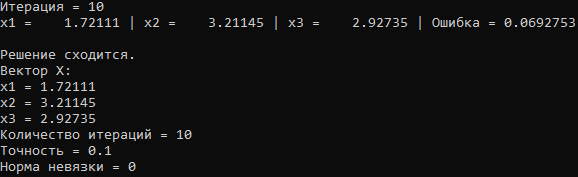
* 1. Зависимость числа итераций от точности. Исходная расширенная матрица:



Точность = 0.00001 Решение:

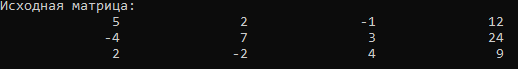


Точность = 0.1 Решение:

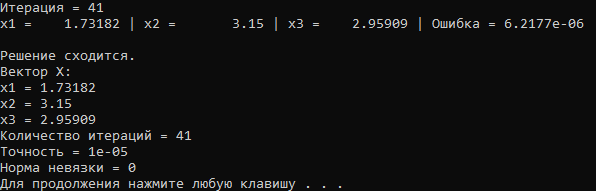


Вывод: Чем больше значение точности, тем меньше итераций потребуется для решения СЛАУ.

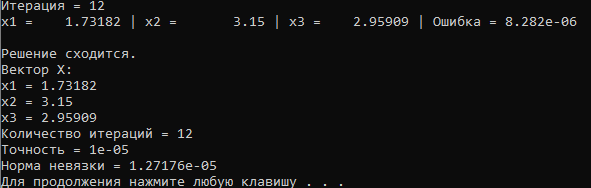
* 1. Зависимость числа итераций от выбора метода решения. Исходная расширенная матрица:



Метод Якоби:



Метод Зейделя:

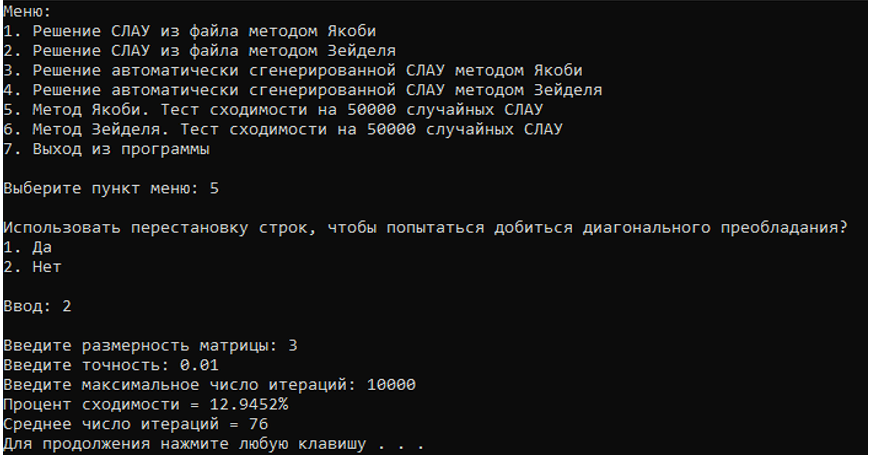


Вывод: Метод Зейделя сходится быстрее метода Якоби.

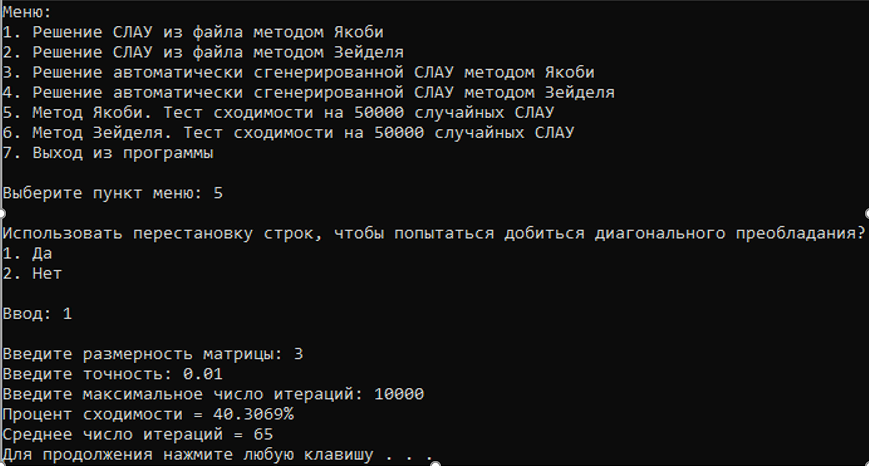
# Изучить влияние на сходимость величины диагонального преобладания матрицы, то есть величины отношения суммы модулей недиагональных элементов строки к модулю диагонального элемента.

Для изучения влияния диагонального преобладания на сходимость я проводил тестирование на 50000 различных случайных матрицах двумя способами. В первом способе я никак не изменял случайно заполненную матрицу. Во втором способе я пытался привести исходную матрицу к матрице с диагональным преобладанием. И сравнивал процент сходимости решений методами Якоби и Зейделя:

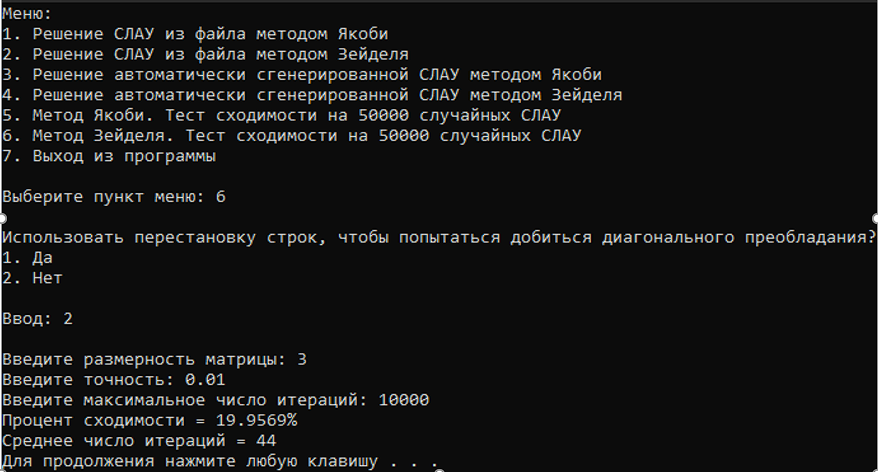
Метод Якоби без изменения матрицы:



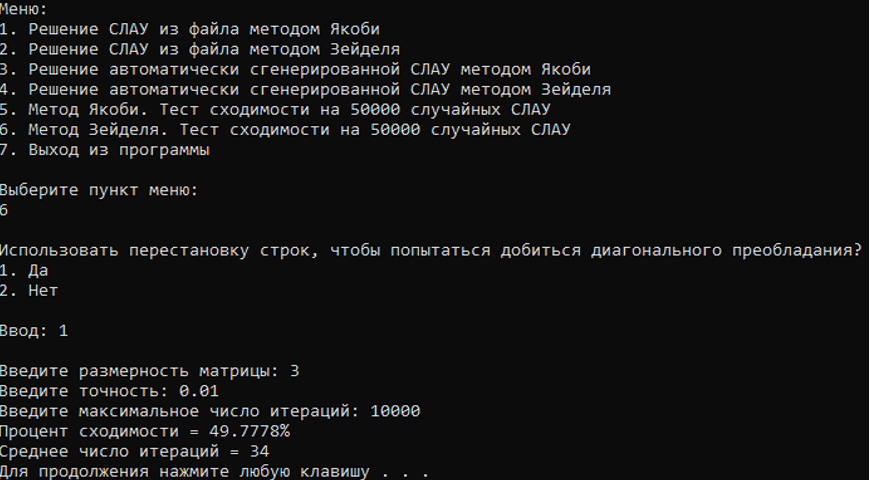
Метод Якоби с попыткой добиться диагонального преобладания:



Метод Зейделя без изменения матрицы:



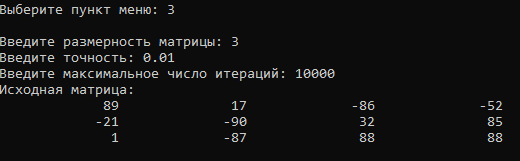
Метод Зейделя с попыткой добиться диагонального преобладания:

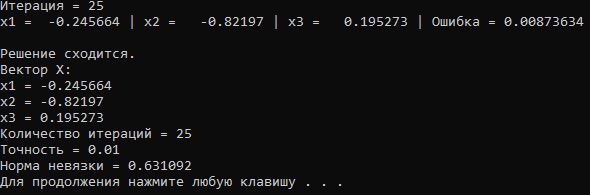


Выводы: Перестановки строк с попыткой вывести наибольшие элементы матрицы на главную диагональ действительно влияют на сходимость данных методов, но так как не существует универсального алгоритма получения диагонального преобладания в матрице, 100% сходимости данных методов добиться невозможно.

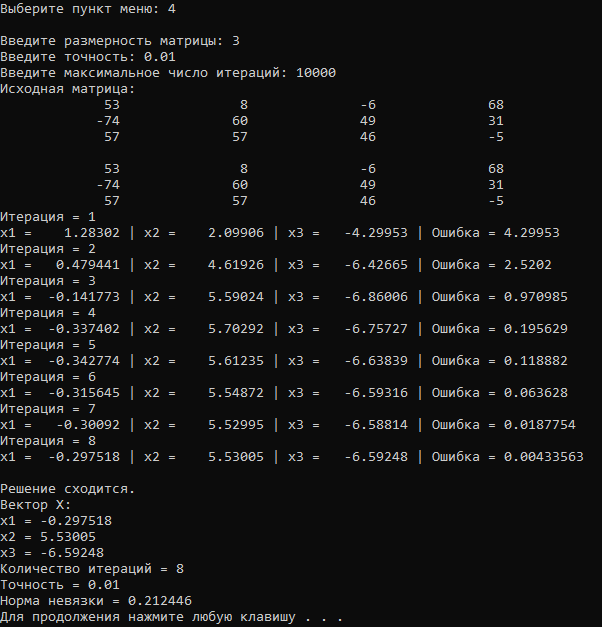
# Подобрать примеры, показывающие, что диагональное преобладание не является необходимым условием сходимости.

* 1. Пример 1 (Метод Якоби)





* 1. Пример 2 (Метод Зейделя)



Вывод: Диагональное преобладание не является необходимым условием сходимости.