**Белорусский государственный университет**

**Факультет прикладной математики и информатики**

Лабораторная работа №4

Решение СЛАУ методом верхней релаксации

Вариант №8

**Выполнил:**

Студент 2 курса 7 группы ПМ ФПМИ

Шевцов Евгений

**Преподаватель:**

Будник Анатолий Михайлович

**Минск – 2021**

**Описание метода верхней релаксации**

Метод релаксации является неявным одношаговым итерационным методом решения СЛАУ Ax = f. Идея метода исходит из задачи поиска минимума функции ошибок, что порождает задачу поиска минимума функционала:

Имея известное = (), при нахождении следующего будет изменить только i-ю компоненту. Тогда преобразование примет вид:

В координатной форме процесс примет вид:

Параметр берут в промежутке (0; 2), что обеспечивает схождение метода. В случае верхней релаксации параметр 1 < < 2. В программе = 1,12.

Для сходимости метода используется достаточное условие, а именно: матрица системы должна быть симметрической и положительно определённой. Для этого была применена трансформация Гаусса, т.е. матрица системы была домножена на себя транспонированную. В итоге система приняла вид:

ATAx = ATf

В качестве начального приближения был взять вектор неоднородности после вышеуказанного преобразования.

Итерации проходят до момента, пока не выполнится условие остановки: ||x(k+1) – x(k)|| < ε (по условию работы ε = 10-5). Норма использовалась кубическая.

**Листинг**

#include <iostream>

#include <iomanip>

#include <vector>

#include <cmath>

std::vector<std::vector<double>> transpositionMatrix(std::vector<std::vector<double>> M) {

for (int i = 0; i < 4; ++i) {

for (int j = i + 1; j < 5; ++j) {

std::swap(M[i][j], M[j][i]);

}

}

return M;

}

std::vector<std::vector<double>> multMatrix(const std::vector<std::vector<double>> M1, const std::vector<std::vector<double>> M2) {

std::vector<std::vector<double>> M3 = {

{0.0, 0.0, 0.0, 0.0, 0.0},

{0.0, 0.0, 0.0, 0.0, 0.0},

{0.0, 0.0, 0.0, 0.0, 0.0},

{0.0, 0.0, 0.0, 0.0, 0.0},

{0.0, 0.0, 0.0, 0.0, 0.0} };

for (int i = 0; i < 5; ++i) {

for (int j = 0; j < 5; ++j) {

for (int k = 0; k < 5; ++k) {

M3[i][j] += M1[i][k] \* M2[k][j];

}

}

}

return M3;

}

std::vector<double> multVec(const std::vector<std::vector<double>> M, const std::vector<double> f) {

std::vector<double> b = { 0., 0., 0., 0., 0. };

for (int i = 0; i < 5; ++i) {

for (int j = 0; j < 5; ++j) {

b[i] += M[i][j] \* f[j];

}

}

return b;

}

double errorEstimation(std::vector<double> x2, std::vector<double> x1) {

double cubeNorm = 0.;

for (int i = 0; i < 5; ++i) {

double res = abs(x2[i] - x1[i]);

if (res > cubeNorm) {

cubeNorm = res;

}

}

return cubeNorm;

}

int main()

{

const double EPS = 10E-5;

const double w = 1.12;

int count = 0;

std::vector<std::vector<double>> A =

{ {0.7941, 0.0000, -0.2067, 0.1454, 0.2423},

{-0.0485, 0.5168, 0.0000, -0.0985, 0.0323},

{0.0162, -0.1454, 0.9367, 0.0178, 0.0565},

{0.0485, 0.0000, -0.1179, 0.9367, 0.0000},

{0.0323, -0.0485, 0.2342, -0.0194, 0.6783} };

std::vector<double> f = { 1.5569, 2.0656, -2.9054, -8.0282, 3.4819 };

//Для получения транспонированной положительно определённой матрицы системы

std::vector<std::vector<double>> A\_T = multMatrix(transpositionMatrix(A), A);

std::vector<double> f\_T = multVec(transpositionMatrix(A), f);

std::vector<double> x0 = f\_T;

std::vector<double> xk = { 0, 0, 0, 0, 0 };

while (errorEstimation(xk, x0) > EPS) {

if (count != 0) {

x0 = xk;

}

++count;

for (int i = 0; i < 5; ++i) {

xk[i] = 0;

xk[i] += (1 - w) \* x0[i] + w \* f\_T[i] / A\_T[i][i];

double sum = 0.;

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

sum += A\_T[i][j] \* xk[j];

}

for (int j = i + 1; j < 5; ++j) {

sum += A\_T[i][j] \* x0[j];

}

xk[i] -= w \* sum / A\_T[i][i];

}

}

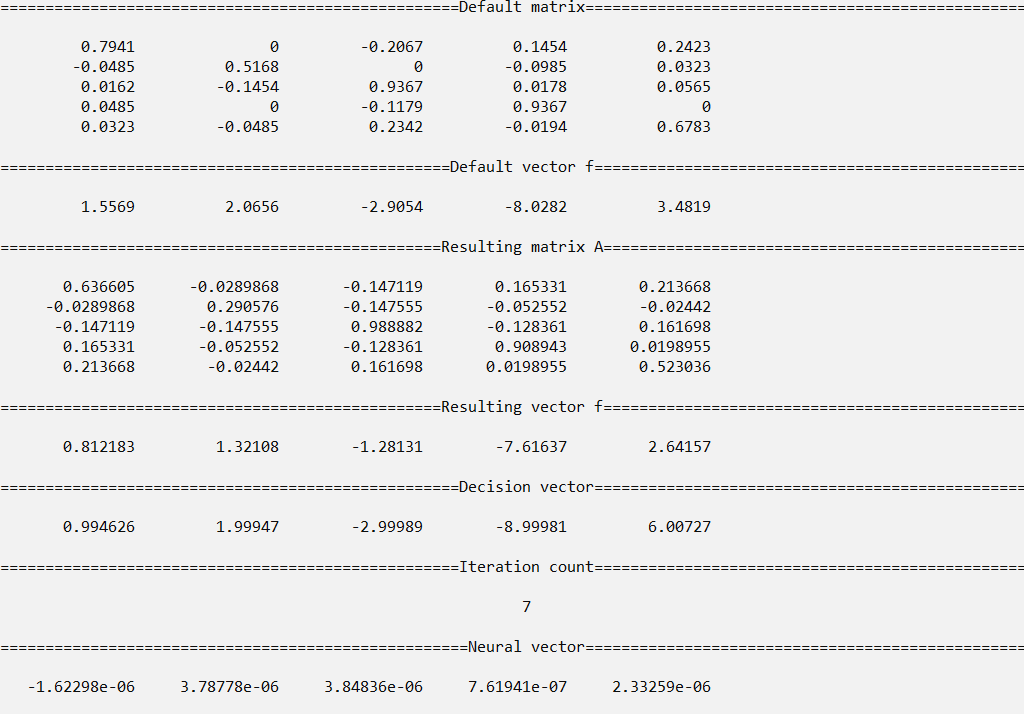
std::vector<double> neuralVector = multVec(A, xk);

for (int i = 0; i < 5; ++i) {

neuralVector[i] -= f[i];

}

**Выходные данные**



**Вывод**

В ходе итерационного процесса было произведено 7 итераций, что, в сравнении с методом Якоби на 1 меньше для аналогичной системы. Норма полученной невязки равна 1.62298\*10-6, что на порядок меньше в методе Якоби. Однако результат очень зависит от выбора , т.к. при изменении параметра на 0.1 количество итераций увеличится (проверено программой).

Следовательно, можно сделать вывод, что при оптимальном выборе параметра , метод верхней релаксации будет сходится с немного бОльшей скоростью и точностью, чем метод Якоби.