**Лабораторная работа**

**«Итерационные методы решения СЛАУ»**

**Арсений Анищенко**

**Группа 2**

1. **Постановка задачи.**

Пусть дана система линейных алгебраических уравнений вида

*a*11*x*1 + *a*12*x*2 +  … + *a*1n*xn* = *f*1,

*a*21*x*2 + *a*22*x*2 +  … + *a*2n*xn* = *f*2,

. . . . . . . . . . . . . . . . . . . . . . . .

*an*1*x*1 + *an*2*x*2 +  … + *annxn* = *fn* .

Для всех заданий лабораторной работы:

Задать матрицу системы:

* недиагональные элементы *ai,j*, *i≠j*, выбираются из чисел 0, –1, –2, –3, *–*4 произвольным образом;
* *ai,i=*, 2≤*i*≤*n*;
* *a*11*=*+1;

матрица системы имеет диагональное преобладание, для первого уравнения преобладание строгое.

Матрица генерируется один раз, для всех заданий она одна и та же.

Задать правую часть *f* умножением матрицы *A* на вектор *x=*(*m*, *m*+1, ... , *n*+*m*–1): *f=Ax*.

Для вычислений выбрать параметры:

* *m* – номер в списке студенческой группы;
* *n* – одно из чисел в пределах от 10 до 12.

В качестве языка программирования выбрать C или C++, для вычислений использовать тип float.

Выход из итерационного процесса выполнять, если <ε, либо если *k*>*kmax*. Задать ε*=*0,0001, *kmax=*1000.

Вывести на печать полученный приближенный вектор решений и номер итерации, при которой достигнута требуемая точность. Предусмотреть сообщение о выходе из итерационного процесса из-за превышения допустимого максимального количества итераций; в этом случае вывести на печать приближенный вектор решений, полученный на итерации *kmax*.

**Задание 1.** Разработать программу численного решения СЛАУ методом Якоби:

(*i*)*=**,*

*i=* 1*,* 2, …, *n*, *k =* 0, 1, 2, …

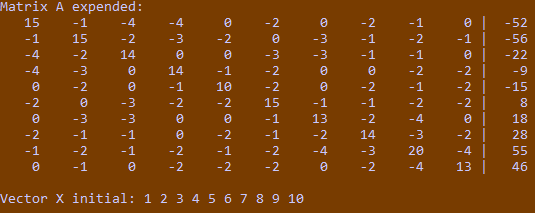
**Задание 2.** Разработать программу численного решения СЛАУ методом релаксации:

(*i*)*=* (*i*) +*,*

*i=* 1*,* 2, …, *n*, *k =* 0, 1, 2, …

Рассмотреть три случая: ω*=*0,5, ω*=*1 (это метод Зейделя), ω*=*1,5.

1. **Входные данные.**



1. **Листинг программы.**

#include <vector>

#include <fstream>

#include <iostream>

#include <algorithm>

#include <math.h>

#include <time.h>

#include <iomanip>

using namespace std;

const int SIZE = 10;

const float EPS = 0.0001;

const float KMAX = 1000;

float \*\*a, s[SIZE], x[SIZE], \*b;

int m = 1;

int n = SIZE;

float delta(float \*prev, float \*x){

float res = fabs(x[0] - prev[0]);

for (int i = 1; i < n; ++i){

res = max(res, (float)fabs(x[0] - prev[0]));

}

return res;

}

// Задание 1 : Метод Якоби

void jacoby(float \*\*a, float \* f){

int iter = 0;

float \*x = new float[SIZE];

float \*prev = new float[SIZE];

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

x[i] = f[i] / a[i][i];

}

// Выполнять, пока количество итераций не превысит заданное Kmax

while (iter < KMAX){

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

prev[i] = x[i];

x[i] = 0;

}

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

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

if (i != j){

x[i] -= a[i][j] \* prev[j];

}

}

x[i] += f[i];

x[i] /= a[i][i];

}

// Проверка на досрочный выход из цикла

if (delta(prev, x) < EPS){

break;

}

// Переходим к следующей итерации

++iter;

}

// Вывод итоговой информации

cout << endl << "The Jacobi Method:" << endl;

cout << "Iterations count: " << iter + 1 << endl;

cout << "Precision: " << delta(prev, x) << endl;

cout << "Vector x: ";

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

cout << fixed << setprecision(4) << x[i] << " ";

}

cout << endl;

}

// Задание 2 : Метод Релаксации

void relaxation(float \*\*a, float \* f, float w){

int iter = 0;

float \*x = new float[SIZE];

float \*prev = new float[SIZE];

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

x[i] = f[i] / a[i][i];

}

// Выполнять, пока количество итераций не превысит заданное Kmax

while (iter < KMAX){

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

prev[i] = x[i];

x[i] = 0;

}

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

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

x[i] -= a[i][j] \* x[j];

}

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

x[i] -= a[i][j] \* prev[j];

}

x[i] += f[i];

x[i] \*= w;

x[i] /= a[i][i];

x[i] += (1 - w) \* prev[i];

}

// Проверка на досрочный выход из цикла

if (delta(prev, x) < EPS){

break;

}

// Переходим к следующей итерации

iter++;

}

// Вывод итоговой информации

cout << endl << "Relaxation method:" << endl;

cout << "Iterations count: " << iter + 1 << endl;

cout << "w = " << w << endl;

cout << "Precision: " << delta(prev, x) << endl;

cout << "Vector x: ";

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

cout << fixed << setprecision(4) << x[i] << " ";

}

cout << endl;

}

int main()

{

srand(time(0));

a = new float\*[SIZE];

b = new float[SIZE];

const float w1 = 0.5;

const float w2 = 1;

const float w3 = 1.5;

// Генерация матрицы A

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

a[i] = new float[SIZE];

}

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

for (int j = i + 1; j < n; ++j)

// Если элемент не на главной диагонали,

// то генерируем числа от -4 до 0

if (i != j){

a[i][j] = rand() % 5 - 4;

a[j][i] = a[i][j];

}

}

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

s[i] = 0;

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

if (i != j){

s[i] += a[i][j];

}

a[i][i] = -s[i];

}

}

a[0][0] += 1;

// Вычисления x и b

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

b[i] = 0;

x[i] = m + i;

}

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

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

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

}

}

cout << "Matrix A expended:" << endl;

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

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

cout << setw(5) << setprecision(5) << a[i][j] << " ";

}

cout << "|" << setw(5) << b[i] << endl;

}

// Вывод вектора X

cout << endl << "Vector X initial: ";

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

cout << x[i] << " ";

}

cout << endl;

jacoby(a, b);

// Вызов метода релаксации для трех различных w

relaxation(a, b, w1);

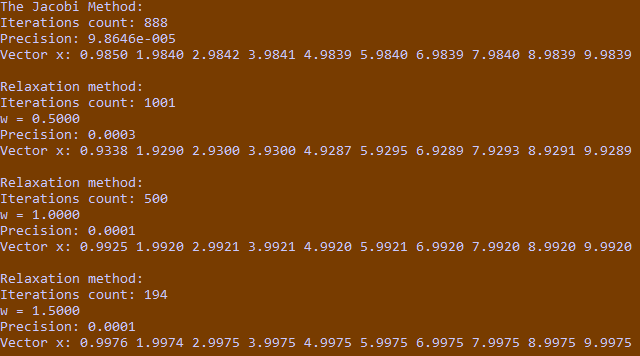
relaxation(a, b, w2);

relaxation(a, b, w3);

return 0;

}

1. **Выходные данные.**



1. **Вывод.**

Была реализована программа численного решения СЛАУ итерационными методами, такими как Якоби и релаксации. При решении СЛАУ методом Якоби мы получили погрешность примерно 0.0001. В методе релаксации мы рассмотрели три случая с w равным 0.5, 1 и 1.5, при этом наблюдалось повышение точности вычислений.