**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

Лабораторная №3

**Итерационные методы решения СЛАУ**

**Выполнил:**

Крученков Евгений Андреевич

студент 2 курса 9 группы,

специальность

“прикладная математика”

**Преподаватель:**

Ассистент кафедры вычислительной

математики ФПМИ,

Ю.Н. Горбачёва

Минск, 2021

**Содержание:**

Постановка задачи ------------------------------------------------------------------ 2

Краткие теоретические сведения ------------------------------------------------ 3

Листинг программы ---------------------------------------------------------------- 3-11

Результаты --------------------------------------------------------------------------- 11-12

Выводы ------------------------------------------------------------------------------- 12

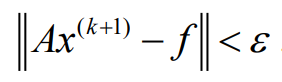
**Постановка задачи**

Написать и отладить программу численного решения систем линейных алгебраических уравнений *Ax = f* c квадратной матрицей порядка n

1)методом минимальных невязок;

2) методом релаксации.

Предусмотреть сообщение о выходе из итерационного процесса из-за превышения допустимого максимального количества итераций. В качестве критерия остановки итерационного процесса использовать:

****

1. Решить СЛАУ с матрицей A порядка n =10 и правой частью f методом градиентного спуска (если порядковый номер в списке подгруппы четный), методом минимальных невязок (если порядковый номер в списке подгруппы нечетный). В результатах необходимо привести следующую информацию:

матрица A (построчно);

точное решение x;

правая часть f;

начальное приближение (0) x ;

номер итерации q, при которой достигнута требуемая точность; o полученное приближенное решение;

абсолютная погрешность.

2) Для СЛАУ с матрицей A порядка n =10 и правой частью f исследовать сходимость метода релаксации в зависимости от параметра релаксации ω∈{0.2, 0.5, 0.8,1,1.3,1.5,1.8} . Результаты вычислительного эксперимента оформить в виде таблицы 1 (см. ниже).

**Краткие теоретические сведения**

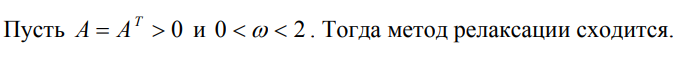
***Метод минимальных невязок***

*k = 1,2,… ; r – невязка*

***Метод релаксации***

*i = 1,2,…,n; k = 1,2,…; w – параметр релаксации*

Теорема (о сходимости метода релаксации).



**Листинг программы**

**Header.h**

#pragma once

#include <iostream>

#include <cstdio>

#include <iomanip>

#include <fstream>

#include <string>

#include <ctime>

#include <cmath>

using namespace std;

class Matr

{

friend class Sist;

double\*\* M;

int n;

public:

Matr()

{

this->n = 1;

M = new double\* [n];

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

{

M[i] = new double[n];

}

srand(time(0));

M[0][0] = rand()%200-100;

}

Matr(int n)

{

this->n = n;

M = new double\* [n];

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

{

M[i] = new double[n];

}

srand(time(0));

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

{

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

{

if(i!=j)

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

else

M[i][j] = 0;

}

}

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

{

int summ = 0;

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

{

summ += abs(M[i][j]);

}

M[i][i] = summ+5+rand()%45;

}

}

Matr(const Matr& B)

{

this->n = B.n;

M = new double\* [n];

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

{

M[i] = new double[n];

}

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

{

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

{

M[i][j] = B.M[i][j];

}

}

}

~Matr()

{

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

{

delete[] M[i];

}

delete[] M;

}

friend ostream& operator <<(ostream& os, Matr& A)

{

os << fixed << showpoint << setprecision(3);

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

{

for (int j = 0; j < A.n; j++)

{

os << A.M[i][j] << " ";

}

os << endl;

}

return os;

}

Matr& operator = (const Matr& B)

{

if (this == &B)

return \*this;

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

{

delete[] M[i];

}

n = B.n;

delete[] M;

M = new double\* [n];

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

{

M[i] = new double[n];

}

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

{

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

{

M[i][j] = B.M[i][j];

}

}

return \*this;

}

Matr operator \* (const Matr& B)

{

if (n != B.n) throw "Матрицы имеют разные размеры";

Matr P(n);

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

{

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

{

P.M[i][j] = 0;

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

{

P.M[i][j] += M[i][k] \* B.M[k][j];

}

}

}

return P;

}

};

class Sist

{

friend class Matr;

private:

Matr A;

double\* X;

double\* F;

double\* X\_;

const double E = 1\*pow(10,-7);

const int k\_max = 5000;

double norm = 0;

int q = 0;

double pogr;

public:

int size()

{

return A.n;

}

void ShowPogr()

{

cout << scientific << pogr << fixed << showpoint << setprecision(3);

}

void ShowNorm()

{

cout << scientific << norm << fixed << showpoint << setprecision(3);

}

void ShowQ()

{

cout << dec << q << fixed << showpoint << setprecision(3);

}

Sist(const Matr& B)

{

A = B;

X = new double[B.n];

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

{

X[i] = 1 + i;

}

F = new double[B.n];

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

{

F[i] = 0;

for (int j = 0; j < size(); j++)

{

F[i] += A.M[i][j] \* X[j];

}

}

X\_ = new double[B.n];

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

{

X\_[i] = F[i];

}

}

Sist(const Sist& B)

{

A = B.A;

X = new double[size()];

F = new double[size()];

X\_ = new double[size()];

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

{

for (int j = 0; j < size(); j++)

{

A.M[i][j] = B.A.M[i][j];

}

}

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

{

X[i] = B.X[i];

}

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

{

F[i] = B.F[i];

}

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

{

X\_[i] = B.X\_[i];

}

q = B.q;

}

~Sist()

{

delete[] X;

delete[] X\_;

delete[] F;

}

friend ostream& operator <<(ostream& os, Sist& B)

{

os << "Матрица системы:" << endl << fixed << showpoint << setprecision(3);

os << B.A << endl;

os << "Столбец свободных членов:" << endl;

for (int i = 0; i < B.size(); i++)

{

os << B.F[i] << " ";

os << endl;

}

os << "Точное решение X:" << endl;

for (int i = 0; i < B.size(); i++)

{

os << B.X[i] << " ";

os << endl;

}

os << "Начальное приближение:" << endl;

for (int i = 0; i < B.size(); i++)

{

os << B.F[i] << " ";

os << endl;

}

return os;

}

void Show()

{

cout << "Номер итерации q, при которой достигнута требуемая точность:" << q << endl;

cout << "Приближенное решение X\_:" << endl;

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

{

cout << fixed << showpoint << setprecision(20) << X\_[i] << " ";

cout << endl;

}

cout<< "Норма AX-F: " << scientific << norm << endl << fixed << showpoint << setprecision(3) ;

cout << "Абсолютная погрешность: " << scientific << pogr <<endl<< fixed << showpoint << setprecision(3);

}

Sist& operator = (const Sist& B)

{

if (this == &B)

return \*this;

A = B.A;

delete[] X;

delete[] X\_;

delete[] F;

X = new double[A.n];

F = new double[A.n];

X\_ = new double[A.n];

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

{

X[i] = B.X[i];

}

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

{

F[i] = B.F[i];

}

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

{

X\_[i] = B.X\_[i];

}

return \*this;

}

void MMN()

{

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

{

X\_[i] = F[i];

}

double\* V = new double[size()];

double\* Ax = new double[size()];

double\* Av = new double[size()];

double T;

norm = E;

for (q = 0; q < k\_max; q++)

{

//1

norm = 0;

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

{

Ax[i] = 0;

for (int j = 0; j < size(); j++)

{

Ax[i] += A.M[i][j] \* X\_[j];

}

V[i] = Ax[i] - F[i];

norm += abs(V[i]);

}

if (norm < E) break;

//2

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

{

Av[i] = 0;

for (int j = 0; j < size(); j++)

{

Av[i] += A.M[i][j] \* V[j];

}

}

double Av\_v = 0, Av\_Av = 0;

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

{

Av\_v += Av[i] \* V[i];

Av\_Av += Av[i] \* Av[i];

}

T = Av\_v / Av\_Av;

//3

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

{

X\_[i] = X\_[i] - T \* V[i];

}

}

if (q == k\_max)

cout << "Итерационный процесс остановлен при достижении максимального числа итераций" << endl;

delete[] Av;

delete[] Ax;

delete[] V;

pogr = 0;

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

{

pogr += abs(X[i] - X\_[i]);

}

this->Show();

}

void Relax(double w)

{

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

{

X\_[i] = F[i];

}

if (w < 0 || w > 2)

cout << "w должно принадлежать промежутку (0;2)" << endl;

double\* Ax = new double[size()];

for (q = 0; q < k\_max; q++)

{

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

{

double summ1 = 0, summ2 = 0;

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

{

summ1 += A.M[i][j] \* X\_[j];

}

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

{

summ2 += A.M[i][j] \* X\_[j];

}

X\_[i] = (1 - w) \* X\_[i] + (w / A.M[i][i]) \* (F[i] - (summ1 + summ2));

}

norm = 0;

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

{

Ax[i] = 0;

for (int j = 0; j < size(); j++)

{

Ax[i] += A.M[i][j] \* X\_[j];

}

norm += abs(Ax[i] - F[i]);

}

if (norm < E) break;

}

if (q == k\_max)

cout << "Итерационный процесс остановлен при достижении максимального числа итераций" << endl;

pogr = 0;

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

{

pogr += abs(X[i] - X\_[i]);

}

delete[] Ax;

}

};

**Main.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include "Header.h"

#include <fstream>

#include<time.h>

using namespace std;

ifstream fin;

ofstream fout;

int main()

{

srand(time(0));

setlocale(LC\_ALL, "ru");

Matr M(10);

Sist A1(M), A2(M);

cout << A1 << endl;

A1.MMN();

cout << endl;

cout <<" W\t\t Количество итераций q\t\t Норма\t\t\t Абсолютная погрешность"<<endl;

for (double w = 0.2; w <= 1.8; w += 0.2)

{

A2.Relax(w);

cout << w << "\t\t\t";

A2.ShowQ();

cout << "\t\t\t";

A2.ShowNorm();

cout << "\t\t\t";

A2.ShowPogr();

cout << endl;

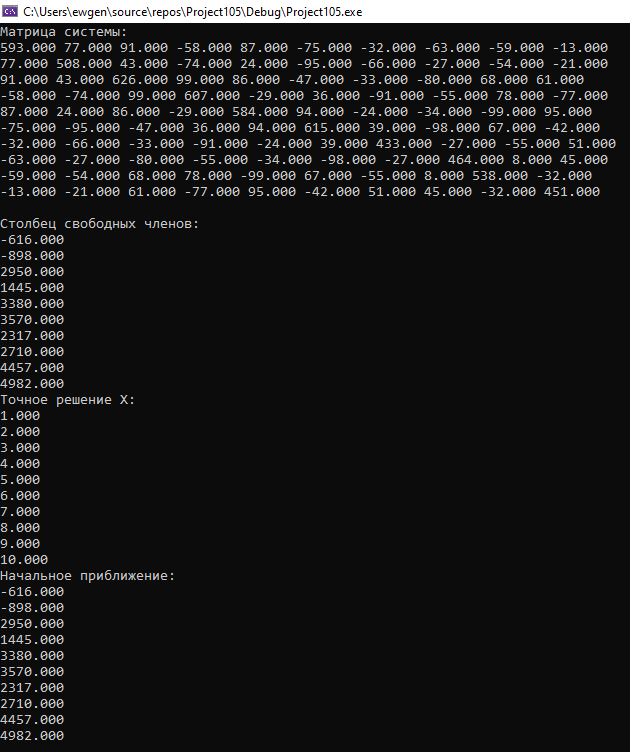
}

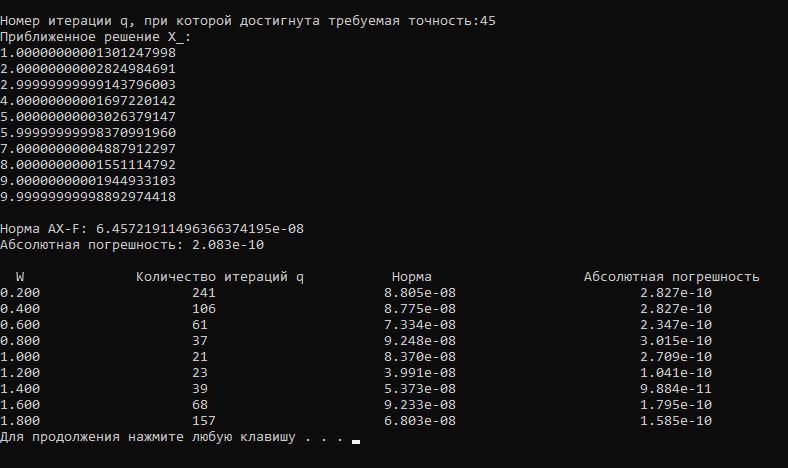
system("pause");

return 0;

}

**Результаты**

****

****

**Выводы**

1. Абсолютная погрешность при использовании итерационных методов будет зависеть от количества итераций и, следовательно, в нашем случае от наперёд заданного числа ε.
2. Для схождения используемых методов достаточно, чтобы матрица была симметрическая и положительно определённой, что выполняется исходя из условия
3. В данном случае оптимальным оказался коэффициент релаксации, равный 1, при котором метод релаксации совпадает с методом Гаусса-Зейделя.