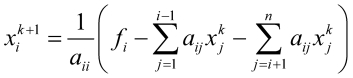
**Лабораторная работа 4 «Итерационные методы решения СЛАУ»**

Выполнил студент 3 курса 4 группы ФПМИ БГУ Видевич Александр

**Задача.**

Разработать программу численного решения СЛАУ методом Якоби (*n* – порядок матрицы системы *Ax=f*):

*, i=* 1*,* 2, …, *n*, *k =* 0, 1, 2, …

Разработать программу численного решения СЛАУ методом релаксации:



*, i=* 1*,* 2, …, *n;k =* 0, 1,2, …

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

Для вычислений выбрать параметры:

* Для вычислений выбрать параметры:
* *n* – одно из чисел в пределах от 10 до 12;
* *m* – номер в списке студенческой группы.
* K - максимальное число итераций = 1000
* ε - ограничение разности векторов следующей итерации = 0,0001

Для оценки изменения вектора на итерации использовалась метрика:



**Входные данные.**

n = 11, m = 4

**Листинг программы.**

*#include* <iostream>

*#include* <vector>

*#include* <iomanip>

*#include* <numeric>

*#include* <limits>

int n, K = *1000*;

*const* float eps = *0.0001*;

*// фунция генерации матрицы*

std::*vector*<std::*vector*<float>> GenerateMatrix()

{

    std::*vector*<std::*vector*<float>> matrix(n, std::*vector*<float>(n));

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

    {

        float sum = *0*;

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

        {

*if* (i != j)

            {

                matrix[i][j] = static\_cast<float>(std::rand() % *5* - *4*);

                sum += matrix[i][j];

            }

        }

*if* (i == *0*)

        {

            matrix[i][i] = -sum + *1*;

        }

*else*

        {

            matrix[i][i] = -sum;

        }

    }

*return* matrix;

}

*#define* M *4*

*// фунция создания вектора x*

std::*vector*<float> GenerateVector(*const* int size)

{

    std::*vector*<float> vector(size);

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

    {

        vector[i] = M + i;

    }

*return* vector;

}

*// функция умножения матрицы на вектор*

template <typename T>

std::*vector*<float> MatrixVectorMultiply(*const* std::*vector*<std::*vector*<float>> *&*matrix, *const* std::*vector*<T> *&*vector)

{

*const* int size = vector.size();

    std::*vector*<float> result(size);

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

    {

        float sum = *0*;

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

        {

            sum += matrix[i][j] \* static\_cast<float>(vector[j]);

        }

        result[i] = sum;

    }

*return* result;

}

*// разность векторов*

std::*vector*<float> CalculateDifference(*const* std::*vector*<float> *&*first, *const* std::*vector*<float> *&*second)

{

    std::*vector*<float> diff(first.size());

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

    {

        diff[i] = first[i] - second[i];

    }

*return* diff;

}

*// максимум норма*

float GetMaxCoordinate(*const* std::*vector*<float> *&*vector)

{

    float element = std::*numeric\_limits*<float>::min();

*for* (auto el : vector)

    {

        element = std::max(element, std::abs(el));

    }

*return* element;

}

*// функция решения СЛАУ методом Якоби*

std::*vector*<float> SolveUsingJacobyMethod(*const* std::*vector*<std::*vector*<float>> *&*matrix, *const* std::*vector*<float> *&*f)

{

    std::*vector*<float> x(n, *0*);

    std::*vector*<float> temp(n);

    int k = *1*;

*do*

    {

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

        {

            float sum = *0*;

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

            {

*if* (i == j)

*continue*;

                sum -= matrix[i][j] \* x[j];

            }

            temp[i] = *1* / matrix[i][i] \* (f[i] + sum);

        }

*if* (GetMaxCoordinate(CalculateDifference(x, temp)) < eps)

        {

            std::cout << "Not reached max iter in Jacoby" << '*\n*';

            std::cout << "Number of iteration " << k << '*\n*';

            x = temp;

*break*;

        }

        x = temp;

    } *while* (++k <= K);

*return* x;

}

*// метод Ралаксации*

std::*vector*<float> SolveUsingRelaxationMethod(*const* std::*vector*<std::*vector*<float>> *&*matrix, *const* std::*vector*<float> *&*f, float om)

{

    std::*vector*<float> x(n, *0*);

    std::*vector*<float> temp(n, *0*);

    int k = *1*;

*do*

    {

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

        {

            float sum = *0*;

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

            {

                sum += matrix[i][j] \* temp[j];

            }

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

            {

                sum += matrix[i][j] \* x[j];

            }

            temp[i] = (*1* - om) \* x[i] + (om / matrix[i][i]) \* (f[i] - sum);

        }

*if* (GetMaxCoordinate(CalculateDifference(x, temp)) < eps)

        {

            std::cout << "Not reached max iter in SolveUsingRelaxationMethod "

                      << "om: " << om << std::endl;

            std::cout << "Number of iteration " << k << std::endl;

            x = temp;

*break*;

        }

        x = temp;

    } *while* (++k <= K);

*return* x;

}

*// функция вывода вектора*

template <typename T>

void PrintVector(*const* std::*vector*<T> *&*vector, *const* int numberOfElements = *0*, *const* std::string *&*message = "")

{

*if* (message != "")

    {

        std::cout << message << ' ';

    }

*const* int border = numberOfElements != *0* ? numberOfElements : vector.size();

*for* (int i = *0*; i < border; ++i)

    {

        std::cout << vector[i] << ' ';

    }

    std::cout << '*\n*';

}

int main()

{

    std::ios\_base::sync\_with\_stdio(*false*);

    std::cin.tie(*nullptr*);

    std::srand(std::time(*NULL*));

*// ввод данных*

    std::cin >> n;

*// генерация задачи*

    std::*vector*<std::*vector*<float>> matrix = GenerateMatrix();

*const* std::*vector*<float> vector = GenerateVector(n);

*const* std::*vector*<float> b = MatrixVectorMultiply(matrix, vector);

*// расчеты*

*const* std::*vector*<float> jacoby = SolveUsingJacobyMethod(matrix, b);

*const* std::*vector*<float> relax0\_5 = SolveUsingRelaxationMethod(matrix, b, *0.5*);

*const* std::*vector*<float> zeidel = SolveUsingRelaxationMethod(matrix, b, *1*);

*const* std::*vector*<float> relax1\_5 = SolveUsingRelaxationMethod(matrix, b, *1.5*);

*// вывод ответов*

    PrintVector(vector);

    PrintVector(jacoby, n, "Jacoby:");

    PrintVector(relax0\_5, n, "Relaxation om = 0.5:");

    PrintVector(zeidel, n, "Relaxation om = 1 (Zeidel):");

    PrintVector(relax1\_5, n, "Relaxation om = 1.5:");

*return* *0*;

}

**Выходные данные.**

Not reached max iter in SolveUsingRelaxationMethod om: 1

Number of iteration 809

Not reached max iter in SolveUsingRelaxationMethod om: 1.5

Number of iteration 305

4 5 6 7 8 9 10 11 12 13 14

Jacoby: 3.855 4.84846 5.84778 6.84895 7.84852 8.84833 9.8484 10.8483 11.8484 12.8483 13.8486

Relaxation om = 0.5: 3.39565 4.36827 5.36555 6.37076 7.36902 8.36852 9.3689 10.3685 11.3692 12.3692 13.3707

Relaxation om = 1 (Zeidel): 3.98917 4.98868 5.98863 6.98874 7.98871 8.98871 9.98872 10.9887 11.9887 12.9888 13.9888

Relaxation om = 1.5: 3.9967 4.99654 5.99653 6.99658 7.99657 8.99658 9.99659 10.9966 11.9966 12.9966 13.9966

**Выводы.**

Итерационные методы решения СЛАУ позволяют получить приближенное решение с некоторой точностью. Для метода релаксации с омега равным 1(он же метод Зейделя) и 1.5 понадобилось 809 и 305 итераций соответственно, а методы Якоби и релаксации при омега равным 0.5 использовали максимальное количество итераций.