Отчет по дисциплине: «Численные методы»

Лабораторная работа №4

«**Итерационные методы решения СЛАУ**»

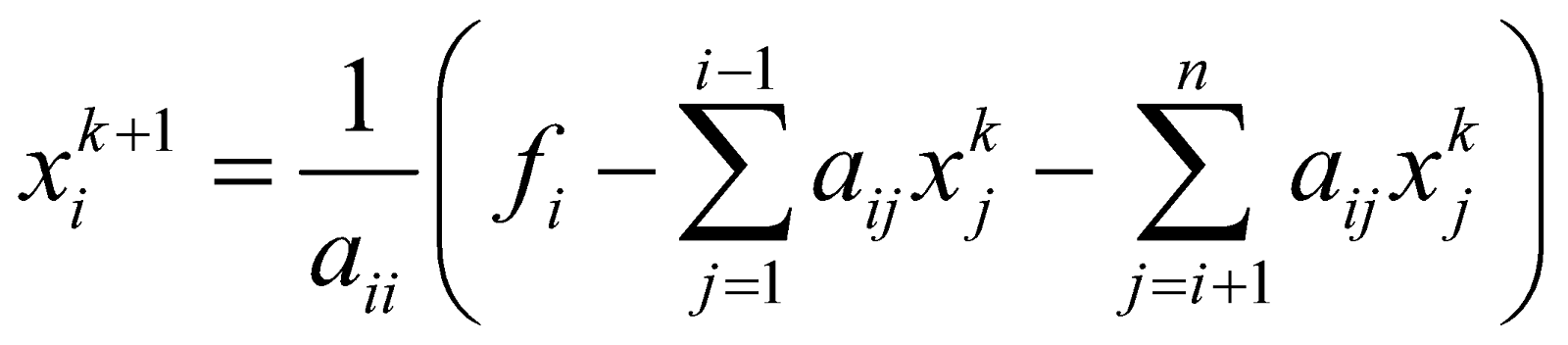
Подготовил студент 3 курса 4 группы

Кондратович Артём

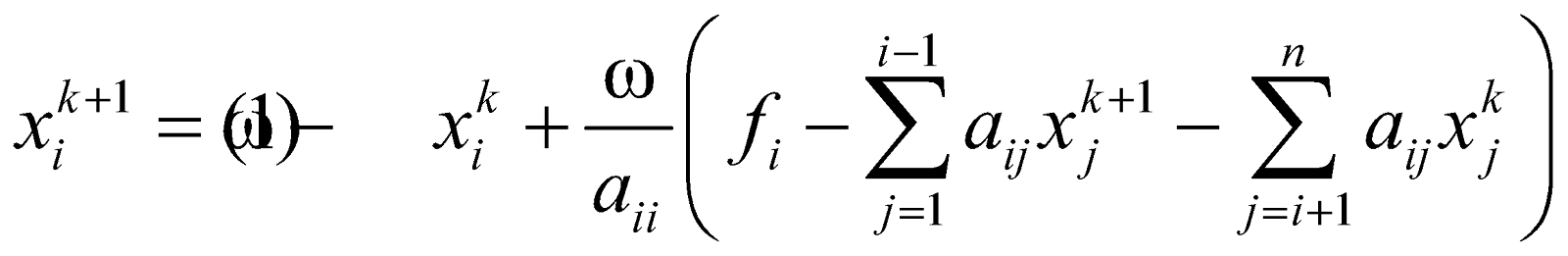
**Задача.**

Изучить основы методов Якоби, Гаусса-Зейделя, релаксации. Сравнить (для рассматриваемого примера) количество итераций, при которых каждым из этих методов достигнута требуемая точность.

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.

**Входные данные.**

n = 10, m = 9, max\_k = 1000, eps=0.0001

Листинг:

#include <iostream>

#include <chrono>

#include <iomanip>

#include <random>

const int m = 4;

const int max\_k = 10000;

const float epsilon = 0.0001f;

int GetRandomFromRange(int a, int b) // generate random fro range [a, b]

{

static std::random\_device rd;

static std::mt19937 gen(rd());

std::uniform\_int\_distribution<int> dis(a, b);

return dis(gen);

}

std::vector<std::vector<float>> GenerateMatrixA(int n)

{

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

for (size\_t i = 0; i < n ; i++)

{

for (size\_t j = 0; j < n; j++)

{

if (i != j)

{

matrix[i][j] = (float)GetRandomFromRange(-4, 0);

}

}

matrix[i][i] += i == 0 ? matrix[i][1] : matrix[i][i - 1];

matrix[i][i] -= i == 0 ? matrix[i][1] : matrix[i][i - 1];

matrix[i][i] += i == 0 ? 1.f : 0.f;

for (size\_t j = 0; j < n; j++)

{

if (i != j)

{

matrix[i][i] -= matrix[i][j];

}

}

}

return matrix;

}

std::vector<int> GenerateVectorX(int n)

{

std::vector<int> x(n);

x[0] = m;

for (size\_t i = 1; i < n; i++)

{

x[i] = x[i - 1] + 1;

}

return x;

}

std::vector<float> GenerateVectorF(std::vector<std::vector<float>> matrix, std::vector<int> x)

{

std::vector<float> f(x.size());

for (size\_t i = 0; i < x.size(); i++)

{

float sum = 0.;

for (size\_t j = 0; j < x.size(); j++)

{

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

}

f[i] = sum;

}

return f;

}

template<typename T>

std::ostream& operator <<(std::ostream& output, const std::vector<T>& vector)

{

for (size\_t i = 0; i < vector.size(); i++)

{

output << vector[i] << " ";

}

output << "\n";

return output;

}

std::ostream& operator <<(std::ostream& output, const std::vector<std::vector<float>>& matrix)

{

for(auto const& line : matrix)

{

for(auto element : line)

{

output << std::setw(3) << element << " ";

}

output << "\n";

}

return output;

}

std::vector<float>& operator /(std::vector<float>& vec, const int num)

{

for (size\_t i = 0; i < vec.size(); i++)

{

vec[i] /= (float)num;

}

return vec;

}

float GetError(const std::vector<float>& a, const std::vector<float>& b)

{

float max = 0.f;

for (size\_t i = 0; i < a.size(); i++)

{

max = std::max(max, std::abs(a[i] - b[i]));

}

return max;

}

void JacobiMethod(std::vector<std::vector<float>> a, std::vector<float> f,int k)

{

int n = f.size();

std::vector<float> x(n, 0.f); // Инициализация начального приближения

std::vector<float> x\_new(f.size()); // Вектор для хранения нового приближения

while (k < max\_k) {

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

float t = f[i];

for (int j = 0; j < a.size(); j++) {

if (j != i) {

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

}

}

x\_new[i] = t / a[i][i];

}

if (GetError(x, x\_new) < epsilon)

{

break;

}

x = x\_new; // Обновление приближения

k++;

}

// Вывод результатов

std::cout << "Jacobi method" << std::endl;

std::cout << "System solution:" << std::endl;

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

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

}

std::cout << "Iterations: " << k << std::endl << std::endl;

}

void RelaxationMethod(float w, const std::vector<std::vector<float>>& a, const std::vector<float>& f, int k)

{

int n = a.size();

std::vector<float> x(n, 0.f); // Инициализация начального приближения x

while (k < max\_k) {

std::vector<float> x\_new(n, 0.f); // Создание нового вектора x

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

float sum = 0.f;

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

{

sum += a[i][j] \* x\_new[j];

}

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

{

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

}

x\_new[i] = (1 - w) \* x[i] + (w / a[i][i]) \* (f[i] - sum);

}

if (GetError(x, x\_new) < epsilon)

{

break;

}

x = x\_new; // Обновление вектора x

k++; // Уменьшение числа итераций

}

// Вывод результатов

std::cout << "Relaxation method with w = " << w << std::endl;

std::cout << "System solution:" << std::endl;

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

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

}

std::cout << "Iterations: " << k << std::endl << std::endl;

}

int main()

{

auto n = GetRandomFromRange(10, 12);

std::cout << "Genrated n = " << n << std::endl;

auto a = GenerateMatrixA(n);

auto x = GenerateVectorX(n);

auto f = GenerateVectorF(a, x);

std::cout << "A\n" << a;

std::cout << "x = " << x;

std::cout << "f = " << f;

JacobiMethod(a, f, 0);

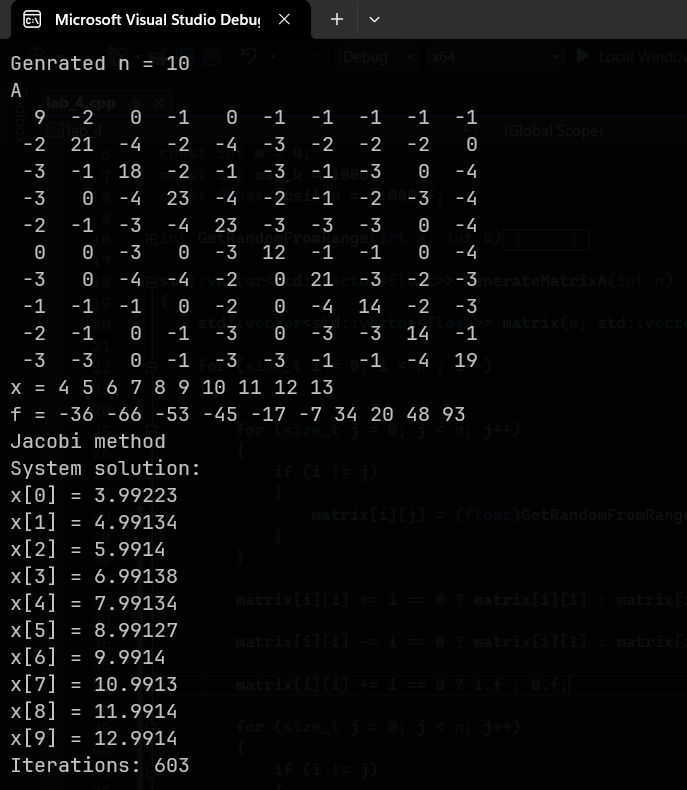
RelaxationMethod(0.5f, a,f,0);

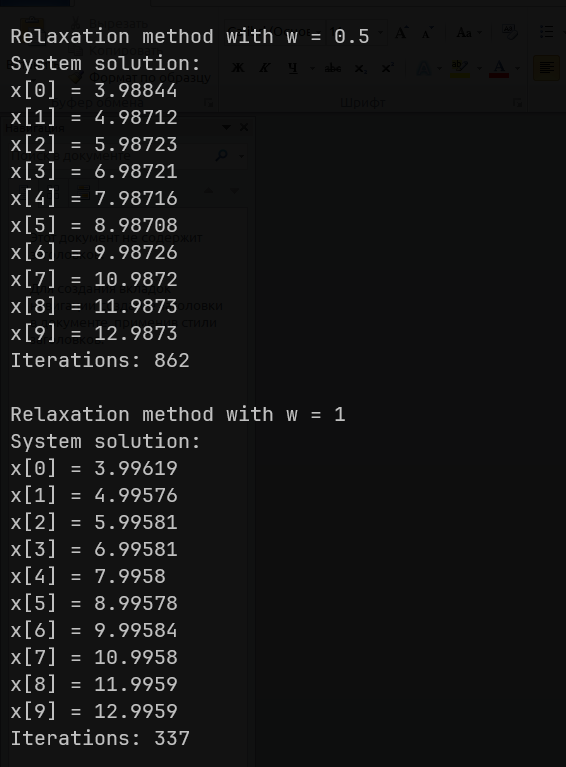
RelaxationMethod(1.f, a, f, 0);

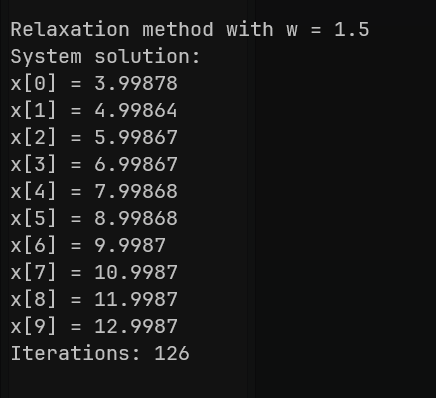
RelaxationMethod(1.5f, a, f, 0);

}

Результаты:







**Выводы.**

Изучили основы методов Якоби, Гаусса-Зейделя, релаксации. Сравнили (для рассматриваемого примера) количество итераций, при которых каждым из этих методов достигнута требуемая точность.