Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт вычислительной математики и информационных технологий

Кафедра прикладной математики

Направление подготовки: 01.03.02 – Прикладная математика и информатика

КУРСОВАЯ РАБОТА ПО ДИСЦИПЛИНЕ  
«ОПЕРАЦИОННЫЕ СИСТЕМЫ»

Студент группы 09-\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. подпись И.О. Фамилия

Научный руководитель

ассистент кафедры  
прикладной математики

"\_\_\_"\_\_\_\_\_\_\_\_\_ 2020 г. подпись Д.Х. Гиниятова

Казань 2020

Содержание

[**Постановка задачи** 2](#_Toc40916219)

[**Описание метода решения** 3](#_Toc40916220)

[**Последовательная реализация метода** 3](#_Toc40916221)

[**Параллельная реализация метода** 4](#_Toc40916222)

[**Результаты вычислений** 6](#_Toc40916223)

[**Заключение** 6](#_Toc40916224)

[**Список использованных источников** 7](#_Toc40916225)

[**Листинг программы** 8](#_Toc40916226)

# **Постановка задачи**

Рассмотрим систему линейных алгебраических уравнений

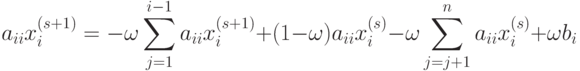
Ax=f, (1)

где A – квадратная матрица размерности n, x=(x1, x2, ..., xn)−вектор решения, f=(f1, f2, ..., fn) −вектор правых частей. Матрица А является симметричной и положительно определенной.

Метод верхней релаксации является итерационным. Суть итерационного метода состоит в том, что для решения системы (1) необходимо найти предел последовательных приближений x(n) при n→∞ (n −номер итерации). Применение итерационных методов требует задания начального значения неизвестных х(0) и точности вычислений ε>0. Вычисления проводятся до тех пор, пока не будет выполнена оценка (2).

||x(n) - x||< ε. (2)

У итерационных методом есть ключевая особенность: точность искомого решения задается. Для нахождения решения используется формула :



где s-номер текущей операции, s+1 – номер следующей операции, -число (параметр метода). Необходимым условием сходимости метода релаксации с любого начального приближения x0 к точному х\* является выполнение условия При этом если, то говорят *о методе нижней релаксации* , а при - *о методе верхней релаксации* , при метод релаксации будет совпадать с известным *методом Зейделя* .

Необходимо найти все решения с учетом точности. Реализовать метод с помощью программы, написанной на С++. Для ускорения используем технологию OpenMP.Выполнить сравнение работы последовательной и параллельной программ. Продемонстрировать разницу результатов на графике. Сделать вывод.

# 

# **Описание метода решения**

## **Последовательная реализация метода**

int Relax1\_metod(int kol, float\*\* A, float\* B, float\* x, float\* xn, int n, float eps, float w)

{

int j, k, i;

float norma = 0;

float sum1 = 0, sum2 = 0;

do

{

kol++;

norma = 0;

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

{

x[j] = x[j] + (w \* B[j] / A[j][j]) + ((1 - w) \* xn[j]);

for (k = j + 1; k < n; k++)

sum1 = sum1 + (A[j][k] \* xn[k] / A[j][j]);

x[j] = x[j] - (w \* sum1);

for (k = 0; k < j; k++)

sum2 = sum2 + (A[j][k] \* x[k] / A[j][j]);

x[j] = x[j] - (w \* sum2);

sum1 = 0;

sum2 = 0;

if (fabs(x[j] - xn[j]) > norma)

norma = fabs(x[j] - xn[j]);

}

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

{

xn[i] = x[i];

x[i] = 0;

}

cout << norma << endl;

}while (norma > eps);

return kol;}

## **Параллельная реализация метода**

int Relax2\_metod(int kol, float\*\* A, float\* B, float\* x, float\* xn, int n, float eps, float w)

{

int j, k, i, a = 0;

float norma = 0, t1, t2;

float sum1 = 0, sum2 = 0;

omp\_set\_num\_threads(8);

#pragma omp parallel shared(A,B,x,xn,a) private(i,j,k,sum1, sum2)

{

do

{

a++;

norma = 0;

#pragma omp parallel for

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

{

sum1 = 0;

sum2 = 0;

x[j] = x[j] + (w \* B[j] / A[j][j]) + ((1 - w) \* xn[j]);

for (k = j + 1; k < n; k++)

sum1 = sum1 + (A[j][k] \* xn[k] / A[j][j]);

x[j] = x[j] - (w \* sum1);

for (k = 0; k < j; k++)

sum2 = sum2 + (A[j][k] \* x[k] / A[j][j]);

x[j] = x[j] - (w \* sum2);

if (fabs(x[j] - xn[j]) > norma)

norma = fabs(x[j] - xn[j]);

}

#pragma omp parallel for

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

{

xn[i] = x[i];

x[i] = 0;

}

printf("Num threads: %d. Thread number %d\n", omp\_get\_num\_threads(), omp\_get\_thread\_num());

} while (norma > eps);

}

kol = a;

return kol;

}

# **Результаты вычислений**

Зависимость скорости вычисления от числа запущенных процессов

# **Заключение**

Можно сделать вывод, что если увеличивать количество одновременно работающих нитей в параллельной области при больших размерах матрицы, то время работы программы будет уменьшаться. Однако, если размеры матрицы невелики, то последовательная реализация программы будет эффективнее. К тому же, от выбора коэффициента релаксации зависит скорость программы(1<w<2). Решение можно получить достаточно точно, указав ε минимальным.

# **Список использованных источников**

1. <https://www.intuit.ru/studies/courses/4447/983/lecture/14931?page=9>
2. <http://www.itlab.unn.ru/uploads/num/Relax.pdf>
3. <http://www.itlab.unn.ru/uploads/num/Relax.pdf>

**Листинг программы**

#include <iostream>

#include<conio.h>

#include<math.h>

#include <fstream>

#include <omp.h>

#include <chrono>

using namespace std;

int Relax1\_metod(int kol, float\*\* A, float\* B, float\* x, float\* xn, int n, float eps, float w)

{

int j, k, i;

float norma = 0;

float sum1 = 0, sum2 = 0;

do

{

kol++;

norma = 0;

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

{

x[j] = x[j] + (w \* B[j] / A[j][j]) + ((1 - w) \* xn[j]);

for (k = j + 1; k < n; k++)

sum1 = sum1 + (A[j][k] \* xn[k] / A[j][j]);

x[j] = x[j] - (w \* sum1);

for (k = 0; k < j; k++)

sum2 = sum2 + (A[j][k] \* x[k] / A[j][j]);

x[j] = x[j] - (w \* sum2);

sum1 = 0;

sum2 = 0;

if (fabs(x[j] - xn[j]) > norma)

norma = fabs(x[j] - xn[j]);

}

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

{

xn[i] = x[i];

x[i] = 0;

}

cout << norma << endl;

}while (norma > eps);

return kol;

}

int Relax2\_metod(int kol, float\*\* A, float\* B, float\* x, float\* xn, int n, float eps, float w)

{

int j, k, i, a = 0;

float norma = 0, t1, t2;

float sum1 = 0, sum2 = 0;

omp\_set\_num\_threads(8);

#pragma omp parallel shared(A,B,x,xn,a) private(i,j,k,sum1, sum2)

{

do

{

a++;

norma = 0;

#pragma omp parallel for

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

{

sum1 = 0;

sum2 = 0;

x[j] = x[j] + (w \* B[j] / A[j][j]) + ((1 - w) \* xn[j]);

for (k = j + 1; k < n; k++)

sum1 = sum1 + (A[j][k] \* xn[k] / A[j][j]);

x[j] = x[j] - (w \* sum1);

for (k = 0; k < j; k++)

sum2 = sum2 + (A[j][k] \* x[k] / A[j][j]);

x[j] = x[j] - (w \* sum2);

if (fabs(x[j] - xn[j]) > norma)

norma = fabs(x[j] - xn[j]);

}

#pragma omp parallel for

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

{

xn[i] = x[i];

x[i] = 0;

}

printf("Num threads: %d. Thread number %d\n", omp\_get\_num\_threads(), omp\_get\_thread\_num());

} while (norma > eps);

}

kol = a;

return kol;

}

void Create\_Matr(float\*\* A, int n)

{

float s = 0;

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

{

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

{

if (i != j)

{

A[i][j] = rand() % 10;

s += A[i][j];

}

}

}

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

A[i][i] = (rand() % 100) + s;

}

void Create\_Vector(float\* B, int n)

{

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

{

B[i] = rand() % 10 + 1;

}

}

int main()

{

ifstream in("arb.txt");

setlocale(LC\_CTYPE, "RUSSIAN");

int k, n, i, j, kol = 0;

float eps, sum1 = 0, sum2 = 0, norma = 0;

float w;

cout << "Введите точность:";

cin >> eps;

cout << "Введите коэффициент релаксации (1,2):";

cin >> w;

in.close();

for (int n = 1000; n <= 10000; n += 1000)

{

float\* x = new float[n];

float\* xn = new float[n];

float\*\* A = new float\* [n];

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

A[i] = new float[n];

float\* B = new float[n];

Create\_Matr(A, n);

Create\_Vector(B, n);

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

{

xn[i] = B[i] / A[i][i];

x[i] = 0;

}

//auto begin = chrono::steady\_clock::now();

unsigned int start\_time = clock();

kol = Relax2\_metod(kol, A, B, x, xn, n, eps, w);

unsigned int end\_time = clock();

//auto end = chrono::steady\_clock::now();

//auto time = chrono::duration\_cast<chrono::seconds>(end - begin);

printf("Размер матрицы = %d. Время = %d. Количество итераций = %d\n", n, end\_time-start\_time, kol);

} return 0;}