# Технологии параллельных систем и распределенных вычислений

# Лабораторная работа №8

Одной из основных задач линейной алгебры является решение систем линейных алгебраических уравнений (СЛАУ):

или *A x* = *b* (1)

СЛАУ возникают при решении многих прикладных задач, а матрицы коэффициентов систем линейных уравнений могут иметь различные структуру и свойства. Будем считать, что матрица А неособенная, det A ≠ 0 т.е. решение системы (1) единственно.

Численные методы решения СЛАУ делятся на две большие группы: прямые и итерационные. Прямые методы при отсутствии ошибок округления за конечное число арифметических операций позволяют получить точное решение *x\**. В итерационных методах задается начальное приближение *x*0 и строится последовательность приближенных решений *xk → x\**, где *k* – номер итерации. Итерационный процесс прекращается, как только *xk* становится достаточно близким к *x*\*.

Итерационные методы привлекательнее с точки зрения объема вычислений и требуемой памяти, когда решаются системы с матрицами высокой размерности. При небольших порядках системы используют прямые методы либо прямые методы в сочетании с итерационными методами.

Одним из прямых методов решения линейных систем (1) является применение метода исключения Гаусса. Суть этого метода состоит в том, что матрица A сначала упрощается – приводится эквивалентными преобразованиями к треугольному или диагональному виду, а затем решается система с упрощенной матрицей. Наиболее известной формой гауссова исключения является та, в которой система линейных уравнений приводится к верхнетреугольному виду путём вычитания одних уравнений, умноженных на подходящие числа из других уравнений. Полученная треугольная система решается с помощью обратной подстановки.

Для тестирования работы программы предлагается следующий подход. Сгенерировать исходные матрицы размером [*n×n*], *n*=[3, 10, 100, 1000, 10000] с помощью программы, приведенной в лист. 1. Решить системы уравнений для заданных матриц методом Гаусса (лист. 2) и получить оценку зависимости времени выполнения от размерности матрицы. Выполнить распараллеливание кода программы с помощью технологии OpenMP. Оценить выигрыш по времени выполнения программы от ее распараллеливания. Для сравнения реализовать итерационный алгоритм решения СЛАУ (например https://ru.wikipedia.org/wiki/Метод\_Гаусса\_—\_Зейделя) и выполнить его распараллеливание. Сравнить результаты.

Листинг 1 — Программа, генерирующая файл с исходными матрицами СЛАУ

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

int main()

{ const int n = 100; // Размер матрицы

const char\* filename="./input100.dat"; // Имя файла

srand(time(NULL));

FILE \*f = fopen(filename,"wb");

if (f == NULL)

{ puts("can't open file");

return 0;

}

fwrite(&n,sizeof(n),1,f);

double B[n];

for (int j=0;j<n+1;j++)

{

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

B[i] = rand()%n;

fwrite(B,sizeof(B[0]),n,f);

}

fclose(f);

}

Листинг 2 — Программа, реалузующая метод Гаусса

#include <stdio.h>

#include <math.h>

const char\* filename="./input100.dat";

void ForwardSolve(int n, double \*\*a, double \*b)

{

double v;

int i,j,im;

for(int k = 0; k < n - 1; k++)

{

im = k;

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

if(fabs(a[im][k]) < fabs(a[i][k]))

im = i;

if(im != k)

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

{

v = a[im][j];

a[im][j] = a[k][j];

a[k][j] = v;

}

v = b[im];

b[im] = b[k];

b[k] = v;

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

{

v = 1.0\*a[i][k]/a[k][k];

a[i][k] = 0;

b[i] = b[i] - v\*b[k];

if(v != 0)

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

a[i][j] = a[i][j] - v\*a[k][j];

}

}

}

void BackSolve(int n, double \*\*a, double \*b, double \*x)

{

double s = 0;

x[n - 1] = 1.0\*b[n - 1]/a[n - 1][n - 1];

for(int i = n - 2, j; 0 <= i; i--)

{

s = 0;

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

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

x[i] = 1.0\*(b[i] - s)/a[i][i];

}

}

int main()

{

FILE\* f = fopen(filename,"rb");

if (f == NULL)

{

puts("can't open file");

return 0;

}

int n = 0;

fread(&n,sizeof(n),1,f);

printf("\n size of matrix: %d x %d", n, n);

if (n < 0)

{

puts("Error n<0");

return 0;

}

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

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

A[i] = new double[n];

double \*x = new double[n];

double \*B = new double[n];

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

fread(A[i],sizeof(A[0][0]),n,f);

fread(B,sizeof(B[0]),n,f);

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

{

puts("\n");

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

printf("%g\t",A[i][j]);

printf("%g\t",B[i]);

}

ForwardSolve(n, A, B);

BackSolve(n,A,B,x);

puts("\n");

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

printf("%g\n",x[i]);

delete []B;

delete []x;

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

delete []A[i];

delete []A;

return(0);

}