**«Реализация метода Гаусса»**

1. Постановка задачи

Разработать программу для демонстрации реализованного метода Гаусса (умножение матриц) при помощи параллельного программирования.

1. Краткое описание

Основная идея метода Гаусса: приведение матрицы A к верхнему треугольному виду с помощью эквивалентных преобразований.

Метод Гаусса – прямой ход:

На итерации i, 1≤ i ≤ n, метода производится исключение неизвестной i для всех уравнений с номерами k, i ≤ k ≤ n. Для этого из этих уравнений осуществляется вычитание строки i, умножение на константу (aki / aii), чтобы результирующий коэффициент при неизвестной xi в строках оказался нулевым.

Все необходимые вычисления определяются при помощи соотношений:

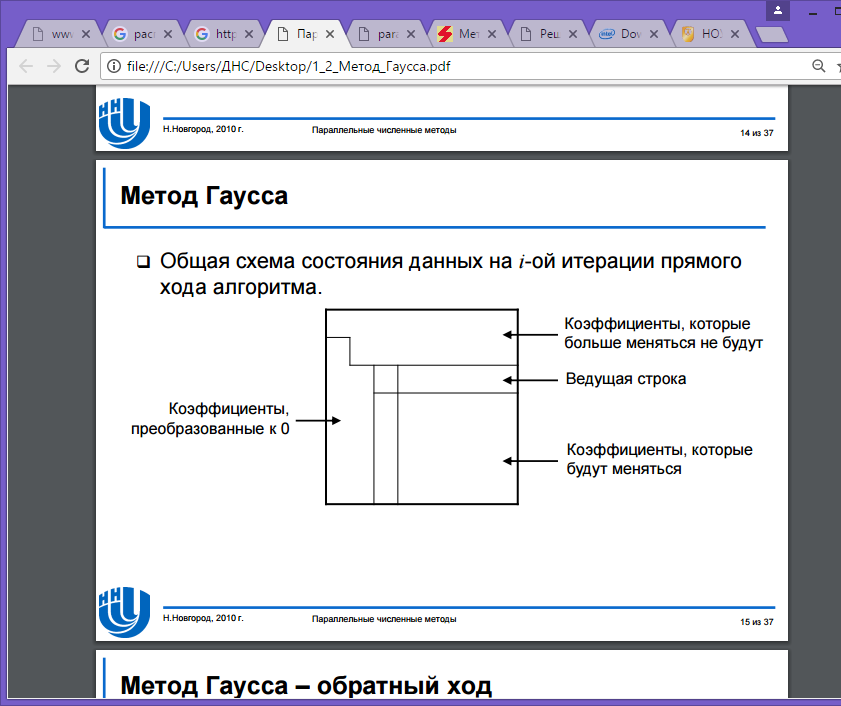
a'kj = akj - µki \* aij,

i ≤ j ≤ n, i < k ≤ n, 1 ≤ i < n

b’k = bk - µki \* bi,

где µki = aki / aii – множители Гаусса.

Общая схема состояния данных на i-ой итерации прямого хода алгоритма.

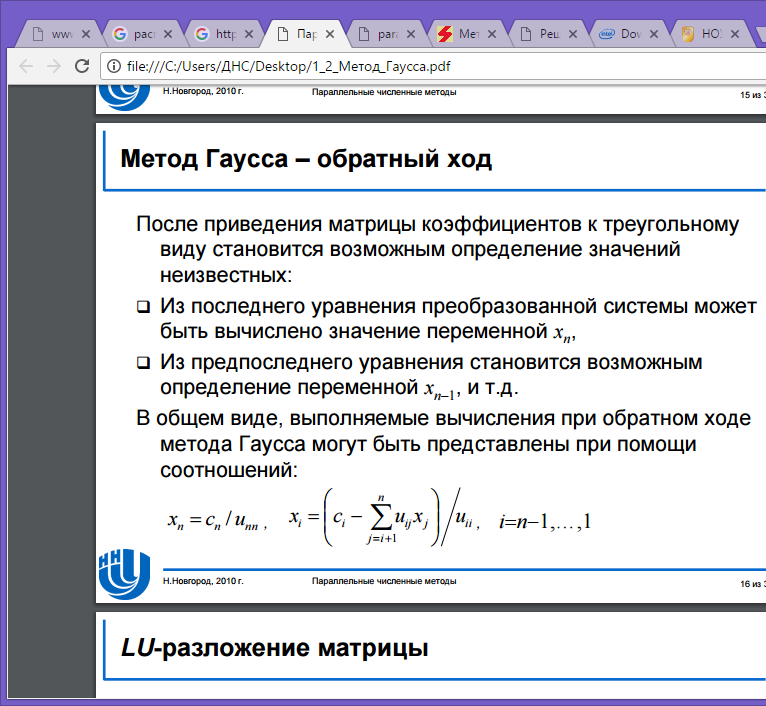


Метод Гаусса – обратный ход:

После приведения матрицы коэффициентов к треугольному виду становится возможным определение значений неизвестных:

* Из последнего уравнения преобразованной системы может быть вычислено значение переменной xn,
* Из последнего уравнения становится возможным определение переменной xn-1 и т.д.

В общем виде, выполняемые вычисления при обратном ходе метода Гаусса могут быть представлены при помощи соотношений:



1. Ожидаемые и полученные результаты совпали

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

//#include "main.h"

#include <iostream>

//#include<sys\time.h>

#include <ctime>

using namespace std;

#define M 4

double MA[M][M + 1], V[M];

int main()

{

setlocale(LC\_ALL, "");

int i, j, p;

double a;

//struct timeval tv\_start, tv\_end;

clock\_t t0 = clock();

cout << "\nМатрица коэффициентов:\n";

for (i = 0; i < M; i++) {

for (j = 0; j < M; j++) {

if (i == j)

MA[i][j] = 2.0;

else

MA[i][j] = 1.0;

cout << MA[i][j] << " ";

}

cout << endl;

}

cout << "\nСгенерированные значения х:\n";

for (j = 0; j < M; j++) {

V[j] = -(double)(j + 1) / 2.;

cout << V[j] << " ";

}

cout << endl;

cout << "\nСвободные члены (b):\n";

for (i = 0; i < M; i++) {

MA[i][M] = 0.0;

for (j = 0; j < M; j++) {

MA[i][M] += MA[i][j] \* V[j];

cout << MA[i][M] << " ";

}

cout << endl;

}

/\* Прямой ход \*/

for (p = 0; p < M; p++) {

a = MA[p][p];

for (i = p; i <= M; i++)

MA[p][i] = MA[p][i] / a;

/\* Цикл k - цикл по строкам. \*/

#pragma omp parallel for private (i, j, a)

for (j = p + 1; j < M; j++) {

a = MA[j][p];

for (i = p; i <= M; i++)

MA[j][i] = MA[j][i] - a \* MA[p][i];

}

}

/\* Обратный ход \*/

for (p = M - 1; p >= 0; p--) {

#pragma omp parallel for private (i, j)

for (j = p - 1; j >= 0; j--) {

for (i = M; i > j; i--)

MA[j][i] = MA[j][i] - MA[j][p] \* MA[p][i];

}

}

clock\_t t1 = clock();

cout << "\nВычисленные значения х:\n";

for (j = 0; j < M; j++) {

cout << MA[j][M] << " ";

}

cout << endl;

cout << "\nВремя прямого хода: " << (double)(t1 - t0) / CLOCKS\_PER\_SEC << endl;

system("pause");

return 0;

}

