

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной техники

ОТЧЕТ
по лабораторной работе №5
по дисциплине «Параллельные Вычисления»
Тема: «Численные методы»

Студент гр. 1307

Тростин М. Ю.

Преподаватель

Санкт-Петербург

2025

Цели и задачи

Цель: Приобрести навыки в распараллеливании программы.

Задание (по вариантам). Представить последовательный и параллельный вариант программы, реализующей метод приведения разреженной матрицы к треугольному виду.

Выполнение работы

В данном задании «главный процесс» (с rank = 0) генерирует квадратную матрицу в котором количество элементов зависит от количества запущенных процессов. И рассылает её остальным массивам.

Для приведения матрицы к треугольному виду удобно использовать метод Гаусса.

1. на первом шаге с помощью первого уравнения исключается x_1 из всех последних уравнений системы, в результате получается новая система, имеющая то же решение, но в первом столбце матрицы будет не нулевым только первый элемент.
2. на втором шаге с помощью второго уравнения исключается x_2 из всех уравнений. Этот процесс продолжается до тех пор, пока в левой части последнего n-го уравнения не останется лишь один член с неизвестным x_n .

В непараллельной программе временная сложность алгоритма составляет (в худшем случае) $O(n^3)$, однако при распараллеливании программы мы получаем сложность: $O(n^2 \log(n))$

Исходный код доступен в Приложении А

Скриншот выполнения

```
mt@MacBook-Pro-MT lab5 % mpirun -n 5 ./task0.o
Generated matrix:
6.00 -5.00 -9.00 -1.00 5.00
6.00 8.00 -6.00 -6.00 -5.00
0.00 8.00 -2.00 5.00 7.00
-3.00 -7.00 7.00 -6.00 -8.00
-2.00 2.00 6.00 -9.00 -5.00
Row echelon form of original matrix:
1.00 -0.83 -1.50 -0.17 0.83
0.00 1.00 0.23 -0.38 -0.77
0.00 0.00 1.00 -2.10 -3.42
0.00 0.00 0.00 1.00 -10.80
0.00 0.00 0.00 0.00 1.00
Time elapsed: 0.001042s
```

Результаты тестирования

Номер теста	Кол-во процессов	Время выполнения (с)
1	4	0.001109
2	4	0.000768
3	6	0.001021
4	6	0.001411
5	8	0.012008
6	8	0.009145
7	16	0.356410
8	16	0.341053
9	32	1.400224
10	32	1.550725

Выводы

В рамках выполнения данной работы были получены навыки в распараллеливании программы

ПРИЛОЖЕНИЕ А

Задание 1. Файл task1.cpp

```
#include <stdio.h>
#include <mpi.h>
#include <random>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>

#define element_t float
#define row_t      float *
#define matrix_t   float **

#define error_t unsigned char
#define ERR_VAL 1
#define ERR_CLK 2

#define BILLION 1000000000L
typedef struct SqareMatrix {
    matrix_t data;
    size_t size;
} SqareMatrix;

void smPrint(SqareMatrix sm) {
    for (size_t i = 0; i < sm.size; i++) {
        for (size_t j = 0; j < sm.size; j++) {
            printf("%.2f ", sm.data[i][j]);
        }
        printf("\n");
    }
}

SqareMatrix smEmpty(size_t size) {
    SqareMatrix sm;
    sm.size = size;
    sm.data = (matrix_t)malloc(sizeof(row_t) * sm.size);

    for (int i = 0; i < sm.size; i++) {
        sm.data[i] = (row_t)malloc(sizeof(element_t) * sm.size);
        for (int j = 0; j < sm.size; j++) {
            sm.data[i][j] = 0;
        }
    }

    return sm;
}

SqareMatrix smGenerate(size_t size) {
    srand(time(NULL));
    SqareMatrix sm;
```

```

sm.size = size;
sm.data = (matrix_t)malloc(sizeof(row_t) * sm.size);

for (int i = 0; i < sm.size; i++) {
    sm.data[i] = (row_t)malloc(sizeof(element_t) * sm.size);
    for (int j = 0; j < sm.size; j++) {
        sm.data[i][j] = (element_t)(rand() % 19 - 9);
    }
}

return sm;
}

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);

    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    struct timespec start, stop;
    if (rank == 0) {
        if (clock_gettime(CLOCK_REALTIME, &start) == -1) {
            printf("could not get clock time");
            return ERR_CLK;
        }
    }

    if (size < 2) {
        if (rank == 0) printf("Expected more than one process\n");
        MPI_Finalize();
        return ERR_VAL;
    }

    SqaureMatrix sm = rank == 0
        ? smGenerate(size)
        : smEmpty(size);

    if (rank == 0) {
        printf("Generated matrix:\n");
        smPrint(sm);
    }

    for (int i = 0; i < sm.size; i++) {
        MPI_Bcast(sm.data[i], sm.size, MPI_FLOAT, 0, MPI_COMM_WORLD);
    }

    for (int pivot = 0; pivot < sm.size; pivot++) {
        if (rank == pivot) {
            element_t pval = sm.data[pivot][pivot];

```

```

        for (int j = pivot; j < sm.size; j++) {
            sm.data[pivot][j] /= pval;
        }
    }

    MPI_Bcast(sm.data[pivot], sm.size, MPI_FLOAT, pivot,
MPI_COMM_WORLD);
    for (int i = pivot + 1; i < sm.size; i++) {
        if (rank == i) {
            element_t factor = sm.data[i][pivot];
            for (int j = pivot; j < sm.size; j++) {
                sm.data[i][j] -= factor * sm.data[pivot][j];
            }
        }
    }
}

if (rank == 0) {
    if (clock_gettime(CLOCK_REALTIME, &stop) == -1) {
        printf("could not get stop clock time");
        return ERR_CLK;
    }

    printf("Row echelon form of original matrix:\n");
    smPrint(sm);

    double accum = ( stop.tv_sec - start.tv_sec )
        + (double)( stop.tv_nsec - start.tv_nsec )
        / (double)BILLION;
    printf( "Time elapsed: %lfs\n", accum);
}

MPI_Finalize();
return 0;
}

```