



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

# Отчёт

## по лабораторной работе № 4

Название: Параллельное умножение матриц

Дисциплина: Анализ алгоритмов

Студент

ИУ7-55Б

(Группа)

Д.О. Склифасовский

(Подпись, дата)

(И.О. Фамилия)

Преподаватель

Л.Л. Волкова

(Подпись, дата)

(И.О. Фамилия)

*Москва, 2020*

## Содержание

<b>Введение</b>	<b>3</b>
<b>1 Аналитический раздел</b>	<b>4</b>
1.1 Общие сведения . . . . .	4
1.2 Алгоритм Винограда . . . . .	4
1.3 Параллельное программирование . . . . .	5
1.4 Параллельный алгоритм Винограда . . . . .	5
1.5 Вывод . . . . .	6
<b>2 Конструкторский раздел</b>	<b>7</b>
2.1 Разработка алгоритма . . . . .	7
2.2 Распараллеливание программы . . . . .	8
2.3 Вывод . . . . .	8
<b>3 Технологический раздел</b>	<b>9</b>
3.1 Общие требования . . . . .	9
3.2 Средства реализации . . . . .	9
3.3 Сведения о модулях программы . . . . .	9
3.4 Листинг кода программы . . . . .	10
3.5 Вывод . . . . .	15
<b>Литература</b>	<b>16</b>

## Введение

Цель работы: изучение возможности параллельных вычислений и использование такого подхода на практике.

В ходе лабораторной работы требуется:

- 1) выбрать алгоритм для рассмотрения;
- 2) описать обычную версию;
- 3) выполнить 2 параллельные версии;
- 4) запустить эксперименты на каждый при различном числе потоков 1,2,4,8 ...4M, где M - количество логических ядер на компьютере;
- 5) проверить, всегда ли при росте потоков, время работы снижается;
- 6) описать главный и рабочий поток схемой;
- 7) описать точки сборки.

В данной лабораторной работе был выбран алгоритм Винограда. Необходимо сравнить зависимость времени работы алгоритма от числа параллельных потоков и размера матриц, провести сравнение стандартного и параллельного алгоритмов.

## 1 Аналитический раздел

В данном разделе представлено описание стандартного и параллельного алгоритмов Винограда.

### 1.1 Общие сведения

Матрица — математический объект, записываемый в виде прямоугольной таблицы элементов кольца или поля (например, целых, действительных или комплексных чисел), которая представляет собой совокупность строк и столбцов, на пересечении которых находятся её элементы. Количество строк и столбцов задает размер матрицы. Хотя исторически рассматривались, например, треугольные матрицы, в настоящее время говорят исключительно о матрицах прямоугольной формы, так как они являются наиболее удобными и общими.

Умножение матриц — одна из основных операций над матрицами. Матрица, получаемая в результате операции умножения, называется произведением матриц.

### 1.2 Алгоритм Винограда

Если посмотреть на результат умножения двух матриц, то видно, что каждый элемент в нем представляет собой скалярное произведение соответствующих строки и столбца исходных матриц. Можно заметить также, что такое умножение допускает предварительную обработку, позволяющую часть работы выполнить заранее. [1]

Рассмотрим два вектора  $V = (v_1, v_2, v_3, v_4)$  и  $W = (w_1, w_2, w_3, w_4)$ . Их скалярное произведение равно:

$$V * W = v_1w_1 + v_2w_2 + v_3w_3 + v_4w_4 \quad (1)$$

Это равенство можно переписать в виде:

$$V * W = (v_1 + w_2)(v_2 + w_1) + (v_3 + w_4)(v_4 + w_3) - v_1 v_2 - v_3 v_4 - w_1 w_2 - w_3 w_4 \quad (2)$$

Менее очевидно, что выражение в правой части последнего равенства допускает предварительную обработку: его части можно вычислить заранее и запомнить для каждой строки первой матрицы и для каждого столбца второй. На практике это означает, что над предварительно обработанными элементами придется выполнять лишь первые два умножения и последующие пять сложений, а также дополнительно два сложения.

### 1.3 Параллельное программирование

Параллельное программирование. Параллельное программирование служит для создания программ, эффективно использующих вычислительные ресурсы за счет одновременного исполнения кода на нескольких вычислительных узлах. Для создания параллельных приложений используются параллельные языки программирования и специализированные системы поддержки параллельного программирования, такие как MPI и OpenMP. Параллельное программирование является более сложным по сравнению с последовательным как в написании кода, так и в его отладки. Для облегчения процесса параллельного программирования существуют специализированные инструменты, например, отладчик TotalView.[2]

### 1.4 Параллельный алгоритм Винограда

Для того, чтобы добиться большей эффективности алгоритма умножения матриц Винограда, необходимо распараллелить ту часть алгоритма, которая содержит больше всего вложенных циклов. Вычисление результата для каждой строки не зависит от результата умножения для других строк. Каждый поток будет выполнять вычисления определенных строк результирующей матрицы.

Для проверки эффективности выбранного метода, также можно распараллелить циклы двойной вложенности в начале алгоритма (циклы вычисления векторов  $\text{mulH}$  и  $\text{mulV}$ ).

## 1.5 Вывод

Было представлено математическое описание стандартного и параллельного алгоритмов Винограда. Основная идея заключается в том, чтобы распараллелить главный цикл тройной вложенности.

## 2 Конструкторский раздел

В данном разделе представлены схемы разработанных алгоритмов. Также описывается часть алгоритма, которая будет распараллеливаться.

### 2.1 Разработка алгоритма

На рисунке 1 изображена схема стандартного алгоритма умножения матриц.

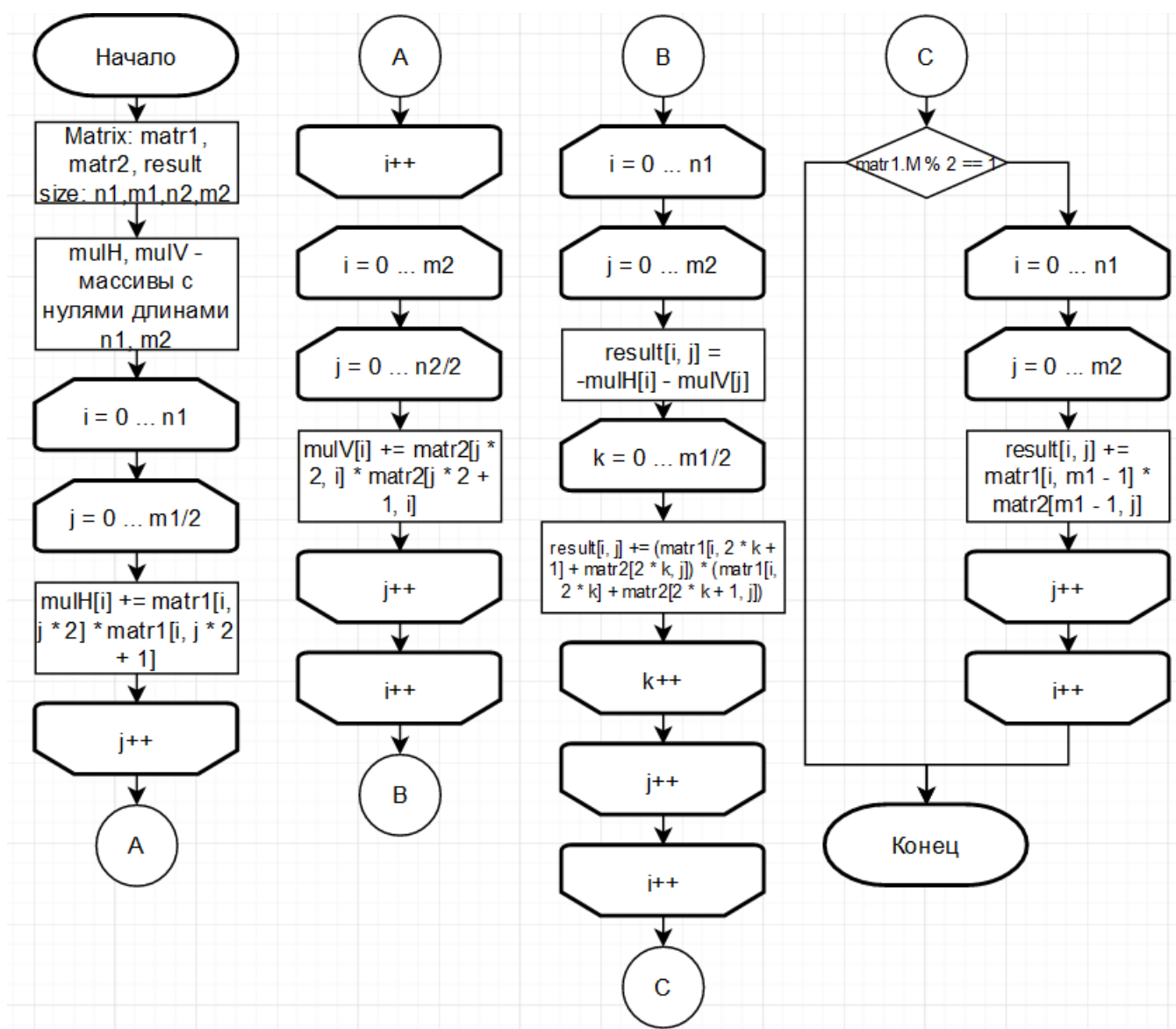


Рисунок 1 – Схема стандартного алгоритма умножения матриц

## **2.2 Распараллеливание программы**

В представленном алгоритме будет распараллеливаться цикл тройной вложенности (участок между В и С). Это должно ускорить время работы алгоритма.

Также для проверки эффективности будут распараллеливаться два первых цикла отдельно. Будут построены графики для данной реализации и сравниться время работы алгоритмов.

## **2.3 Вывод**

В данном разделе была рассмотрена схема алгоритма Винограда и описалась часть алгоритма, которая будет распараллеливаться.



### 3 Технологический раздел

В данном разделе даны общие требования к программе, средства реализации и сама реализация алгоритмов.

#### 3.1 Общие требования

##### Требования к вводу:

- 1) вводятся размеры матриц;
- 2) вводятся (или автоматически генерируются) матрицы.

##### Требования к программе

- 1) при вводе неправильных размеров матриц программа не должна завершиться аварийно;
- 2) должно выполняться корректное умножение матриц.

#### 3.2 Средства реализации

В качестве языка программирования был выбран C#, [3] так как я знаком с данным языком программирования, имею представление о способах тестирования программы.

Средой разработки Visual Studio. [4]

Для замеров процессорного времени используется функция *Stopwatch*. [5]

Многопоточное программирование было реализовано с помощью пространства имен *System.Threading*. [6]

#### 3.3 Сведения о модулях программы

Программа состоит из:

- 1) Program.cs - главный файл программы, в котором располагается точка входа в программу;

- 2) Matrix.cs - файл класса Matrix. Класс реализует матрицу размером  $n * m$ , а также он содержит методы для работы с матрицами;
- 3) Parameters.cs - файл классов ParametersForFirst и ParametersForSecond, которые используются для передачи данных функциям при параллелизации циклов;
- 4) MultMatr.cs - файл класса MultMatr. В нем находится алгоритм Винограда;
- 5) ParallelMultMatr.cs - файл класса ParallelMultMatr, в котором находятся параллельные реализации алгоритма Винограда.

### 3.4 Листинг кода программы

**Листинг 1** – Стандартный алгоритм Винограда

```
1 public static Matrix VinogradMult(Matrix matr1, Matrix matr2)
2 {
3     int n1 = matr1.N;
4     int n2 = matr2.N;
5     int m1 = matr1.M;
6     int m2 = matr2.M;
7     if ((m1 != n2) || n1 == 0 || n2 == 0)
8     {
9         return null;
10    }
11
12    Matrix result = new Matrix(n1, m2);
13    int[] mulH = new int[n1];
14    int[] mulV = new int[m2];
15
16    for (int i = 0; i < n1; i++)
17    {
18        for (int j = 0; j < m1 / 2; j++)
```

```

19         {
20             mulH[i] += matr1[i, j * 2] * matr1[i, j * 2 + 1];
21         }
22     }
23
24     for (int i = 0; i < m2; i++)
25     {
26         for (int j = 0; j < n2 / 2; j++)
27         {
28             mulV[i] += matr2[j * 2, i] * matr2[j * 2 + 1, i];
29         }
30     }
31
32     for (int i = 0; i < n1; i++)
33     {
34         for (int j = 0; j < m2; j++)
35         {
36             result[i, j] = -mulH[i] - mulV[j];
37             for (int k = 0; k < m1 / 2; k++)
38             {
39                 result[i, j] += (matr1[i, 2 * k + 1] + matr2[2
40                     * k, j]) * (matr1[i, 2 * k] + matr2[2 * k
41                     + 1, j]);
42             }
43         }
44     }
45
46     if (m1 % 2 == 1)
47     {
48         for (int i = 0; i < n1; i++)
49         {
50             for (int j = 0; j < m2; j++)
51             {
52                 result[i, j] += matr1[i, m1 - 1] * matr2[m1 -
53                     1, j];

```

```

51         }
52     }
53 }
54
55     return result;
56 }

```

## Листинг 2 – Параллельный алгоритм Винограда

```

1  public static Matrix ParallelVinogradMult1(Matrix matr1,
2      Matrix matr2, int countOfThreads)
3  {
4      int n1 = matr1.N;
5      int n2 = matr2.N;
6      int m1 = matr1.M;
7      int m2 = matr2.M;
8      if ((m1 != n2) || n1 == 0 || n2 == 0)
9      {
10         return null;
11     }
12
13     Matrix result = new Matrix(n1, m2);
14     int[] mulH = new int[n1];
15     int[] mulV = new int[m2];
16
17     for (int i = 0; i < n1; i++)
18     {
19         for (int j = 0; j < m1 / 2; j++)
20         {
21             mulH[i] += matr1[i, j * 2] * matr1[i, j * 2 + 1];
22         }
23     }
24
25     for (int i = 0; i < m2; i++)
26     {
27         for (int j = 0; j < n2 / 2; j++)

```

```

27         {
28             mulV[i] += matr2[j * 2, i] * matr2[j * 2 + 1, i];
29         }
30     }
31
32     Thread[] threads = new Thread[countOfThreads];
33     int distribution = n1 / countOfThreads;
34     int startl = 0;
35     for (int i = 0; i < countOfThreads; i++)
36     {
37         int endl = startl + distribution;
38         if (i == countOfThreads - 1)
39         {
40             endl = n1;
41         }
42         ParametersForFirst param = new ParametersForFirst(
43             result, matr1, matr2, mulV, mulH, startl, endl, m2,
44             m1);
45         threads[i] = new Thread(ComputationLoopForMain);
46         threads[i].Start(param);
47         startl = endl;
48     }
49     foreach (Thread curThread in threads)
50     {
51         curThread.Join();
52     }
53
54     if (m1 % 2 == 1)
55     {
56         for (int i = 0; i < n1; i++)
57         {
58             for (int j = 0; j < m2; j++)
59             {
60                 result[i, j] += matr1[i, m1 - 1] * matr2[m1 -
61                     1, j];

```

```

59         }
60     }
61 }
62
63     return result;
64 }

```

### Листинг 3 – Распараллеленный главный цикл

```

1  private static void ComputationLoopForMain (object curObj)
2  {
3      ParametersForFirst param = (ParametersForFirst)curObj;
4      int startl = param.startl,
5          endl = param.endl,
6          m2 = param.m2,
7          m1 = param.m1;
8      Matrix res = param.res,
9          matr1 = param.matr1,
10         matr2 = param.matr2;
11     int[] mulH = param.mulH,
12         mulV = param.mulV;
13
14     for (int i = startl; i < endl; i++)
15     {
16         for (int j = 0; j < m2; j++)
17         {
18             res[i, j] = -mulH[i] - mulV[j];
19             for (int k = 0; k < m1 / 2; k++)
20             {
21                 res[i, j] += (matr1[i, 2 * k + 1] + matr2[2 *
22                     k, j]) * (matr1[i, 2 * k] + matr2[2 * k +
23                     1, j]);
24             }
25         }
26     }
27 }

```

---

### 3.5 Вывод

В данном разделе были даны общие требования к программе, описаны средства реализации, были представлены сведения о модулях программы, а также реализованы алгоритмы умножения матриц (Винограда и параллельный Винограда).

## Литература

1. Умножение матриц. -URL: <http://www.algolib.narod.ru/Math/Matrix.html> (дата обращения: 24.10.2020). -Текст: электронный.
2. Параллельное программирование. -URL: <https://www.viva64.com/ru/t/0038/> (дата обращения: 24.10.2020). -Текст: электронный.
3. Документация по C#. -URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/> (дата обращения: 24.10.2020). -Текст: электронный.
4. Документация по семейству продуктов Visual Studio. -URL: <https://docs.microsoft.com/ru-ru/visualstudio/?view=vs-2019> (дата обращения: 01.10.2020). -Текст: электронный.
5. Stopwatch Класс. -URL: <https://goo.su/2e99> (дата обращения: 24.10.2020). -Текст: электронный.
6. Под капотом у Stopwatch. -URL: <https://habr.com/ru/post/226279/> (дата обращения: 24.10.2020). Текст: электронный.
7. Thread Класс. -URL: <https://docs.microsoft.com/ru-ru/dotnet/api/system.threading.thread> 3.1 (дата обращения 24.10.2020). -Текст: электронный.