



**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»

ДИСЦИПЛИНА «Анализ алгоритмов»

**Лабораторная работа № 2**

**Тема** Алгоритмы умножения матриц

**Студент** Воякин А. Я.

**Группа** ИУ7-54Б

**Преподаватели** Волкова Л. Л., Строганов Ю. В.

Москва.  
2020 г.

# Оглавление

<b>Введение</b>	<b>4</b>
<b>1 Аналитическая часть</b>	<b>5</b>
1.1 Классический алгоритм умножения матриц . . . . .	5
1.2 Алгоритм Винограда . . . . .	6
1.3 Вывод . . . . .	6
<b>2 Конструкторская часть</b>	<b>7</b>
2.1 Техническое задание . . . . .	7
2.2 Схемы алгоритмов . . . . .	7
2.3 Трудоемкость алгоритмов . . . . .	11
2.3.1 Классический алгоритм . . . . .	11
2.3.2 Алгоритм Винограда . . . . .	11
2.3.3 Оптимизированный алгоритм Винограда . . . . .	12
2.4 Вывод . . . . .	12
<b>3 Технологическая часть</b>	<b>13</b>
3.1 Выбор ЯП . . . . .	13
3.2 Реализации алгоритмов . . . . .	13
3.2.1 Оптимизация алгоритма Винограда . . . . .	15
3.3 Вывод . . . . .	16
<b>4 Исследовательская часть</b>	<b>17</b>
4.1 Сравнительный анализ на основе замеров времени работы алгоритмов . . . . .	17
4.2 Вывод . . . . .	18
<b>Заключение</b>	<b>19</b>



# Введение

Цель работы - изучение алгоритмов умножения матриц. В данной лабораторной работе рассматривается стандартный алгоритм умножения матриц, алгоритм Винограда и модифицированный алгоритм Винограда. Также требуется провести расчет сложности алгоритмов, получить навыки в оптимизации алгоритмов. Используемые алгоритмы активно применяются во всех областях, применяющих линейную алгебру, таких как:

- компьютерная графика;
- физика;
- экономика;
- прочее.

В ходе лабораторной работы предстоит:

- изучить алгоритмы умножения матриц: стандартный и алгоритм Винограда;
- улучшить алгоритм Винограда;
- дать теоретическую оценку базового алгоритма умножения матриц, алгоритма Винограда и улучшенного алгоритма Винограда;
- реализовать три алгоритма умножения матриц на одном из языков программирования;
- сравнить алгоритмы умножения матриц.

# 1 | Аналитическая часть

Матрица - математический объект, эквивалентный двумерному массиву. Числа располагаются в матрице по строкам и столбцам. Если число столбцов в первой матрице совпадает с числом строк во второй, то эти две матрицы можно перемножить. У произведения будет столько же строк, сколько в первой матрице, и столько же столбцов, сколько во второй. Постановка задачи перемножения матриц описана в [1].

## 1.1 Классический алгоритм умножения матриц

Пусть даны две прямоугольные матрицы А и В размерности m на n и n на l соответственно:

$$\begin{bmatrix} a_{1,1} & \dots & a_{1,n} \\ \dots & \dots & \dots \\ a_{m,1} & \dots & a_{m,n} \end{bmatrix}$$

$$\begin{bmatrix} b_{1,1} & \dots & b_{1,l} \\ \dots & \dots & \dots \\ b_{n,1} & \dots & b_{n,l} \end{bmatrix}$$

В результате получим матрицу С размерности m на l:

$$\begin{bmatrix} c_{1,1} & \dots & c_{1,l} \\ \dots & \dots & \dots \\ c_{m,1} & \dots & c_{m,l} \end{bmatrix}$$

$c_{i,j} = \sum_{r=1}^n a_{i,r} \cdot b_{r,j}$  называется произведением матриц  $A$  и  $B$ .

## 1.2 Алгоритм Винограда

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:

$$V \cdot W = v_1 \cdot w_1 + v_2 \cdot w_2 + v_3 \cdot w_3 + v_4 \cdot w_4$$

Это равенство можно переписать в виде:

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

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. Это означает, что над предварительно обработанными элементами нам придется выполнять лишь первые два умножения и последующие пять сложений, а также дополнительно два сложения. Подробное описание алгоритма Винограда можно найти в [2].

## 1.3 Вывод

Были рассмотрены алгоритмы классического умножения матриц и алгоритм Винограда, основная отличительная черта которого — наличие предварительной обработки, а также уменьшение количества операций умножения.

## 2 | Конструкторская часть

### 2.1 Техническое задание

**Требования к вводу:**

- На вход подаются размерности матриц и сами матрицы.

**Требования к выводу:**

- Корректное произведение введённых матриц или сообщение об ошибке в случае некорректного ввода.

### 2.2 Схемы алгоритмов

В данной части будут рассмотрены схемы алгоритмов. Схема классического алгоритма умножения матриц показана на рисунке 2.1, схема алгоритма Винограда - на рисунке 2.2, схема оптимизированного алгоритма Винограда - на рисунке 2.3.

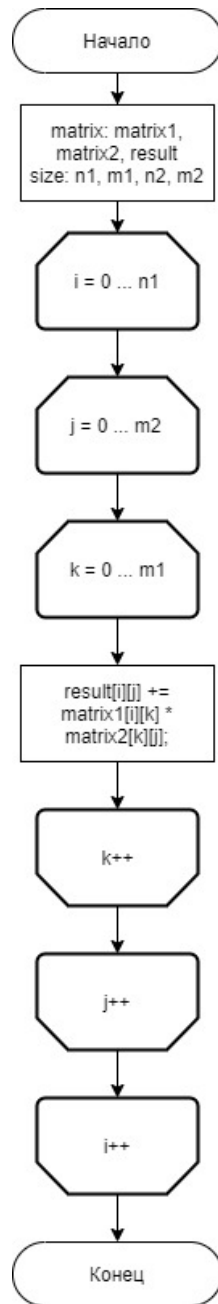


Рис. 2.1: Схема классического алгоритма умножения матриц



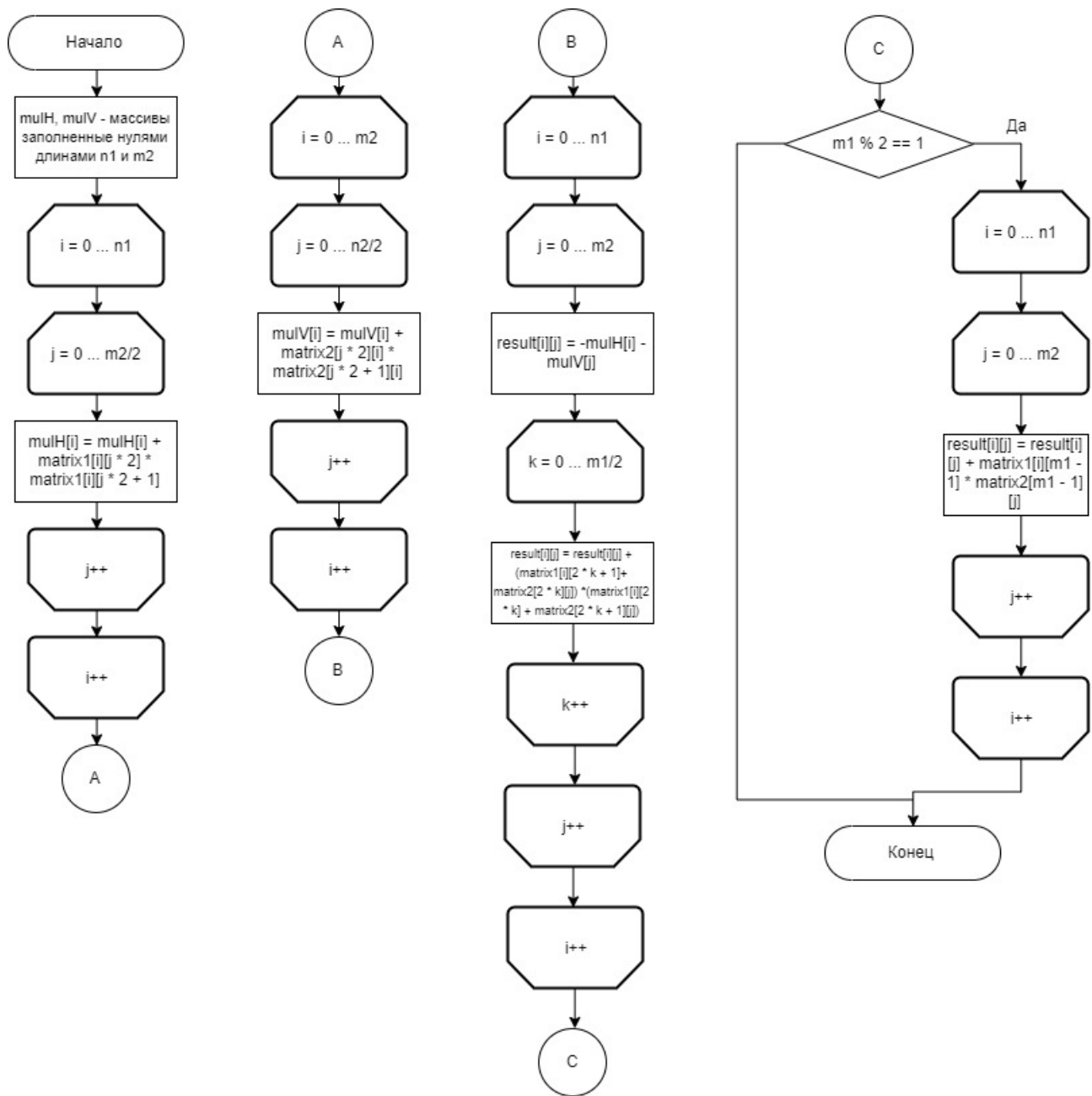


Рис. 2.2: Схема алгоритма Винограда

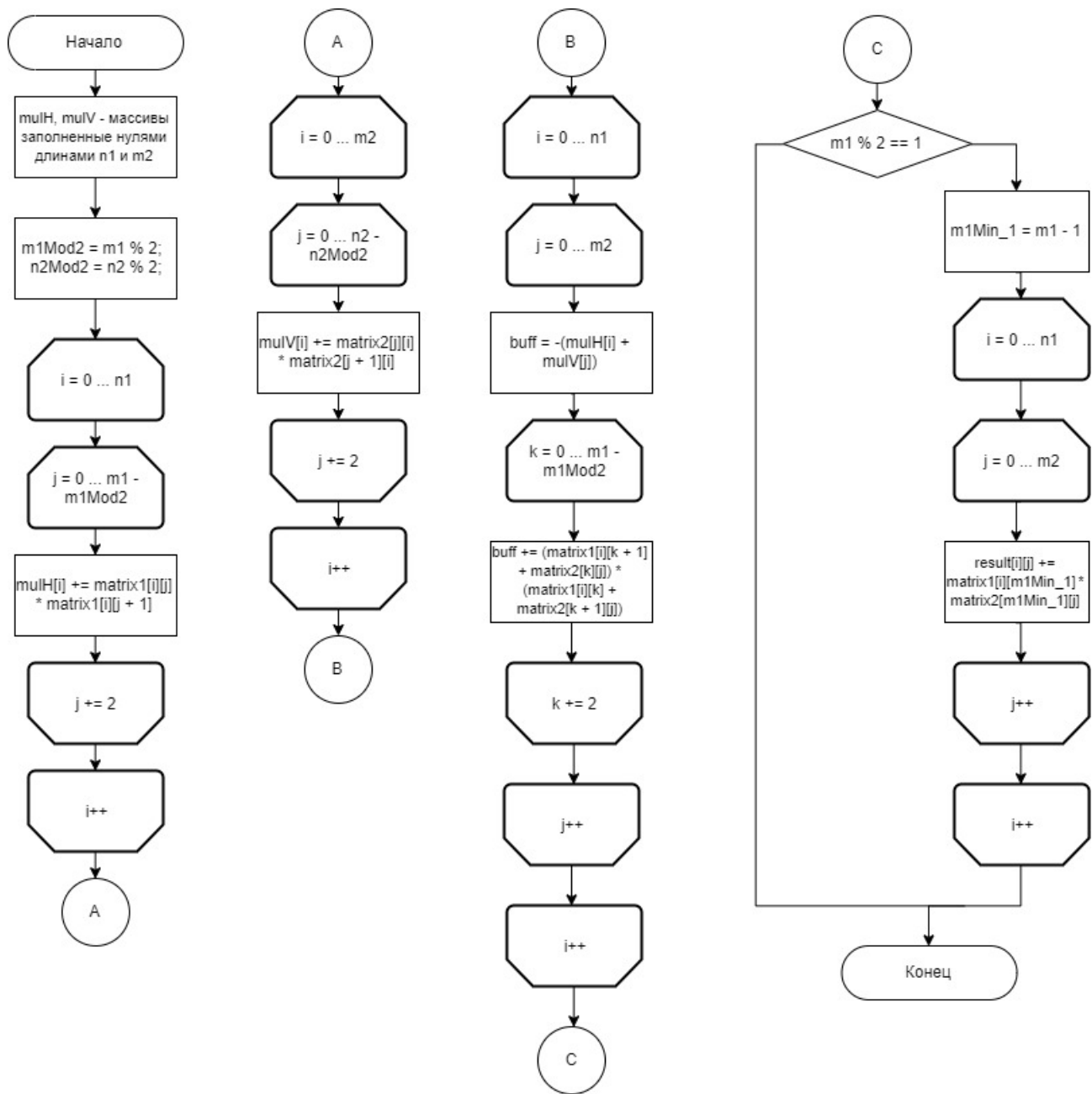


Рис. 2.3: Схема оптимизированного алгоритма Винограда

## 2.3 Трудоемкость алгоритмов

Введем модель трудоемкости для оценки алгоритмов:

- базовые операции стоимостью 1 —  $+$ ,  $-$ ,  $*$ ,  $/$ ,  $=$ ,  $==$ ,  $<=$ ,  $>=$ ,  $!=$ ,  $+=$ ,  $[]$ ;
- оценка трудоемкости цикла `for` от 0 до  $N$  с шагом 1  $F_{for} = 2 + N \cdot (2 + F_{body})$ , где  $F_{body}$  — тело цикла;
- стоимость условного перехода применим за 0, стоимость вычисления условия остаётся.

Оценим трудоемкость алгоритмов по коду программы.

### 2.3.1 Классический алгоритм

Рассмотрим трудоемкость классического алгоритма:

$$10MNQ + 4MQ + 4M + 2$$

### 2.3.2 Алгоритм Винограда

Рассмотрим трудоемкость алгоритма Винограда:

Трудоемкость алгоритма Винограда:

$$\text{Первый цикл: } 15/2 \cdot MN + 5 \cdot M + 2$$

$$\text{Второй цикл: } 15/2 \cdot MN + 5 \cdot M + 2$$

$$\text{Третий цикл: } 13 \cdot MNQ + 12 \cdot MQ + 4 \cdot M + 2$$

$$\text{Условный переход: } \left[ \begin{array}{ll} 2 & , \text{ в случае невыполнения условия} \\ 15 \cdot QM + 4 \cdot M + 2 & , \text{ в случае выполнения условия} \end{array} \right]$$

$$\text{Итого: } 15/2 \cdot MN + 5 \cdot M + 2 + 15/2 \cdot MN + 5 \cdot M + 2 + 13 \cdot MNQ + 12 \cdot MQ + 4 \cdot M + 2 + \left[ \begin{array}{ll} 2 & , \text{ в случае невыполнения условия} \\ 15 \cdot QM + 4 \cdot M + 2 & , \text{ в случае выполнения условия} \end{array} \right]$$

### 2.3.3 Оптимизированный алгоритм Винограда

Рассмотрим трудоемкость алгоритма Винограда:

Трудоемкость алгоритма Винограда:

Первый цикл:  $11/2 \cdot MN + 4 \cdot M + 2$

Второй цикл:  $11/2 \cdot MN + 4 \cdot M + 2$

Третий цикл:  $17/2 \cdot MNQ + 9 \cdot MQ + 4 \cdot M + 2$

Условный переход:  $\begin{bmatrix} 1 & , \text{ в случае невыполнения условия} \\ 10 \cdot QM + 4 \cdot M + 2 & , \text{ в случае выполнения условия} \end{bmatrix}$

Итого:  $11/2 \cdot MN + 4 \cdot M + 2 + 11/2 \cdot MN + 4 \cdot M + 2 + 15/2 \cdot MNQ + 9 \cdot MQ + 4 \cdot M + 2 + \begin{bmatrix} 1 & , \text{ в случае невыполнения условия} \\ 10 \cdot QM + 4 \cdot M + 2 & , \text{ в случае выполнения условия} \end{bmatrix}$

## 2.4 Вывод

В данном разделе были рассмотрены схемы алгоритмов умножения матриц, введена модель оценки трудоёмкости алгоритма, были рассчитаны трудоёмкости реализованных алгоритмов в соответствии с этой моделью.

## 3 | Технологическая часть

### 3.1 Выбор ЯП

Для реализации программ был выбран язык программирования Python, ввиду наличия опыта разработки на нём. Среда разработки - PyCharm.

Для замера процессорного времени используется функция, возвращающая количество наносекунд.

Листинг 3.1: Функция получения процессорного времени

```
1 def cpu_time(func, mt_1, mt_2):  
2     start = process_time_ns()  
3     func(mt_1, mt_2)  
4     end = process_time_ns()  
5     return end - start
```

### 3.2 Реализации алгоритмов

Листинг 3.2: Функция классического умножения матриц

```
1 def standard_alg(mt_1, mt_2):  
2     if len(mt_2) != len(mt_1[0]):  
3         print("Incorrect matrix sizes.")  
4         return  
5  
6     res = [[0 for _ in range(len(mt_2[0]))] for _ in range(  
7         len(mt_1))]  
7     for i in range(len(mt_1)):
```

```

8     for j in range(len(mt_2[0])):
9         for k in range(len(mt_1[0])):
10             res[i][j] += mt_1[i][k] * mt_2[k][j]
11     return res

```

Листинг 3.3: Алгоритм Винограда

```

1 def Winograd_alg(mt_1, mt_2):
2     n1 = len(mt_1)
3     n2 = len(mt_2)
4     m2 = len(mt_2[0])
5
6     if n2 != len(mt_1[0]):
7         print("Incorrect matrix sizes.")
8         return
9
10    mulH = [0 for _ in range(n1)]
11    mulV = [0 for _ in range(m2)]
12
13    for i in range(n1):
14        for j in range(n2 // 2):
15            mulH[i] += mt_1[i][2 * j] * mt_1[i][2 * j + 1]
16
17    for i in range(m2):
18        for j in range(n2 // 2):
19            mulV[i] += mt_2[2 * j][i] * mt_2[2 * j + 1][i]
20
21    res = [[0 for _ in range(m2)] for _ in range(n1)]
22    for i in range(n1):
23        for j in range(m2):
24            res[i][j] = - mulH[i] - mulV[j]
25            for k in range(n2 // 2):
26                res[i][j] += ((mt_1[i][2 * k] + mt_2[2 * k + 1][j])
27                             * (mt_1[i][2 * k + 1] + mt_2[2 * k][j]))
28
29    if n2 % 2:
30        for i in range(n1):
31            for j in range(m2):
32                res[i][j] += mt_1[i][n2 - 1] * mt_2[n2 - 1][j]
33
34    return res

```

---

### 3.2.1 Оптимизация алгоритма Винограда

В качестве оптимизации алгоритма Винограда можно выделить следующие пункты:

1. избавиться от деления в цикле;
2. замена  $mulH[i] = mulH[i] + \dots$  на  $mulH[i] += \dots$  (аналогично для  $mulV[i]$ );
3. накопление результата в буфер, а вне цикла сброс буфера в ячейку матрицы;

Листинг 3.4: Оптимизированный алгоритм Винограда

```
1 def Winograd_alg_improved(mt_1, mt_2):
2     n1 = len(mt_1)
3     n2 = len(mt_2)
4     m2 = len(mt_2[0])
5
6     if n2 != len(mt_1[0]):
7         print("Incorrect matrix sizes.")
8         return
9
10    d = n2 // 2
11
12    mulH = [0 for _ in range(n1)]
13    mulV = [0 for _ in range(m2)]
14
15    for i in range(n1):
16        mulH[i] = sum(mt_1[i][2 * j] * mt_1[i][2 * j + 1] for j
17                       in range(d))
18
19    for i in range(m2):
20        mulV[i] = sum(mt_2[2 * j][i] * mt_2[2 * j + 1][i] for j
21                       in range(d))
22
23    res = [[0 for _ in range(m2)] for _ in range(n1)]
```

```

22     for i in range(n1):
23         for j in range(m2):
24             res[i][j] = sum(
25 (mt_1[i][2 * k] + mt_2[2 * k + 1][j]) * (mt_1[i][2 * k +
26 1] + mt_2[2 * k][j]) for k in range(d)) \
27 - mulH[i] - mulV[j]
28     return res

```

### 3.3 Вывод

В данном разделе были приведены сведения о выборе языка программирования и приведены листинги кода реализованных алгоритмов.



## 4 | Исследовательская часть

### 4.1 Сравнительный анализ на основе замеров времени работы алгоритмов

Был проведен замер времени работы каждого из алгоритмов. Первый эксперимент производится для лучшего случая на матрицах с размерами от 100 x 100 до 500 x 500 с шагом 100.

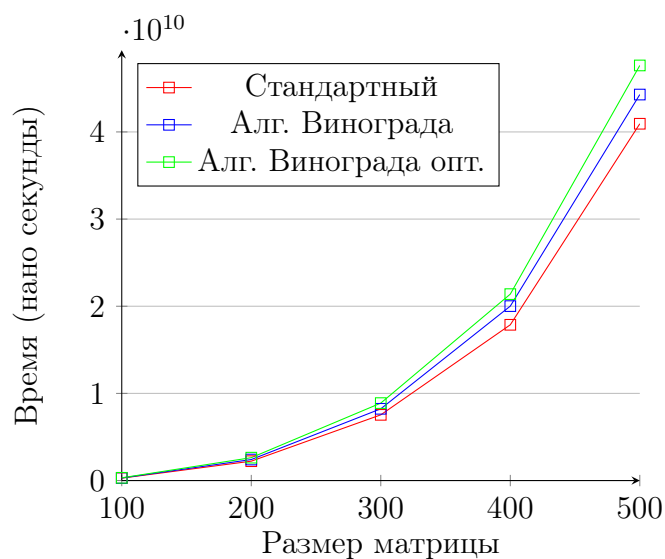


Рисунок 4.1. График времени работы алгоритмов на матрицах четной размерности

Второй эксперимент производится для худшего случая, когда поданы матрицы с нечетными размерами от 101 x 101 до 501 x 501 с шагом 100.

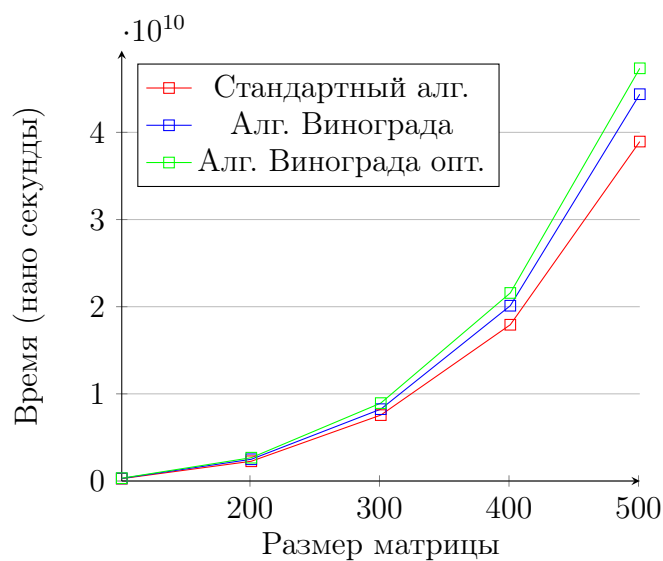


Рисунок 4.2. График времени работы алгоритмов на матрицах нечетной размерности

По результатам тестирования все рассматриваемые алгоритмы реализованы правильно. Самым медленным алгоритмом оказался алгоритм классического умножения матриц, а самым быстрым — оптимизированный алгоритм Винограда.

## 4.2 Вывод

В данном разделе были протестированы алгоритмы умножения матриц. Классический алгоритм показал худшие результаты, как и ожидалось. Оптимизированный алгоритм проявил себя лучше остальных.

# Заключение

В ходе лабораторной работы были изучены алгоритмы умножения матриц: стандартный и алгоритм Винограда, оптимизирован алгоритм Винограда, дана теоретическая оценка базового алгоритма умножения матриц, алгоритма Винограда и улучшенного алгоритма Винограда, реализованы три алгоритма умножения матриц.

## Список использованных источников

1. Алгоритм Кошперсмита - Винограда [Электронный ресурс]. – Режим доступа: <https://math.wikia.org/ru/wiki/Алгоритм-Кошперсмита-Винограда>. – Дата доступа: 16.10.2020.
2. Алгоритм Штрассена - Винограда [Электронный ресурс]. – Режим доступа: <http://wikiredia.ru/wiki/Алгоритм-Винограда-Штрассена>. – Дата доступа: 16.10.2020.
3. Умножение матриц. Эффективная реализация шаг за шагом [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/359272/>. – Дата доступа: 16.10.2020.