Цель работы — исследовать и сравнить эффективность различных алгоритмов умножения матриц, включая стандартный алгоритм и метод Штрассена. Для этого необходимо разработать реализации на языках C++ и Python, провести вычислительные эксперименты, оценить асимптотическую сложность алгоритмов и визуализировать результаты. Итогом является сравнительный анализ точности, производительности и масштабируемости рассматриваемых методов.

Задачи работы:

- Реализовать стандартный алгоритм умножения матриц на языке C++ и выполнить аналогичное вычисление на Python с использованием библиотеки NumPy.
- Провести замеры времени выполнения стандартного алгоритма для различных размеров матриц и оценить его вычислительную сложность, подкрепив выводы графической визуализацией.
- Реализовать алгоритм Штрассена на языке C++, обеспечить вывод всех промежуточных матриц
- Измерить время работы алгоритма Штрассена для разных размеров матриц и оценить его асимптотическую сложность с использованием графиков.
- Провести сравнительный анализ стандартного алгоритма, алгоритма Штрассена и реализации NumPy по производительности и масштабируемости.
- Сформулировать выводы о практической эффективности каждого метода и о влиянии асимптотической сложности на реальное время работы программ.

На рисунке 1 представлен график «Сравнение времени работы (log-log)» созданные в VS Code.

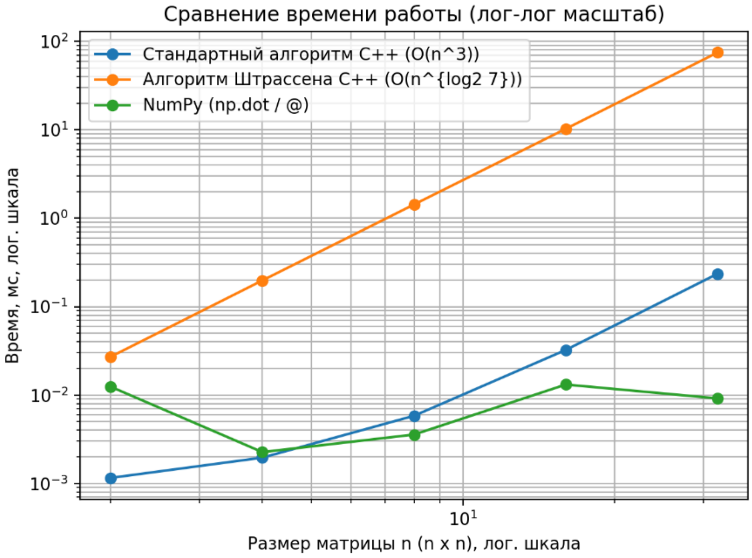


Рисунок 1 - Сравнение времени работы (log-log)

На рисунке 2 представлена сгенерированная визуализация асимптотической сложности.

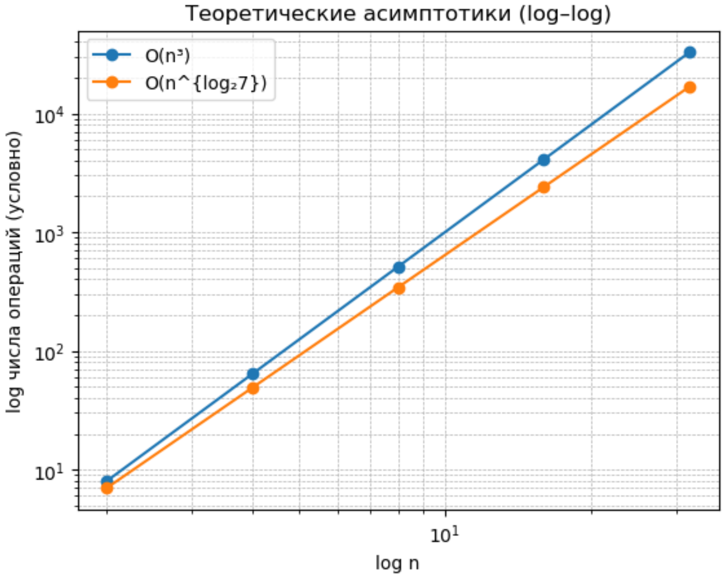


Рисунок 2 - Визуализация асимптотической сложности

Также с помощью Python были сгенерированы два графика показывающие отношение сложностей двух методов и график экономии операций алгоритма Штрассена на рисунках 3 и 4.

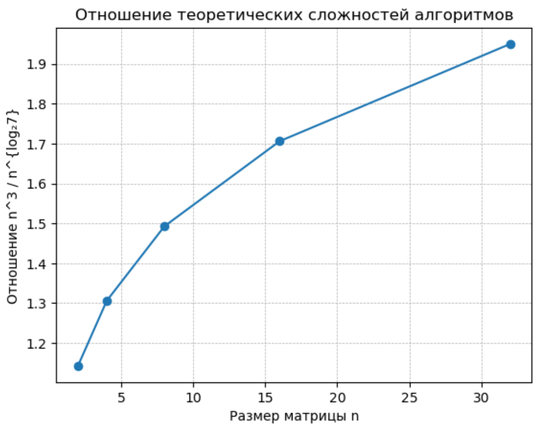


Рисунок 3- Отношение теоретических асимптотических сложностей двух алгоритмов

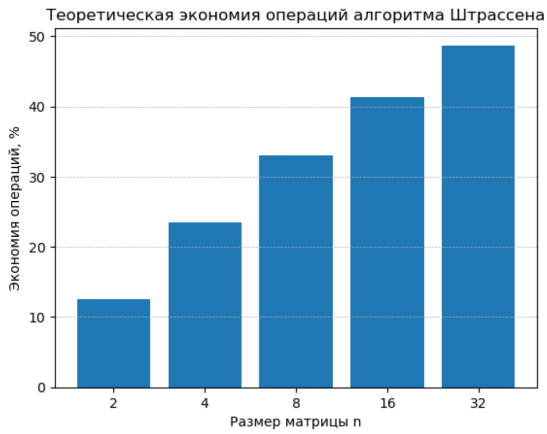


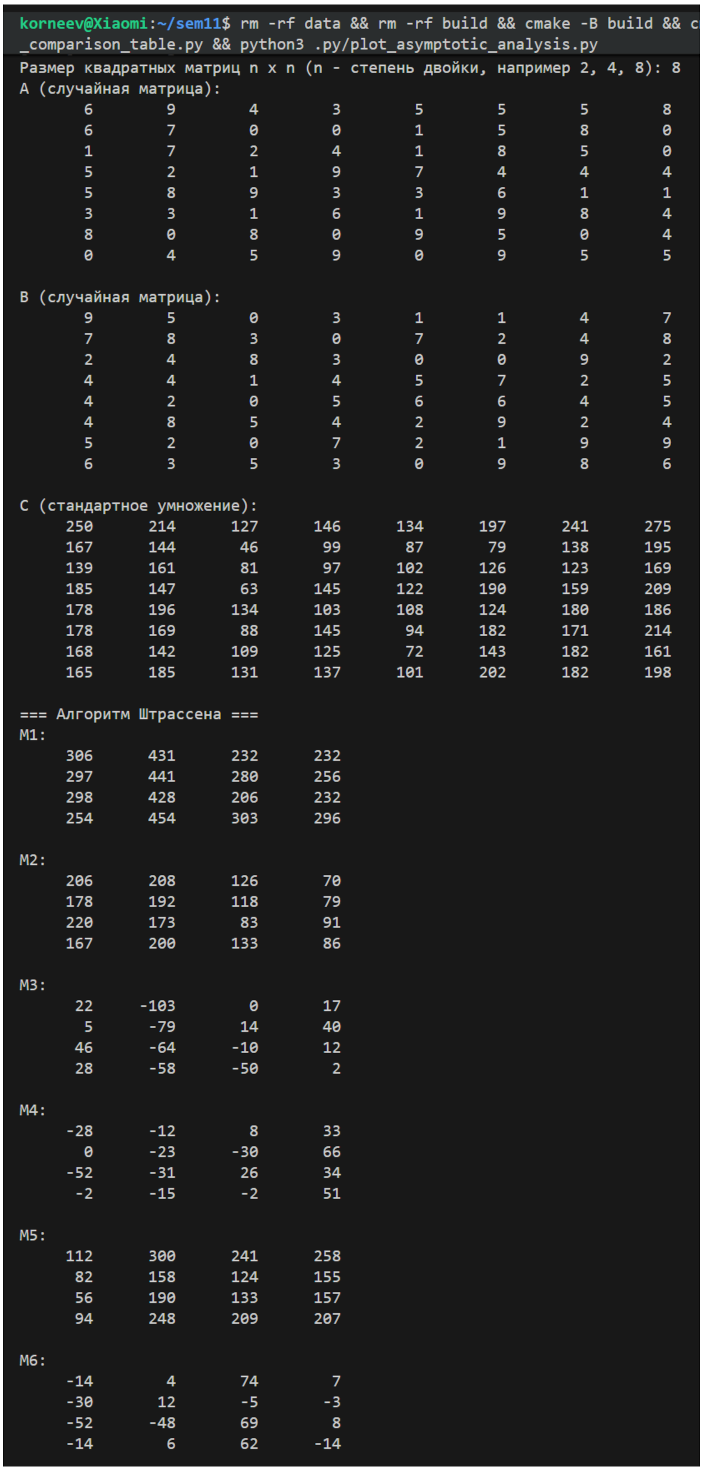
Рисунок 4 - Теоретическая экономия операций при использовании алгоритма Штрассена.

С помощью Python сгенерирована таблица, сравнивающая стандартный алгоритм и алгоритм Штрассена по сложности, эффективности и ресурсам. Из неё видно, что Штрассен теоретически быстрее, но на практике выигрывает только на достаточно больших матрицах.

Характеристика	Стандартный алгоритм	Алгоритм Штрассена
Вычислительная сложность	$O(n^3)$	$O(n^{\log_2 7}) \approx O(n^{2.807})$
Асимптотическое поведение	Кубический рост	Субкубический рост
Константный множитель	Малый (~1)	Большой (~7 рекурсивных вычислений)
Требования к памяти	$O(n^2)$	$O(n^2) + O(\log n)$\n(рекурсивный стек)
Простота реализации	Высокая	Средняя
Практическая эффективность	Лучше для малых матриц	Лучше для больших матриц
Параллелизация	Хорошая	Отличная (7 независимых подзадач)

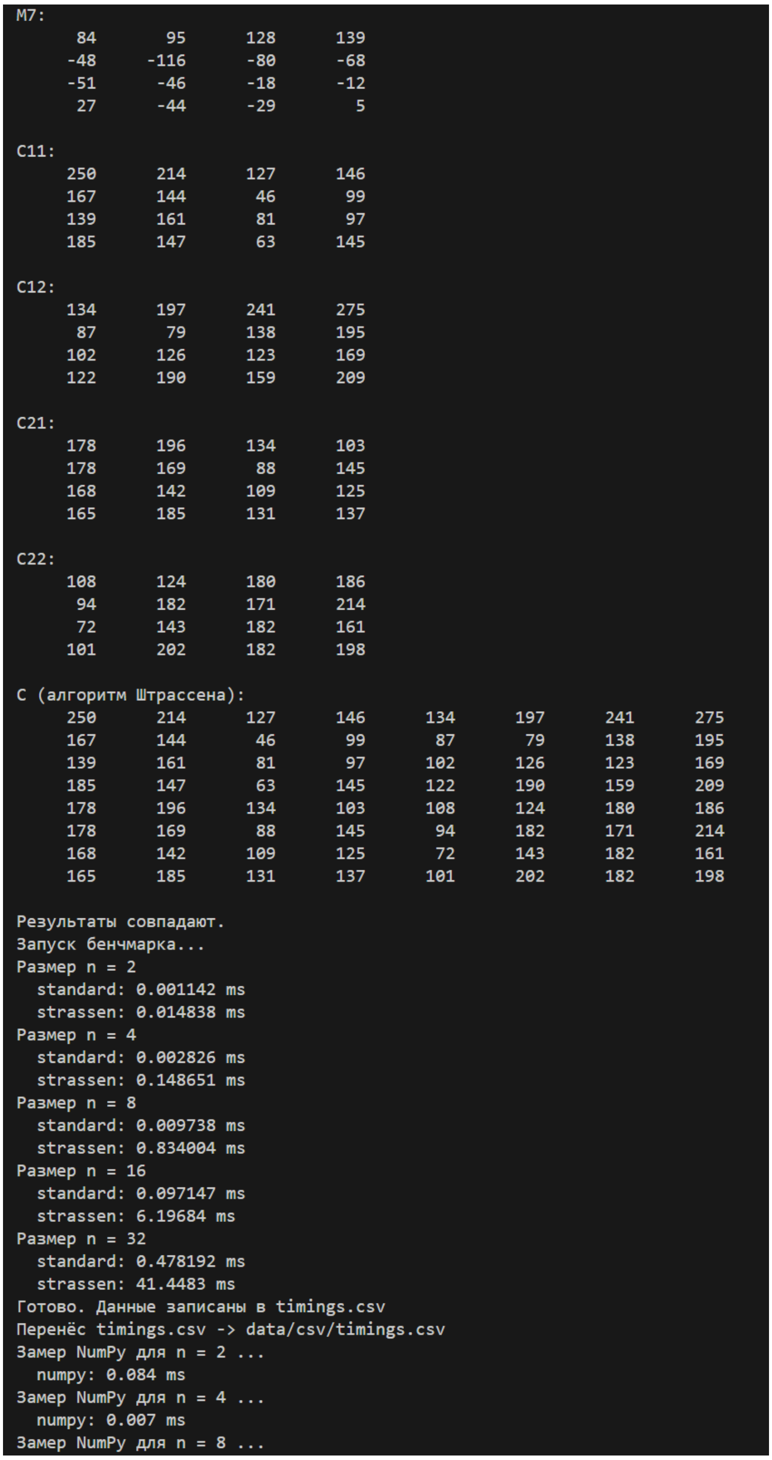
Рисунок 5 - Сравнительная таблица умножения матриц

На рисунках 6,7,8 представлен вывод программы в терминале VS Code.



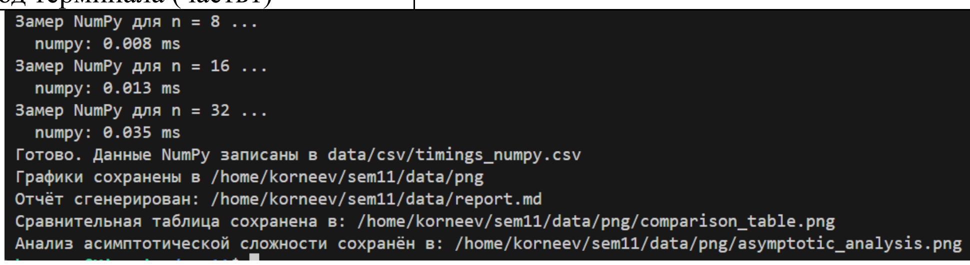
```
korneev@Xiaomi:~/sem11$ rm -rf data && rm -rf build && cmake -B build && c
_comparison_table.py && python3 .py/plot_asymptotic_analysis.py
Размер квадратных матриц n x n (n - степень двойки, например 2, 4, 8): 8
A (случайная матрица):
       6       9       4       3       5       5       5       8
       6       7       0       0       1       5       8       0
       1       7       2       4       1       8       5       0
       5       2       1       9       7       4       4       4
       5       8       9       3       3       6       1       1
       3       3       1       6       1       9       8       4
       8       0       8       0       9       5       0       4
       0       4       5       9       0       9       5       5

B (случайная матрица):
       9       5       0       3       1       1       4       7
       7       8       3       0       7       2       4       8
       2       4       8       3       0       0       9       2
       4       4       1       4       5       7       2       5
       4       2       0       5       6       6       4       5
       4       8       5       4       2       9       2       4
       5       2       0       7       2       1       9       9
       6       3       5       3       0       9       8       6

C (стандартное умножение):
     250     214     127     146     134     197     241     275
     167     144      46      99      87      79     138     195
     139     161      81      97     102     126     123     169
     185     147      63     145     122     190     159     209
     178     196     134     103     108     124     180     186
     178     169      88     145      94     182     171     214
     168     142     109     125      72     143     182     161
     165     185     131     137     101     202     182     198

=== Алгоритм Штрассена ===
M1:
     306     431     232     232
     297     441     280     256
     298     428     206     232
     254     454     303     296

M2:
     206     208     126      70
     178     192     118      79
     220     173      83      91
     167     200     133      86

M3:
      22    -103       0      17
       5     -79      14      40
      46     -64     -10      12
      28     -58     -50       2

M4:
     -28     -12       8      33
       0     -23     -30      66
     -52     -31      26      34
      -2     -15      -2      51

M5:
     112     300     241     258
      82     158     124     155
      56     190     133     157
      94     248     209     207

M6:
     -14       4      74       7
     -30      12      -5      -3
     -52     -48      69       8
     -14       6      62     -14
```

Рисунок 6 – вывод терминала (часть1)



```
M7:
      84      95     128     139
     -48    -116     -80     -68
     -51     -46     -18     -12
      27     -44     -29       5

C11:
     250     214     127     146
     167     144      46      99
     139     161      81      97
     185     147      63     145

C12:
     134     197     241     275
      87      79     138     195
     102     126     123     169
     122     190     159     209

C21:
     178     196     134     103
     178     169      88     145
     168     142     109     125
     165     185     131     137

C22:
     108     124     180     186
      94     182     171     214
      72     143     182     161
     101     202     182     198

C (алгоритм Штрассена):
     250     214     127     146     134     197     241     275
     167     144      46      99      87      79     138     195
     139     161      81      97     102     126     123     169
     185     147      63     145     122     190     159     209
     178     196     134     103     108     124     180     186
     178     169      88     145      94     182     171     214
     168     142     109     125      72     143     182     161
     165     185     131     137     101     202     182     198

Результаты совпадают.
Запуск бенчмарка...
Размер n = 2
  standard: 0.001142 ms
  strassen: 0.014838 ms
Размер n = 4
  standard: 0.002826 ms
  strassen: 0.148651 ms
Размер n = 8
  standard: 0.009738 ms
  strassen: 0.834004 ms
Размер n = 16
  standard: 0.097147 ms
  strassen: 6.19684 ms
Размер n = 32
  standard: 0.478192 ms
  strassen: 41.4483 ms
Готово. Данные записаны в timings.csv
Перенёс timings.csv -> data/csv/timings.csv
Замер NumPy для n = 2 ...
  numpy: 0.084 ms
Замер NumPy для n = 4 ...
  numpy: 0.007 ms
Замер NumPy для n = 8 ...
```

Рисунок 7 – вывод терминала (часть2)



```
Замер NumPy для n = 8 ...
  numpy: 0.008 ms
Замер NumPy для n = 16 ...
  numpy: 0.013 ms
Замер NumPy для n = 32 ...
  numpy: 0.035 ms
Готово. Данные NumPy записаны в data/csv/timings_numpy.csv
Графики сохранены в /home/korneev/sem11/data/png
Отчёт сгенерирован: /home/korneev/sem11/data/report.md
Сравнительная таблица сохранена в: /home/korneev/sem11/data/png/comparison_table.png
Анализ асимптотической сложности сохранён в: /home/korneev/sem11/data/png/asymptotic_analysis.png
```

Рисунок 8 – вывод терминала (часть3)

Анализ асимптотической сложности:

1. Асимптотическое поведение алгоритмов:

   • Стандартный алгоритм умножения матриц имеет сложность $O(n^3)$ (три вложенных цикла)

   • Алгоритм Штрассена имеет сложность $O(n^{(\log_2 7)}) \approx O(n^{2.807})$.

   • Отношение сложностей даётся выражением $n^3 / n^{(\log_2 7)} = n^{(3 - \log_2 7)}$, поэтому при $n \to \infty$ оно неограниченно растёт, что теоретически подтверждает асимптотическое преимущество алгоритма Штрассена.

2. Практическое поведение по данным бенчмарка:

- По результатам бенчмарка в исследованном диапазоне размеров матриц ($n \leq 32$) стандартный алгоритм работает быстрее или сравнимо с алгоритмом Штрассена. Асимптотическое превосходство алгоритма Штрассена не становится очевидным из-за значительных затрат, связанных с его рекурсивной реализацией.

Вывод: в ходе работы были изучены и реализованы три метода умножения матриц: классический алгоритм, метод Штрассена и использование библиотеки NumPy. Теоретическое рассмотрение показало, что алгоритм Штрассена имеет более низкую асимптотическую сложность по сравнению с классическим подходом. Однако на практике, при работе с небольшими матрицами, классический метод оказывается быстрее из-за меньших накладных расходов. Графические и сравнительные анализы демонстрируют, что преимущества алгоритма Штрассена становятся очевидными только при обработке матриц достаточно большого размера. В итоге выбор метода зависит от объема задачи: для небольших матриц предпочтительнее использовать классический алгоритм, а для очень больших — метод Штрассена.