

*Выполнение программной реализации представлена на рисунках 1-2*

```
[gregory@HP sem5-part2]$ rm -rf build && rm -r
[100%] Linking CXX executable app
[100%] Built target app
Matrix Multiplication Performance Comparison
=====
Matrix size: 80x80
=====
(i,j,k): 0.000417791s
(i,k,j): 7.2839e-05s
(k,i,j): 0.000122553s
(k,j,i): 0.000368347s
(j,i,k): 0.000409485s
(j,k,i): 0.000180685s
Best method: (i,k,j) (7.2839e-05s)

Matrix size: 400x400
=====
(i,j,k): 0.0730048s
(i,k,j): 0.0124216s
(k,i,j): 0.0211617s
(k,j,i): 0.103523s
(j,i,k): 0.0742348s
(j,k,i): 0.0531489s
Best method: (i,k,j) (0.0124216s)

Matrix size: 700x700
=====
(i,j,k): 0.405761s
(i,k,j): 0.0739806s
(k,i,j): 0.130117s
(k,j,i): 0.678417s
(j,i,k): 0.407885s
(j,k,i): 0.411429s
Best method: (i,k,j) (0.0739806s)

Matrix size: 1000x1000
=====
(i,j,k): 1.15124s
(i,k,j): 0.303677s
(k,i,j): 0.711906s
(k,j,i): 1.88084s
(j,i,k): 0.784924s
(j,k,i): 0.996683s
Best method: (i,k,j) (0.303677s)

Matrix size: 1500x1500
=====
(i,j,k): 4.63802s
(i,k,j): 0.928038s
(k,i,j): 2.47693s
(k,j,i): 7.49748s
(j,i,k): 2.70046s
(j,k,i): 4.21725s
Best method: (i,k,j) (0.928038s)

Результаты сохранены в файл: results.csv

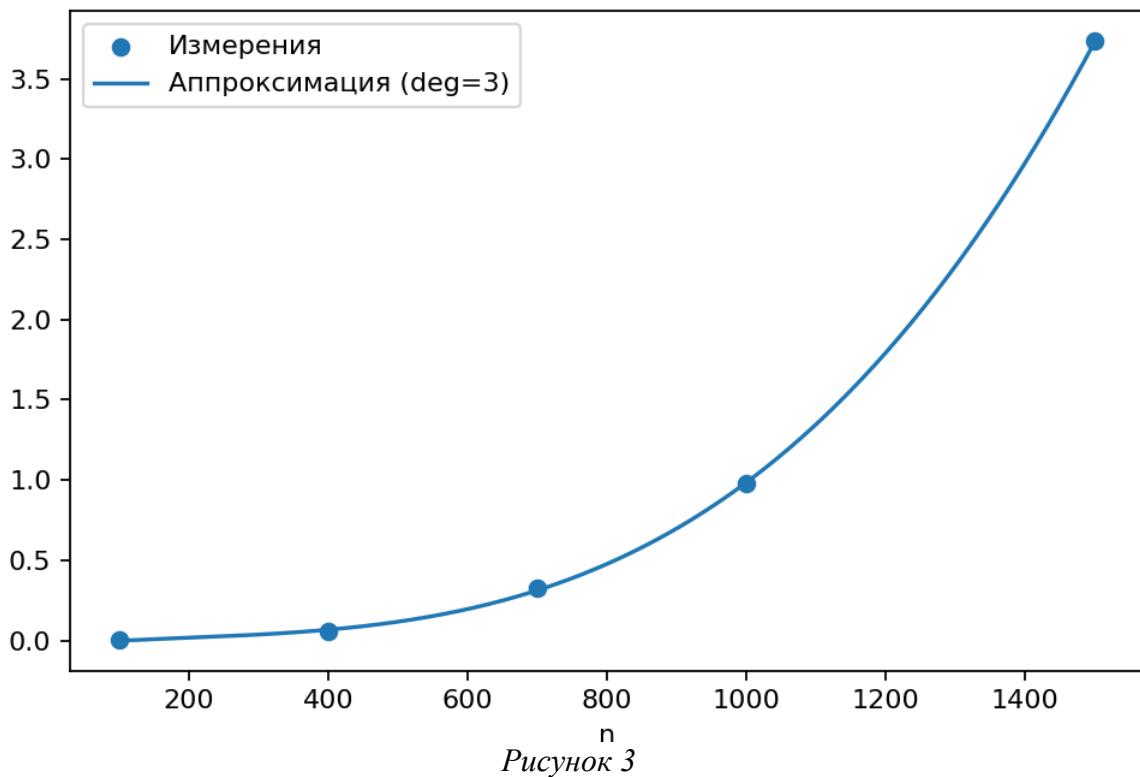
==== Polynomial trend (3rd degree) for each order ===
i,j,k trend: y = 2.172943e-09*n^3 + -1.626332e-06*n^2 + 6.742737e-04*n + -5.038278e-02 ; R^2 = 9.998464e-01
i,k,j trend: y = 1.165514e-10*n^3 + 3.710851e-07*n^2 + -2.110441e-04*n + 1.756346e-02 ; R^2 = 9.998709e-01
k,i,j trend: y = 6.063727e-10*n^3 + 4.472841e-07*n^2 + -4.070346e-04*n + 3.817305e-02 ; R^2 = 9.989596e-01
k,j,i trend: y = 3.393398e-09*n^3 + -2.381693e-06*n^2 + 9.836188e-04*n + -7.684056e-02 ; R^2 = 9.997756e-01
j,i,k trend: y = 1.081520e-09*n^3 + -7.895857e-07*n^2 + 5.837519e-04*n + -5.364756e-02 ; R^2 = 9.982493e-01
j,k,i trend: y = 2.168838e-09*n^3 + -1.928757e-06*n^2 + 8.685486e-04*n + -7.111395e-02 ; R^2 = 9.991640e-01

Общий тренд (среднее по всем порядкам):
avg(all orders) trend: y = 1.589937e-09*n^3 + -9.846664e-07*n^2 + 4.153524e-04*n + -3.270806e-02 ; R^2 = 9.998292e-01
```

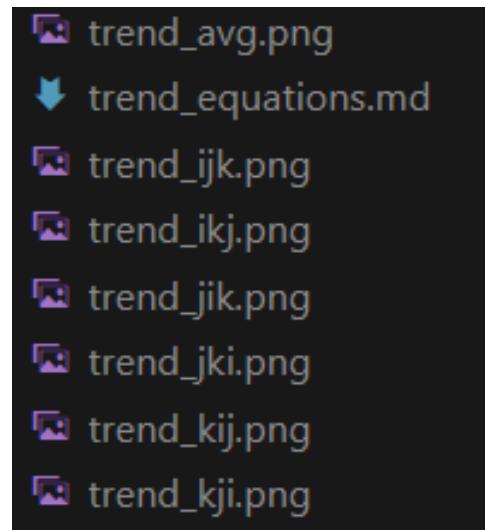
*Рисунки 1-2*

*По этим результатам построен график среднего тренда, представленный*

на рисунке 3.  
 $y = 1.545\text{e-}09 \cdot n^3 + -8.722\text{e-}07 \cdot n^2 + 3.409\text{e-}04 \cdot n + -2.906\text{e-}02$   
 $R^2 = 1.0000$



*Рисунок 3*



*Рисунок 4 – Список полученных графиков*

**Вывод:** В ходе выполнения семинарской работы была исследована зависимость времени выполнения алгоритма умножения матриц от порядка обхода индексов в циклах.

- Оптимальный порядок циклов: Эксперименты показали, что порядок следования циклов ( $i, j, k$ ) не всегда является

оптимальным. Другие варианты, такие как  $(i, k, j)$  или  $(k, i, j)$ , могут значительно повысить эффективность выполнения.

- Влияние на производительность: Различия в скорости работы связаны с принципом локальности данных и функционированием кэш-памяти процессора. Наиболее эффективным является порядок циклов, при котором доступ к элементам матриц осуществляется последовательно, максимально используя кэш-строки. Например, если внутренний цикл проходит по столбцам одной строки, это минимизирует промахи кэша и считается предпочтительным. В то же время, порядок, вызывающий скачкообразный доступ к данным (например, обход столбцов по строкам в языках, где матрицы хранятся построчно), приводит к значительным задержкам.