LAB 1-INTRODUCTION TO EFFICIENT LINEAR ALGEBRA USING C++ AND BLAS

BY

SACHIN SRINIVASA SHETTY

Contents

List of Figures

List of Tables

1 Introduction

This lab aims to analyze the different implementations of matrix multiplication. To study the computation time and GFLOPS for different sizes of the matrix for each Implementation. The six variants of basic dense matrix multiplication are analyzed and are compared with BLAS (Basic Linear Algebra Subroutine) and OpenBLAS.The implementations are carried out in C++ and results are shown in the table and are compared with graphs.

2 First Implementation Of The Matrix Multiplication

The basic dense matrix multiplication is implemented using 3 for loops. The first variant of the handmade loop is implemented as shown in the image below. The size of the matrix is changed and the computation time and GFLOPS are recorded.

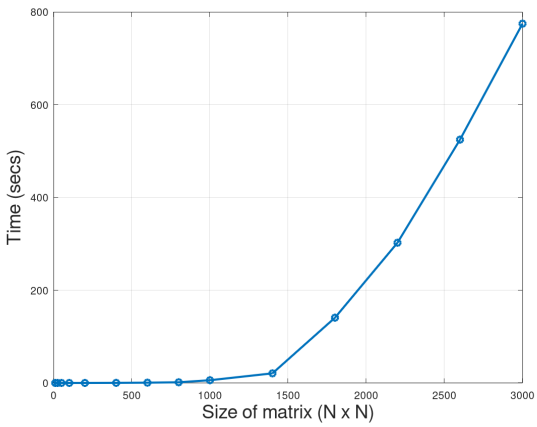
```
for(int i=0;i<M;i++){
  for(int j=0;j<N;j++){
    for(int k=0;k<K;k++){
      C(i,j)+=A(i,k)*B(k,j);
    }
  }
}
```

Figure 1: First variant of for loop implementation

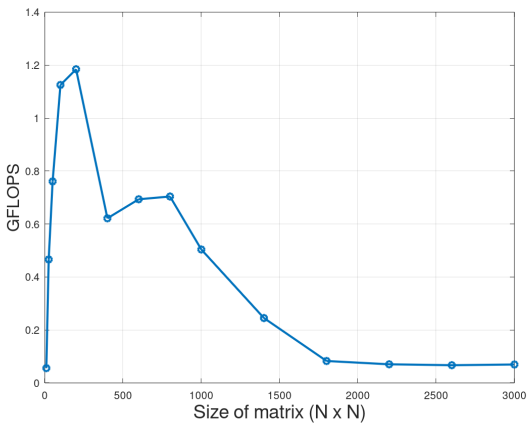
The recorded values are shown in the table and its plotted concerning computation time and GFLOPS. It can be inferred from the computations that the time taken increases as the size of the matrix increases.

Square matrix (N)	Computation time(secs)	GFLOPS
10	3.57E-05	0.0559816
25	6.70E-05	0.466188
50	0.000328562	0.760891
100	0.00170536	1.12508
200	0.0135108	1.18424
400	0.205927	0.62158
600	0.622929	0.693497
800	1.45476	0.703898
1000	5.89479	0.503835
1400	20.7264	0.245038
1800	140.79	0.0828471
2200	302.494	0.0704013
2600	524.947	0.0669629
3000	775.135	0.0696652

Table 1: Time (secs) and GFLOPS for matrix multiplication using First variant of handmade for-loops



(a) Time taken (secs)



(b) GFLOPS

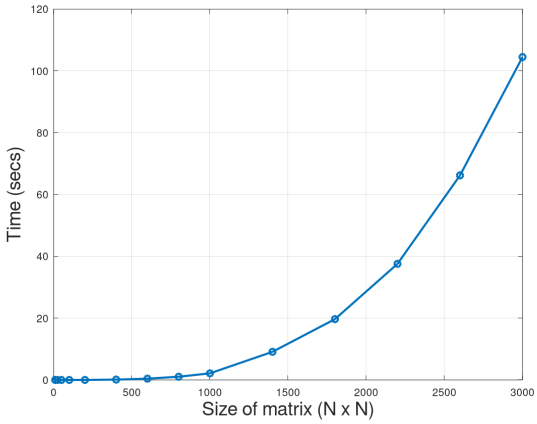
Figure 2: Plot for time taken and GFLOPS for matrix multiplication using First variant of handmade for-loops

3 Second Implementation Of The Matrix Multiplication

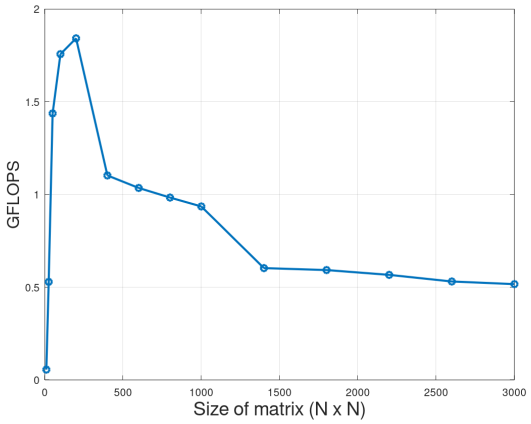
The order of the for loops is changed in the second implementation. The order of for-loops is jki. The implementation is then run for different matrix sizes. The recorded values are shown in the table and its plotted concerning computation time and GFLOPS. It can be observed that the time taken in this implementation is comparatively less than the first implementation. But the number of operations performed between the implementations are the same.

Square matrix (N)	**Computation time(secs)**	**GFLOPS**
10	3.58E-05	0.055855
25	5.91E-05	0.529087
50	0.000173909	1.43753
100	0.00113811	1.7573
200	0.00868427	1.84241
400	0.116054	1.10294
600	0.417175	1.03554
800	1.04095	0.983721
1000	2.13751	0.935666
1400	9.0985	0.603176
1800	19.6795	0.592699
2200	37.5741	0.566774
2600	66.1955	0.531033
3000	104.494	0.516774

Table 2: Time (secs) and GFLOPS for matrix multiplication using Second variant of handmade for-loops



(a) Time taken (secs)

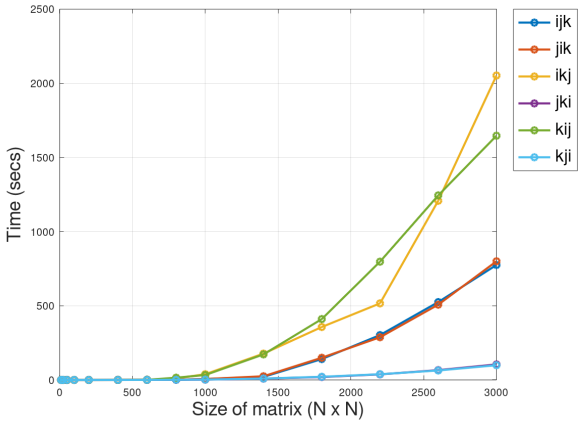


(b) GFLOPS

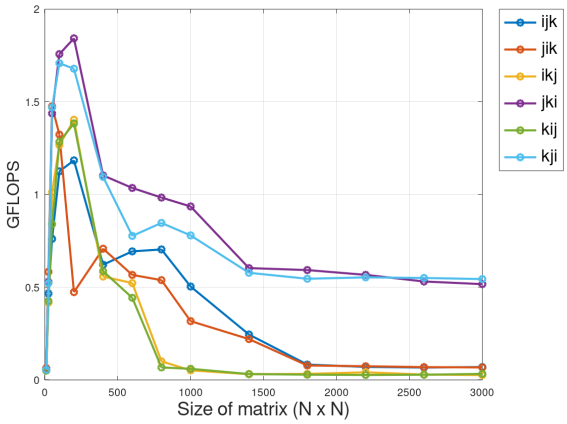
Figure 3: Plot for time taken and GFLOPS for matrix multiplication using Second variant of handmade for-loops

4 All Variants Of Implementation Of The Matrix Multiplication

The implementation of all possible orders of for loop is performed. The values for each variant are shown in the table and its plotted concerning time taken and GFLOPS. It can be inferred from the computations that the time taken for a particular matrix size is different for each variant. The time taken increases as the size of the matrix increases for all the variants. Major differences can be seen in the time taken between variants for the matrix of the same size. The difference in time between the variants is due to the change in the time taken to access and modify the data. It can be seen that loops jki and kji are much faster than the other loop implementations. The evolution of GFLOPS can be observed for each variant. It is seen that the GFLOPS rise to a certain peak and decrease as the size of the matrix increases. But the number of operations performed is the same for all the variants.



(a) Time taken (secs)



(b) GFLOPS

Figure 4: Plot for time taken and GFLOPS for matrix multiplication using various for-loop variants

Size (N)	ijk	jik	ikj	jki	kij	kji
10	3.57E-05	3.14E-05	3.70E-05	3.58E-05	3.98E-05	3.60E-05
25	6.70E-05	5.36E-05	7.49E-05	5.91E-05	7.35E-05	5.98E-05
50	0.000328562	0.000169372	0.000247135	0.000173909	0.000297581	0.000170165
100	0.00170536	0.00151144	0.00157924	0.00113811	0.00155712	0.00117075
200	0.0135108	0.0108574	0.0114021	0.00868427	0.0115649	0.00953247
400	0.205927	0.180817	0.229528	0.116054	0.218257	0.116888
600	0.622929	0.762578	0.826706	0.417175	0.975068	0.555999
800	1.45476	1.90202	10.2027	1.04095	15.1664	1.20916
1000	5.89479	6.306	38.6257	2.13751	33.3028	2.56546
1400	20.7264	24.8998	177.505	9.0985	172.281	9.49587
1800	140.79	149.609	356.912	19.6795	410.335	21.3707
2200	302.494	287.889	516.22	37.5741	797.298	38.4907
2600	524.947	507.672	1208.11	66.1955	1244.23	63.9184
3000	775.135	799.866	2052.52	104.494	1646.51	99.2392

Table 3: Time (secs) for different implementation of matrix multiplication of different matrix sizes (N)

Size (N)	ijk	jik	ikj	jki	kij	kji
10	0.0559816	0.0636193	0.0540979	0.055855	0.0503132	0.0555093
25	0.466188	0.583273	0.417468	0.529087	0.425326	0.522366
50	0.760891	1.47604	1.01159	1.43753	0.840107	1.46916
100	1.12508	1.32324	1.26643	1.7573	1.28442	1.70831
200	1.18424	1.47364	1.40325	1.84241	1.3835	1.67847
400	0.62158	0.7079	0.557667	1.10294	0.586465	1.09506
600	0.693497	0.566499	0.522556	1.03554	0.443046	0.77698
800	0.703898	0.538375	0.100366	0.983721	0.0675178	0.846869
1000	0.503835	0.317158	0.051779	0.935666	0.060055	0.779586
1400	0.245038	0.220403	0.0309174	0.603176	0.031855	0.577935
1800	0.0828471	0.0779632	0.0326803	0.592699	0.0284256	0.545794
2200	0.0704013	0.0739731	0.0412537	0.566774	0.0267102	0.553276
2600	0.0669629	0.0692415	0.0290966	0.531033	0.028252	0.549951
3000	0.0696652	0.0675113	0.0263091	0.516774	0.0327967	0.54414

Table 4: GFLOPS for different implementation of matrix multiplication of different matrix sizes (N)

5 Using BLAS

The matrix multiplication is then implemented using BLAS (Basic Linear Algebra Subroutine) library.BLAS is a set of fundamental operations on vectors and matrix which can be used to create optimized higher-level linear algebra functionality.Level-3 blas matrix-matrix operation dGEMM is used for the implementation. The dGEMM operation follows the below equation.

$$\mathbf{C} = \alpha op(A) op(B) + \beta C$$

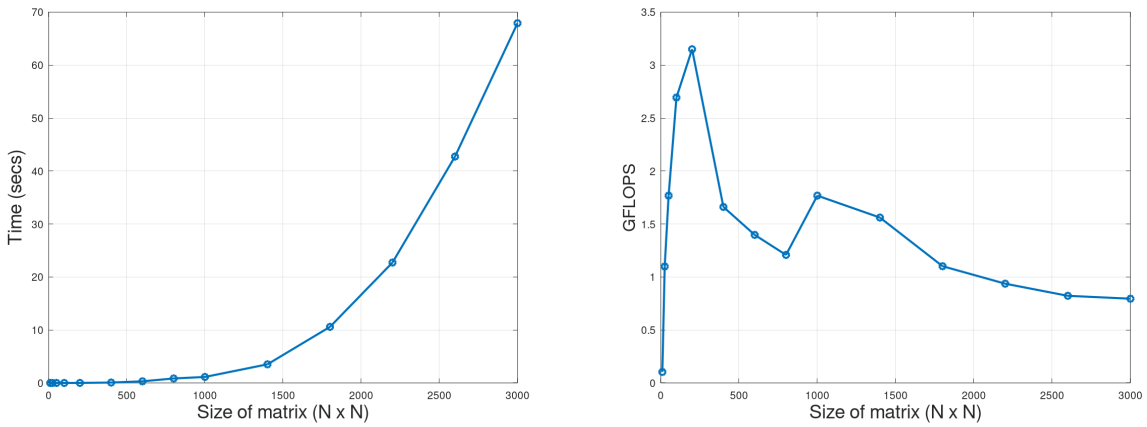
where,

- α and β are scalars
- A,B,C are matrices
- op(A) is a m by k matrix
- op(B) is a k by n matrix

To implement the matrix multiplication in BLAS we need to declare 13 arguments in dGEMM.The arguments are set and the computation time and GFLOPS for different matrix sizes are recorded. The computation time and GFLOPS are shown in the table. It can be inferred that the time taken increases as the size of the matrix increases. The best handmade for-loop (jki) is then compared with the BLAS concerning computation time and GLOPS using graphs. It can be seen that the time taken by BLAS is less than the best handmade for-loop. But the GFLOPS performed during the BLAS operation is more compared to the handmade for-loop implementation. This happens because the time taken to access and modify data in the memory using BLAS happens at a much faster rate when compared with the other implementation.

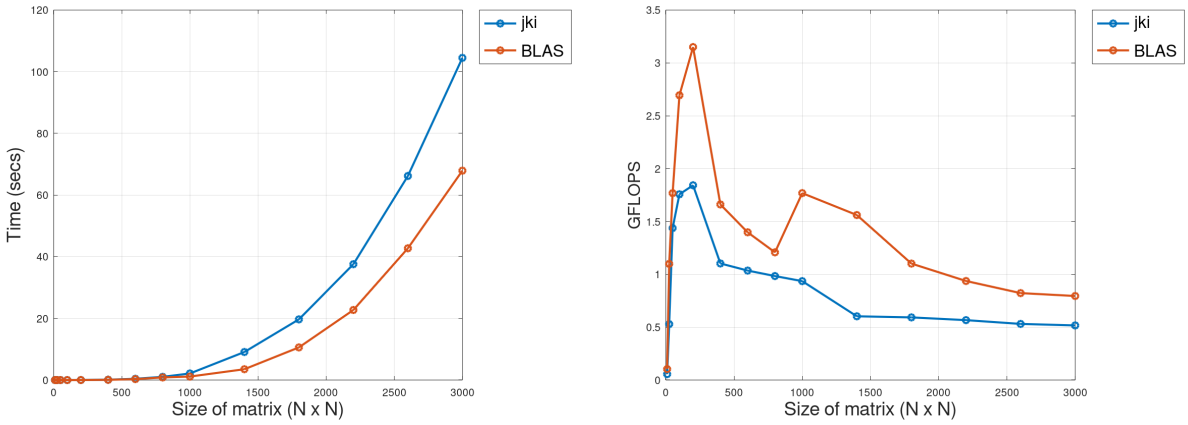
Square Matrix (N)	**Time (secs)**	**GFLOPS**
10	1.93E-05	0.103799
25	2.84E-05	1.09931
50	0.000141389	1.76817
100	0.000742419	2.6939
200	0.00614612	3.14975
400	0.0770535	1.66118
600	0.309078	1.3977
800	0.847502	1.20826
1000	1.13109	1.7682
1400	3.51594	1.56089
1800	10.5871	1.10172
2200	22.7283	0.936983
2600	42.7367	0.822524
3000	67.91	0.79517

Table 5: Time (secs) and GFLOPS using BLAS for matrix multiplication of different matrix size (N)



(a) Time taken (secs) (b) GFLOPS

Figure 5: Plots for time taken and GFLOPS for matrix multiplication using BLAS



(a) Time taken (secs) (b) GFLOPS

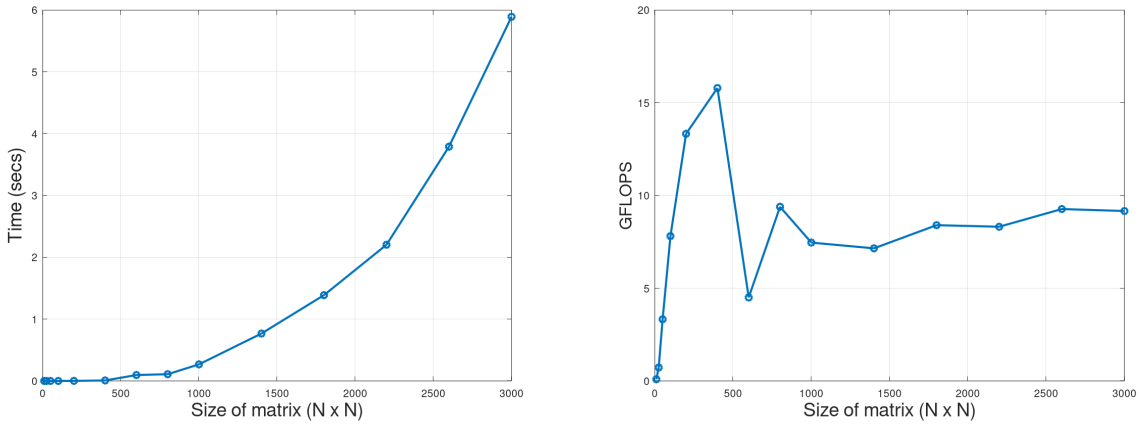
Figure 6: Plots for comparison of best handmade for-loop and BLAS

6 Using OpenBLAS

In this section, we use Openblas which is an optimized BLAS library is used to implement the matrix multiplication. The computation time and GFLOPS are recorded for different matrix sizes and are shown in the table. The plot below shows how the computation time and GFLOPS change with different matrix sizes. The best handmade for-loop(jki), BLAS, and OpenBLAS implementation are compared with each other using plots. It can be seen that the time taken by the OpenBLAS is much less than the other two implementations and also the GFLOPS of the OpenBLAS is higher than the other two implementations. So, OpenBLAS is a very fast implementation of matrix multiplication as it takes comparatively very less computational time for matrix size of higher-order.

Square Matrix (N)	**Time (secs)**	**GFLOPS**
10	2.22E-05	0.089948
25	4.31E-05	0.725647
50	7.52E-05	3.32646
100	0.00025604	7.81128
200	0.00120035	13.3295
400	0.00810244	15.7977
600	0.0959233	4.5036
800	0.109016	9.39311
1000	0.268011	7.46237
1400	0.76731	7.15226
1800	1.38834	8.40138
2200	2.20339	8.3167
2600	3.79155	9.27115
3000	5.89208	9.16485

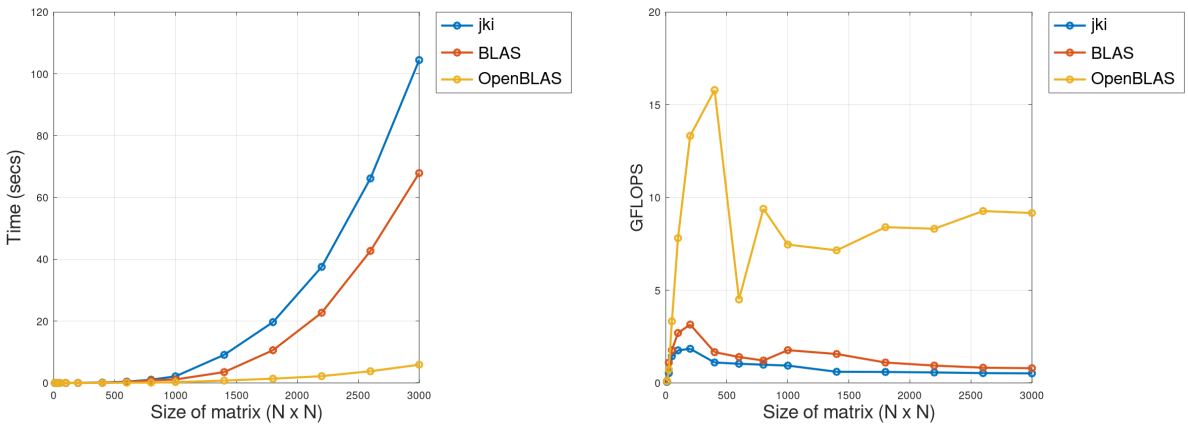
Table 6: Time (secs) and GFLOPS using OpenBLAS for matrix multiplication of different size (N)



(a) Time taken (secs)

(b) GFLOPS

Figure 7: Plots for time taken and GFLOPS for matrix multiplication using OpenBLAS



(a) Time taken (secs)

(b) GFLOPS

Figure 8: Plots for comparison of best handmade for-loop,BLAS and OpenBLAS

7 Conclusion

Based on the observations made on the data obtained for various implementations, If we consider a square matrix of 1400 x1400, it can be seen the best for-loop implementation takes a computation time of about 9.0985 secs. Whereas, the BLAS implementation takes a computation time of 3.5194 secs, which is comparatively less time when compared. But the OpenBLAS implementation takes a computation time of 0.76731 secs, which is a significant decrease in time compared to the other two. So, it can be concluded that the OpenBLAS implementation which is an optimized BLAS library is the best way to compute matrix operations of higher order.