university of groningen

UNIVERSITAT DE BARCELONA

VNIVERSITAT DE VALÈNCIA

Theoretical Chemistry and Computational Modeling

Advanced Computational Techniques

Students: Andrea Carolina Chimarro Contreras, Samantha Lucero Martínez Pineda, Pedro Martínez Zaragoza

Emails: andrea.chimarro@estudiante.uam.es, s.l.martinez.pineda@student.rug.nl, pedro.martinez@estudiante.uam.es

The objective of this task is to compare the computational efficiency of three multiplication methods:

- Multiplication using sparse matrix format
- Multiplication using dense matrix format
- Multiplication using DGEMM routine

These methods were applied to 61 multiplications between five 25×25 matrices and six 125×125 matrices, each with different filling degrees (the proportion of non-zero entries). Three aspects of computational efficiency were analyzed, which are described in the following sections.

1 Number of multiplications

Sparce matrices save only the non-zero elements, resulting in a lower number of multiplications compared to the theoretical maximum. For an $N \times N$ matrix, the theoretical maximum number of multiplications is N^3, assuming all entries are non-zero. Thus, for 25×25 matrices, the maximum is 15625, and for 125×125 matrices, it is 1953125. The actual number of multiplications performed for the different matrix formats were recorded and are shown in Table 1 and 2. As expected, the number of multiplication decreases with lower filling degrees and increases as the matrices approach a dense structure.

2 Profiling

Profiling was performed with gprof by executing the program with 25×25 and 125×125 matrices at both low and high filling degrees. The filling degree refers to the proportion of non-zero elements in the matrix, with "low" filling degree indicating sparse matrices and "high" filling degree indicating matrices that are closer to dense.

a)

Each sample counts as 0.01 seconds.

% time	cumulative seconds	self seconds	calls	self ms/call	total ms/call	name
100.00	0.02	0.02	100	0.20	0.20	multiply_dense.0
0.00	0.02	0.00	100	0.00	0.00	multiply_sparse.2
0.00	0.02	0.00	2	0.00	0.00	dense_matrix.1
0.00	0.02	0.00	2	0.00	0.00	get_dimension.4
0.00	0.02	0.00	2	0.00	0.00	read_matrix.3
0.00	0.02	0.00	1	0.00	20.00	MAIN__

b)

Each sample counts as 0.01 seconds.

% time	cumulative seconds	self seconds	calls	self ms/call	total ms/call	name
100.00	0.03	0.03	100	0.30	0.30	multiply_sparse.2
0.00	0.03	0.00	100	0.00	0.00	multiply_dense.0
0.00	0.03	0.00	2	0.00	0.00	dense_matrix.1
0.00	0.03	0.00	2	0.00	0.00	get_dimension.4
0.00	0.03	0.00	2	0.00	0.00	read_matrix.3
0.00	0.03	0.00	1	0.00	30.00	MAIN__

Figure 1: Results of the profiling analysis of the multiplication of the 25×25 matrices with the a) lowest and b) highest filling degree.

For the 25×25 matrices, each multiplication was repeated 100 times to obtain a measurable running time. For the matrix with the lowest filling degree, 100% of the total running time was spent on the dense multiplication method (Figure 1.a), while for the matrix with the highest filling degree, 100% of the running time was spent on the sparse multiplication method (Figure 1.a).

a)
```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 50.00      0.01     0.01        2     5.00     5.00  dense_matrix.1
 50.00      0.02     0.01        1    10.00    10.00  multiply_dense.0
  0.00      0.02     0.00        2     0.00     0.00  get_dimension.4
  0.00      0.02     0.00        2     0.00     0.00  read_matrix.3
  0.00      0.02     0.00        1     0.00    20.00  MAIN__
  0.00      0.02     0.00        1     0.00     0.00  multiply_sparse.2
```

b)
```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 76.81      1.06     1.06        1     1.06     1.06  multiply_sparse.2
 22.46      1.37     0.31        2     0.15     0.15  dense_matrix.1
  0.72      1.38     0.01        1     0.01     0.01  multiply_dense.0
  0.00      1.38     0.00        2     0.00     0.00  get_dimension.4
  0.00      1.38     0.00        2     0.00     0.00  read_matrix.3
  0.00      1.38     0.00        1     0.00     1.38  MAIN__
```

Figure 2: Results of the profiling analysis of the multiplication of the 125×125 matrices with the a) lowest and b) highest filling degree.

For the 125×125 matrix with the lowest filling degree (i.e., the sparse matrix), the results show that approximately 50% of the total running time was used by the dense multiplication method, while the other 50% was used by the sparse multiplication method (Figure 2.a). In contrast, for the 125×125 matrix with the highest filling degree (i.e., a dense matrix), 76.81% of the total running time was used by the sparse multiplication method, and only 0.72% was used by the dense multiplication (Figure 2.b).
These profiling results reveal a clear trend: sparse multiplication is more computationally efficient in matrices with low filling degrees (especially for smaller matrix sizes) and less efficient for matrices with high filling degrees. This trend is more pronounced for smaller matrices, where the overhead of handling sparse matrix formats can be more significant compared to the density of the matrix.

3 Timing

The execution time for each of the three multiplication methods (sparse, dense, and DGEMM routine) was recorded and plotted against the filling degree for both matrix sizes. The results are shown in Figures 3.a and 3.b, respectively.
Figure 3.a corresponds to the 25×25 matrices. Here, we observe that the execution time for sparse multiplication in low filling degree matrices is significantly shorter than the other two methods. However, as the filling degree increases, the execution time of sparse multiplication increases exponentially, a trend also observed for the 125×125 matrices. Notably, the execution times of both the dense multiplication and DGEMM routine remain relatively constant across all filling degrees. Moreover, at high filling degrees, dense multiplication becomes even more efficient than the DGEMM routine.
Figure 3.b corresponds to the 125×125 matrices. For matrices with low filling degrees, the execution times for all three methods are very similar. However, in most cases, the sparse multiplication and DGEMM routine take less time than the dense multiplication. Interestingly, in some cases, sparse multiplication is even faster than the DGEMM routine. As the filling degree increases, the execution time for sparse multiplication grows exponentially, while the execution time for dense multiplication decreases, becoming significantly shorter than the other methods. The BLAS routine (DGEMM) shows a more constant execution time across varying filling degrees.
In conclusion, sparse matrix multiplication is more efficient for low filling degree matrices, with this effect being more pronounced in smaller matrices. However, its efficiency decreases

exponentially as the filling degree increases, especially for larger matrices. Dense multiplication and the DGEMM routine, in contrast, maintain more stable execution times and become more efficient for matrices with higher filling degrees.

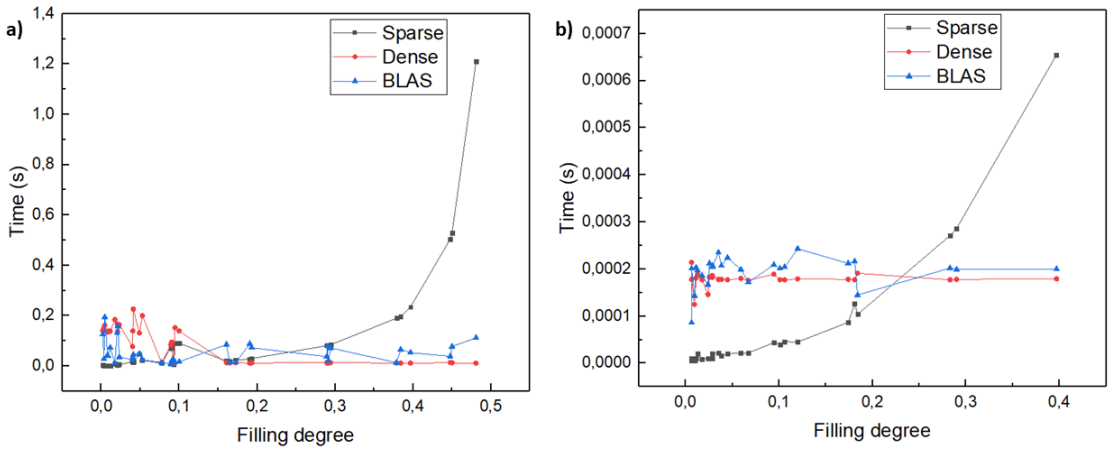

Figure 3: Plots of filling degree vs execution time of the 3 multiplication types with several a) 25x25 and b)125x125 matrices.

Table 1: Performance comparison of matrix operations

Matrix	**Filling Degree**	**Time Sparse (s)**	**Time Dense (s)**	**Time BLAS (s)**	**Sparse:N^3(15625)**
$25_1 \times 25_1$	6.40×10^{-3}	5×10^{-6}	2.14×10^{-4}	8.60×10^{-5}	4
$25_1 \times 25_5$	6.40×10^{-3}	1×10^{-5}	1.78×10^{-4}	2.01×10^{-4}	4
$25_1 \times 25_{10}$	9.60×10^{-3}	6×10^{-6}	1.25×10^{-4}	1.43×10^{-4}	6
$25_5 \times 25_1$	1.12×10^{-2}	1×10^{-5}	1.81×10^{-4}	2.03×10^{-4}	7
$25_{10} \times 25_1$	1.28×10^{-2}	2×10^{-5}	1.88×10^{-4}	1.97×10^{-4}	8
$25_5 \times 25_5$	1.76×10^{-2}	9×10^{-6}	1.77×10^{-4}	1.86×10^{-4}	11
$25_{10} \times 25_5$	2.40×10^{-2}	1×10^{-5}	1.46×10^{-4}	1.67×10^{-4}	15
$25_1 \times 25_{25}$	2.56×10^{-2}	1×10^{-5}	1.83×10^{-4}	2.12×10^{-4}	16
$25_{25} \times 25_1$	2.88×10^{-2}	1×10^{-5}	1.82×10^{-4}	2.07×10^{-4}	19
$25_5 \times 25_{10}$	2.88×10^{-2}	2.1×10^{-5}	1.86×10^{-4}	2.05×10^{-4}	18
$25_1 \times 25_{50}$	3.52×10^{-2}	2.2×10^{-5}	1.78×10^{-4}	2.35×10^{-4}	23
$25_{10} \times 25_{10}$	3.84×10^{-2}	1.6×10^{-5}	1.78×10^{-4}	2.07×10^{-4}	24
$25_{50} \times 25_1$	4.48×10^{-2}	2×10^{-5}	1.77×10^{-4}	2.24×10^{-4}	29
$25_{25} \times 25_5$	5.92×10^{-2}	2.2×10^{-5}	1.80×10^{-4}	1.99×10^{-4}	41
$25_5 \times 25_{25}$	6.72×10^{-2}	2.2×10^{-5}	1.76×10^{-4}	1.72×10^{-4}	46
$25_{10} \times 25_{25}$	9.44×10^{-2}	4.4×10^{-5}	1.89×10^{-4}	2.09×10^{-4}	69
$25_{25} \times 25_{10}$	1.01×10^{-1}	4×10^{-5}	1.77×10^{-4}	2.01×10^{-4}	70
$25_{50} \times 25_5$	1.06×10^{-1}	4.6×10^{-5}	1.77×10^{-4}	2.04×10^{-4}	73
$25_5 \times 25_{50}$	1.20×10^{-1}	4.5×10^{-5}	1.79×10^{-4}	2.43×10^{-4}	84
$25_{10} \times 25_{50}$	1.74×10^{-1}	8.7×10^{-5}	1.78×10^{-4}	2.12×10^{-4}	138
$25_{25} \times 25_{25}$	1.81×10^{-1}	1.26×10^{-4}	1.77×10^{-4}	2.16×10^{-4}	159
$25_{50} \times 25_{10}$	1.84×10^{-1}	1.04×10^{-4}	1.91×10^{-4}	1.45×10^{-4}	131
$25_{50} \times 25_{25}$	2.83×10^{-1}	2.71×10^{-4}	1.77×10^{-4}	2.02×10^{-4}	305
$25_{25} \times 25_{50}$	2.90×10^{-1}	2.86×10^{-4}	1.78×10^{-4}	1.99×10^{-4}	317
$25_{50} \times 25_{50}$	3.97×10^{-1}	6.55×10^{-4}	1.79×10^{-4}	2.00×10^{-4}	598

Table 2: Performance comparison of matrix operations

Matrix	**Filling Degree**	**Time Sparse (s)**	**Time Dense (s)**	**Time BLAS (s)**	**Sparse:N^3(1953125)**
$125_1 \times 125_1$	2.05×10^{-3}	3.40×10^{-3}	0.140 871	0.126 269	32
$125_1 \times 125_2$	3.52×10^{-3}	1.43×10^{-3}	0.152 358	2.76×10^{-2}	55
$125_2 \times 125_1$	4.54×10^{-3}	7.20×10^{-4}	0.160 825	0.194 15	71
$125_1 \times 125_5$	8.13×10^{-3}	1.73×10^{-3}	0.134 334	4.31×10^{-2}	130
$125_2 \times 125_2$	8.77×10^{-3}	1.24×10^{-3}	0.139 062	3.98×10^{-2}	137
$125_5 \times 125_1$	1.16×10^{-2}	1.82×10^{-3}	1.38×10^{-1}	7.26×10^{-2}	185
$125_1 \times 125_{10}$	1.75×10^{-2}	8.11×10^{-3}	0.184 709	1.17×10^{-2}	285
$125_2 \times 125_5$	2.03×10^{-2}	5.88×10^{-3}	0.166 352	0.130 937	325
$125_{10} \times 125_1$	2.11×10^{-2}	4.93×10^{-3}	1.37×10^{-1}	1.58×10^{-1}	340
$125_5 \times 125_2$	2.32×10^{-2}	5.35×10^{-3}	1.64×10^{-1}	3.56×10^{-2}	370
$125_1 \times 125_{25}$	4.02×10^{-2}	1.68×10^{-2}	7.71×10^{-2}	2.51×10^{-2}	684
$125_2 \times 125_{10}$	4.06×10^{-2}	1.39×10^{-2}	0.139 23	3.38×10^{-2}	672
$125_{10} \times 125_2$	4.13×10^{-2}	1.66×10^{-2}	2.26×10^{-1}	4.49×10^{-2}	684
$125_{25} \times 125_1$	4.90×10^{-2}	4.67×10^{-2}	1.31×10^{-1}	4.89×10^{-2}	830
$125_5 \times 125_5$	5.27×10^{-2}	2.37×10^{-2}	2.00×10^{-1}	2.49×10^{-2}	888
$125_1 \times 125_{50}$	7.75×10^{-2}	1.06×10^{-2}	1.27×10^{-2}	1.35×10^{-2}	1429
$125_2 \times 125_{25}$	8.93×10^{-2}	7.19×10^{-2}	8.49×10^{-2}	7.31×10^{-3}	1610
$125_{25} \times 125_2$	9.03×10^{-2}	7.46×10^{-2}	9.50×10^{-2}	1.98×10^{-2}	1626
$125_{50} \times 125_1$	9.31×10^{-2}	6.21×10^{-3}	1.12×10^{-2}	2.34×10^{-2}	1683
$125_{10} \times 125_5$	9.48×10^{-2}	9.14×10^{-2}	1.52×10^{-1}	1.38×10^{-2}	1718
$125_5 \times 125_{10}$	0.100 032	9.00×10^{-2}	1.39×10^{-1}	1.66×10^{-2}	1829
$125_2 \times 125_{50}$	0.160 640	1.96×10^{-2}	1.28×10^{-2}	8.52×10^{-2}	3331
$125_{50} \times 125_2$	0.164 736	1.89×10^{-2}	1.22×10^{-2}	1.57×10^{-2}	3311
$125_{10} \times 125_{10}$	0.172 544	2.40×10^{-2}	1.31×10^{-2}	1.40×10^{-2}	3669
$125_{25} \times 125_5$	0.190 656	2.74×10^{-2}	1.20×10^{-2}	8.79×10^{-2}	4152
$125_5 \times 125_{25}$	0.193 151	2.98×10^{-2}	1.19×10^{-2}	7.23×10^{-2}	4314
$125_{50} \times 125_5$	0.289 408	8.02×10^{-2}	1.32×10^{-2}	3.71×10^{-2}	8441
$125_5 \times 125_{50}$	0.290 176	8.14×10^{-2}	1.14×10^{-2}	7.77×10^{-2}	8746
$125_{25} \times 125_{10}$	0.291 584	7.99×10^{-2}	1.21×10^{-2}	1.91×10^{-2}	8717
$125_{10} \times 125_{25}$	0.294 464	8.40×10^{-2}	1.28×10^{-2}	7.13×10^{-2}	8782
$125_{50} \times 125_{10}$	0.378 944	1.90×10^{-1}	1.21×10^{-2}	1.34×10^{-2}	17 493
$125_{10} \times 125_{50}$	0.384 384	1.96×10^{-1}	1.12×10^{-2}	6.55×10^{-2}	17 786
$125_{25} \times 125_{25}$	0.396 864	2.34×10^{-1}	1.12×10^{-2}	5.27×10^{-2}	20 978
$125_{50} \times 125_{25}$	0.448 000	5.05×10^{-1}	1.34×10^{-2}	3.84×10^{-2}	42 149
$125_{25} \times 125_{50}$	0.450 240	5.30×10^{-1}	1.24×10^{-2}	7.75×10^{-2}	42 505
$125_{50} \times 125_{50}$	0.481 088	1.21	1.13×10^{-2}	1.12×10^{-1}	85 127

Table 3: Performance comparison of different matrix operations.