Exercise #4

A. The Input parameters

1. $\hat{D}$ is a matrix to be computed (with the dimensions $m \times n$)

$$\hat{D} = \alpha * \; op(\hat{A}) * op(\hat{B}) + \beta * \hat{C}. \tag{1}$$

2. The input parameters in rhs of Eq.(1) are the matrixes $op(\hat{A})$ (the dimension $m \times k$), $op(\hat{A})$ $(k \times n)$ and $\hat{C}$ $(m \times n)$ and the constants α and β. The additional input parameters, the constants *transa* and *transb* are the transposition flags for matrixes $\hat{A}$ and $\hat{B}$, correspondingly, i.e.

$$op(\hat{X}) = \begin{cases} \hat{X} & \text{transa=1} \\ \hat{X}^T & \text{transa=0} \end{cases}$$

3. The integers m, k, n, *transa*, *transb* are the input parameters, which are provided by the user, whereas the components of matrixes $\hat{A}$, $\hat{B}$, $\hat{C}$ and constants α and β are generated by the random numbers generator. The computation of $\hat{A}$, $\hat{B}$, $\hat{C}$, α and β can be performed in one of three programming codes

- Practice04−Input.cs
- Practice04−Input.java
- Practice04−Input.py

These codes have a similar resulting data, which are stored in the file

- Input.txt

although each of them are coded in different languages, correspondingly in $C\#$, Java and Python. Specifically, in this work the input parameters were calculated with use of 'Input.py' code, i.e. in Python language.

B. The computation time

1. We calculated the matrix $\hat{D}$ from Eq.(1) with use of three programming codes

- Practice04−Dgemm.cs
- Practice04−Dgemm.java
- Practice04−Dgemm.py

We performed the computation of matrix $\hat{D}$ using each of these programming codes l_m times, calculating the time t of each computation round. As a result each programming code calculated the array of times

$$\{t_k\}_{k=0}^{k=l_m-1}. \tag{2}$$

We stored this array of times in Eq.(2) in the following file

- time.txt.

C. The statistics

In separate program code

- Practice04−Statistics.py

we have calculated the statistics associated with the array in Eq.(2). To this end we sorted this array, found the minimum t_{min} and maximum t_{max} values in the sorted array as

$$t_{min} = t_0, \tag{3}$$

$$t_{max} = t_{l-1}. \tag{4}$$

We also calculated average value of time

$$t_{av} = \frac{1}{l_m} \sum_{i=0}^{l_m-1} t_i, \tag{5}$$

mean dispersion

$$\sigma = \sqrt{\frac{1}{l_m} \sum_{i=0}^{l_m-1} (t_{av} - t_i)^2}, \tag{6}$$

and median value of time

$$t_{med} = \begin{cases} t_k, \quad \text{where} \quad k = \frac{l_m}{2}, \quad l_m \text{ is an even number,} \\ \\ \frac{(t_k + t_{k+1})}{2}, \quad \text{where} \quad k = \frac{(l_m - 1)}{2}, \quad l_m \text{ is an odd number.} \end{cases}$$

D. The results

We have calculated the results for two sets of input parameters. The first of them was

- Set 1: $m = 100, \quad k = 80, \quad n = 60, \quad trans = 1, \quad transb = 1, \quad l_m = 50.$

We provided the results for the Set 1 of input parameters in Table I. The second set was

- Set 2: $m = 120, \quad k = 100, \quad n = 80, \quad trans = 1, \quad transb = 1, \quad l_m = 50.$

The results for the Set 2 are in Table II.

E. The conclusions

It is seen that for one given set of parameters the $C\#$-code provides with the fastest calculations, Python-code with slowest calculations. Since the rank of matrix in Set 2 is higher than the one in Set 1, the calculation with Set 2 parameters is slower as compared to the one with Set 1. The calculation with Java-code provides with the largest spreading between t_{min} and t_{max} in terms of $(t_{max} - t_{min})/t_{av}$.

TABLE I: The input parameters from Set 1

	t_{min} (sec)	t_{max} (sec)	t_{av} (sec)	t_{med} (sec)	σ (sec)	$(t_{max} - t_{min})/t_{av}$
Python	0.561	0.772	0.619	0.600	0.058	0.341
Java	0.006	0.133	0.022	0.012	0.028	5.773
C#	0.005	0.010	0.007	0.007	0.001	0.714

TABLE II: The input parameters from Set 2

	t_{min} (sec)	t_{max} (sec)	t_{av} (sec)	t_{med} (sec)	σ (sec)	$(t_{max} - t_{min})/t_{av}$
Python	1.259	1.993	1.572	1.553	0.169	0.467
Java	0.015	0.195	0.034	0.024	0.029	5.294
C#	0.012	0.037	0.018	0.016	0.006	1.389