CENG 371 - Scientific Computing
Spring 2022
Homework 4

Kargı, Bora
e2380566@ceng.metu.edu.tr

June 30, 2022

1

All files are submitted through ODTUClass.

2

2.1

The scalings used in the calculation of C and R matrices is due to the expected result of the multiplication $E[CR]_{ij}$ and the variance of the $Var[CR]_{ij}$. By scaling them, we make sure that $E[CR]_{ij} = E[AB]_{ij}$ and $Var[CR]_{ij} = Var[AB]_{ij}$ theoritacally.

2.2

The output plot images of the *main.m* script is given in the figures below.

2.3

First, lets compare the runtimes.

Runtime analysis: Figure 1 shows the runtime of the four different algorithms of the multiplication of $A \times B$. It can be seen that fastest algorithm for large c is *mult_row_nonuni* and the *mult_row_uniform*. Since *mult_proj_Gauss* includes more multiplication and random matrix computation while mult_proj_Gauss_orth involves QR factorizatoin, they are slower. There is an also interesting result : when the c is small, *mult_proj_Gauss* works very fast.

On the other hand, Figure 4 shows the runtime of the matrix multiplication $C \times C^T$. All the run times are slower since C is a bigger matrix then the A and B however the results are similar.

Relative error analysis: Figure 2 shows the relative error analysis of the algorithm for the matrix multiplication $A \times B$. It can be seen that mult_proj_Gauss_orth is clearly the winner and performs much more better than other algorithms. Other three algorithms perform close to eachselves. However as c gets large, distance between *mult_proj_Gauss* and random row sample method increases and *mult_proj_Gauss* performs worse than others.

Figure 5 shows the same error analysis applied over $C \times C^T$. It can be seen that the releation of errors and the c is same with the Figure 2 where mult_proj_Gauss_orth is the best. However there is an importance difference between such two different multiplication. mult_row_uniform performes worser than the *mult_proj_Gauss* and there is huge difference between the errors of mult_row_uniform and mult_row_nonuni. The reason is explained in the next part.

2.4

The reason why non-uniform perform worse for $C \times C^T$ would be the fact that when we assign probabilities of the row, some of the row would be much more important (encodes more information about the matrix). However importance of row of A and columns of B could be distributed evenly so even in uniform case, our result would be closer to the true result. For example even if we sample random unimportant row of A with index i, this could be an important column of B with index i. However for the case $C \times C^T$, i'th row of C and i'th column of C_T is equal and hence, have the same importance. For this reason, if we use uniform probabilities for $C \times C^T$, the importance of the i'th index which is both important for A and B would be decreased a lot since the probability of this index would be such larger if we had used non-uniform probabilities (because for index i, norm of a row of C is equal to the norm of a column of C^T which result in a larger weight).

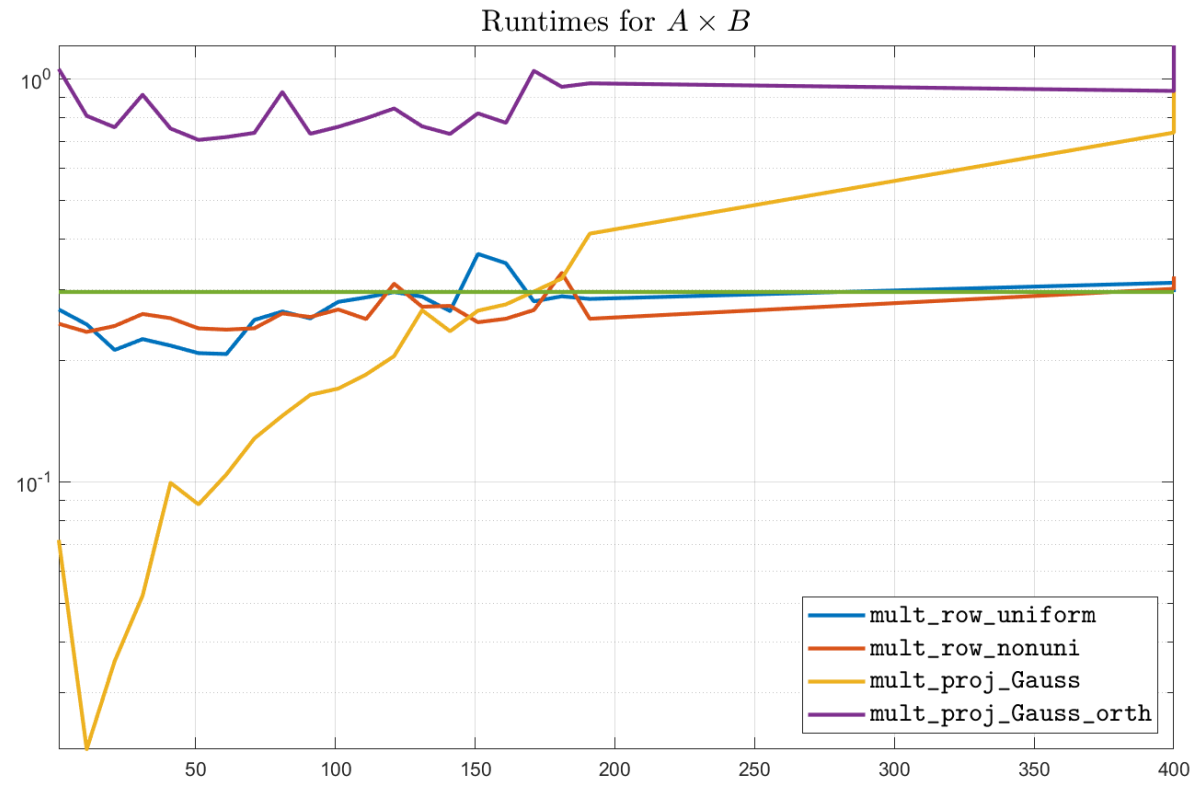

Figure 1:

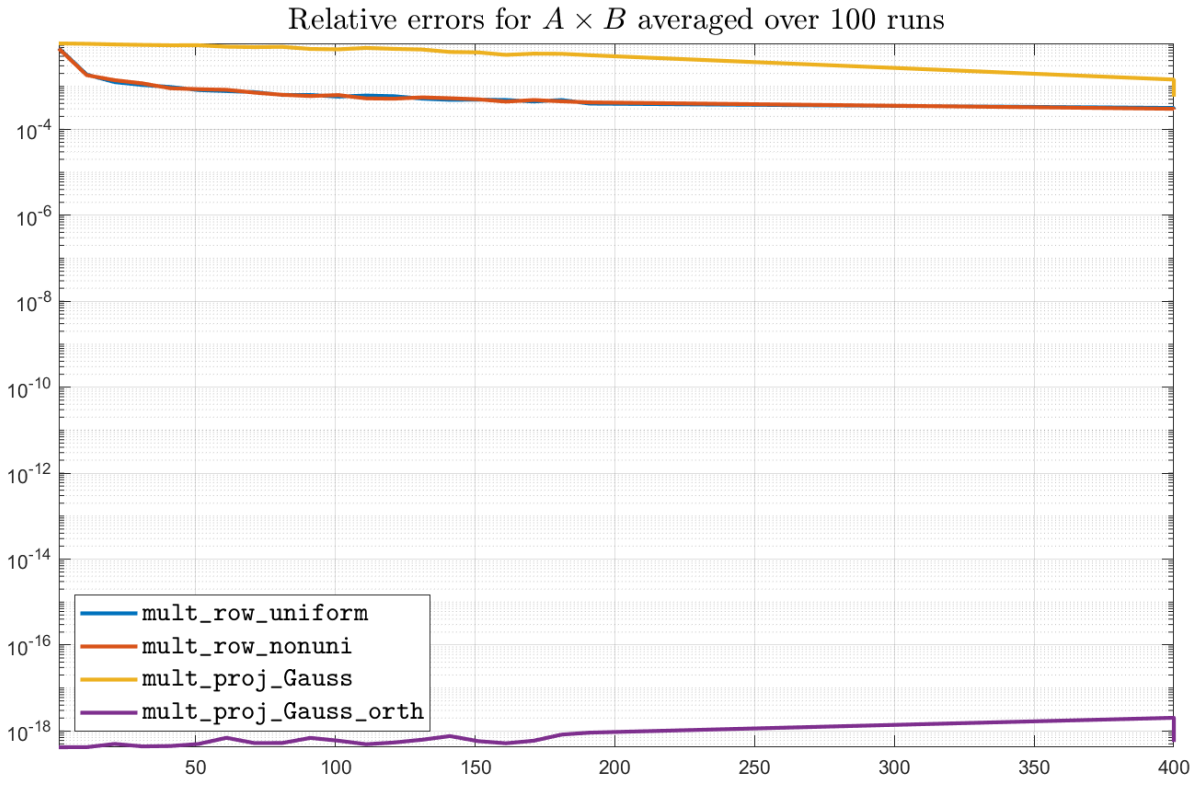

Figure 2:

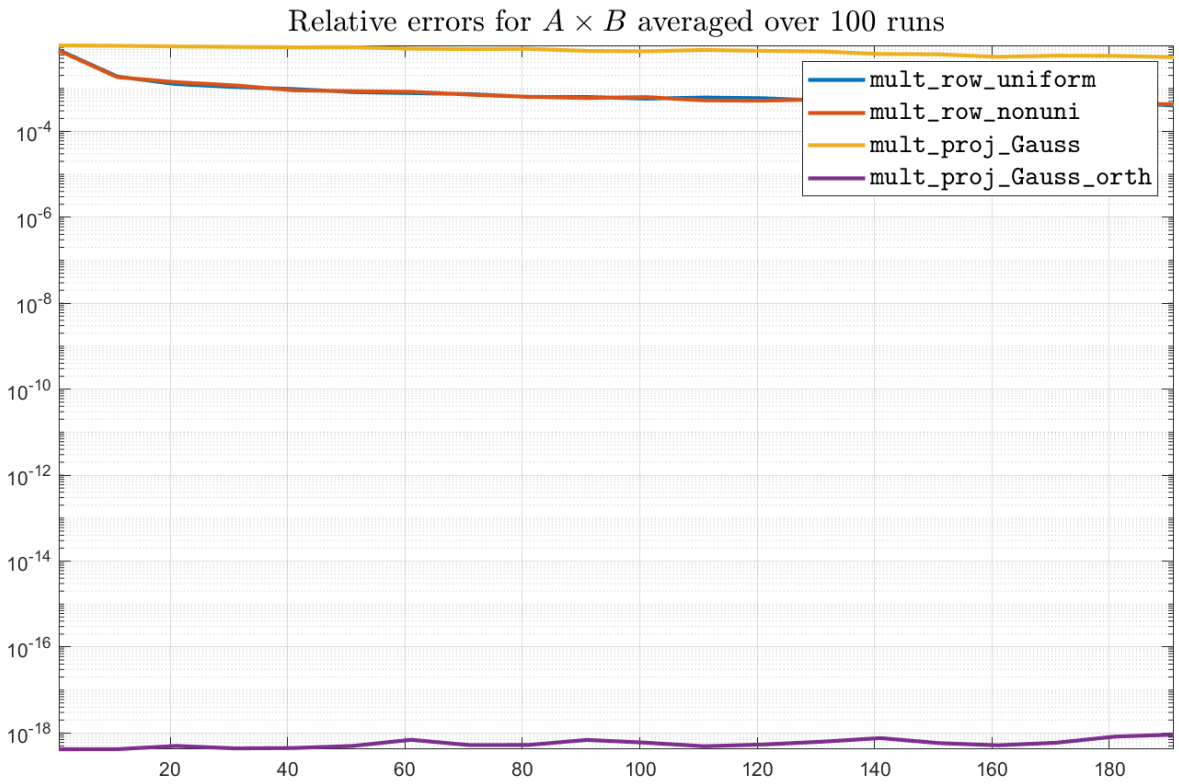

Figure 3:

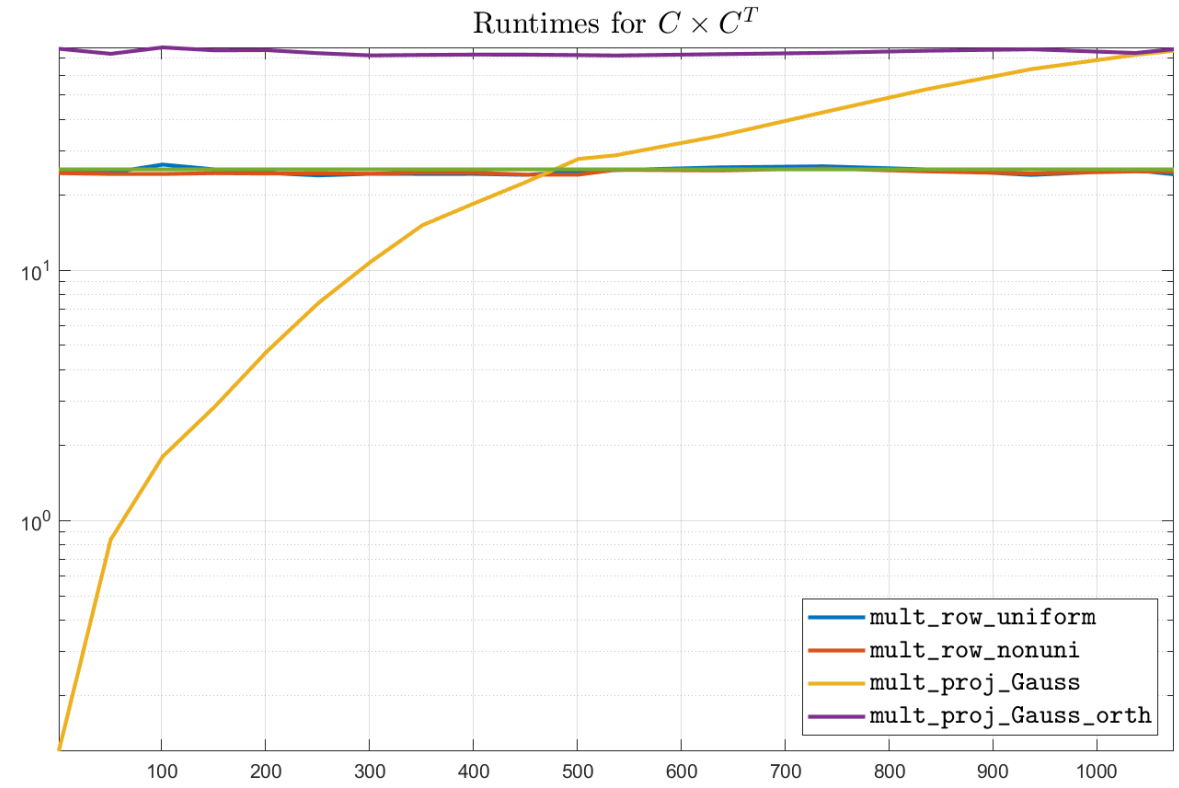

Figure 4:

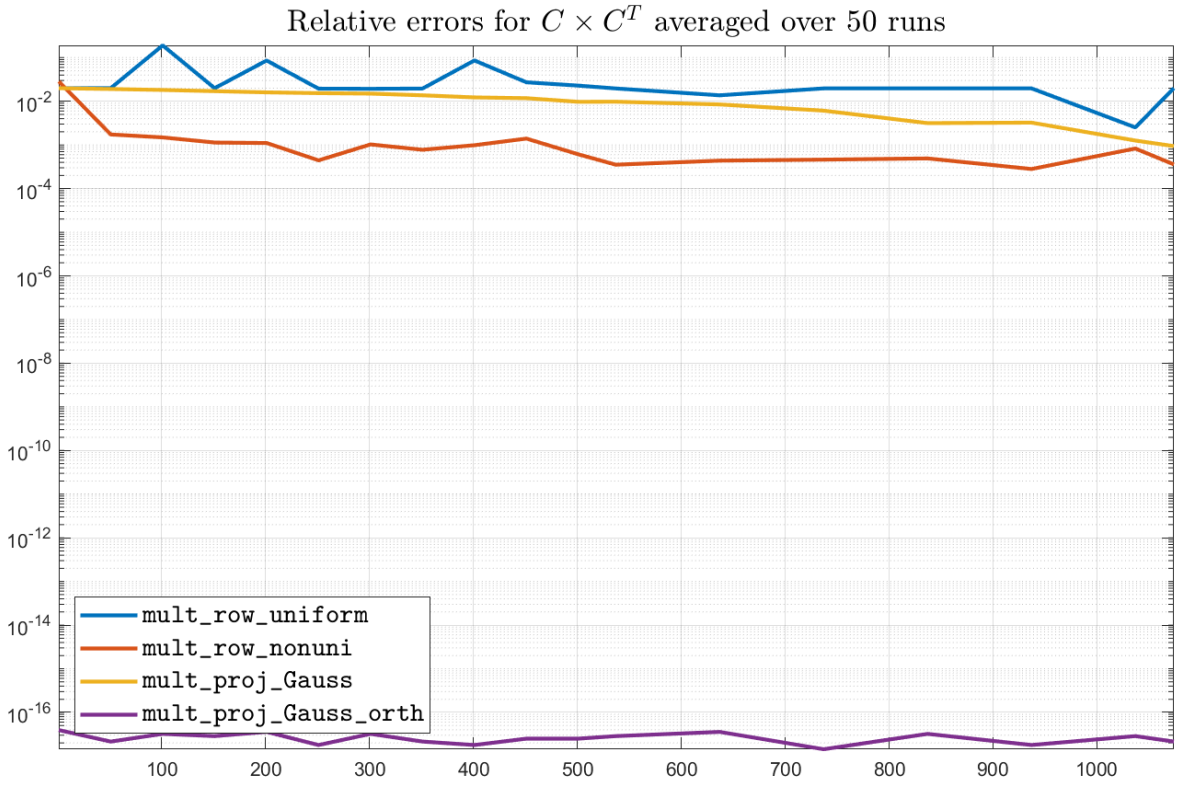

Figure 5:

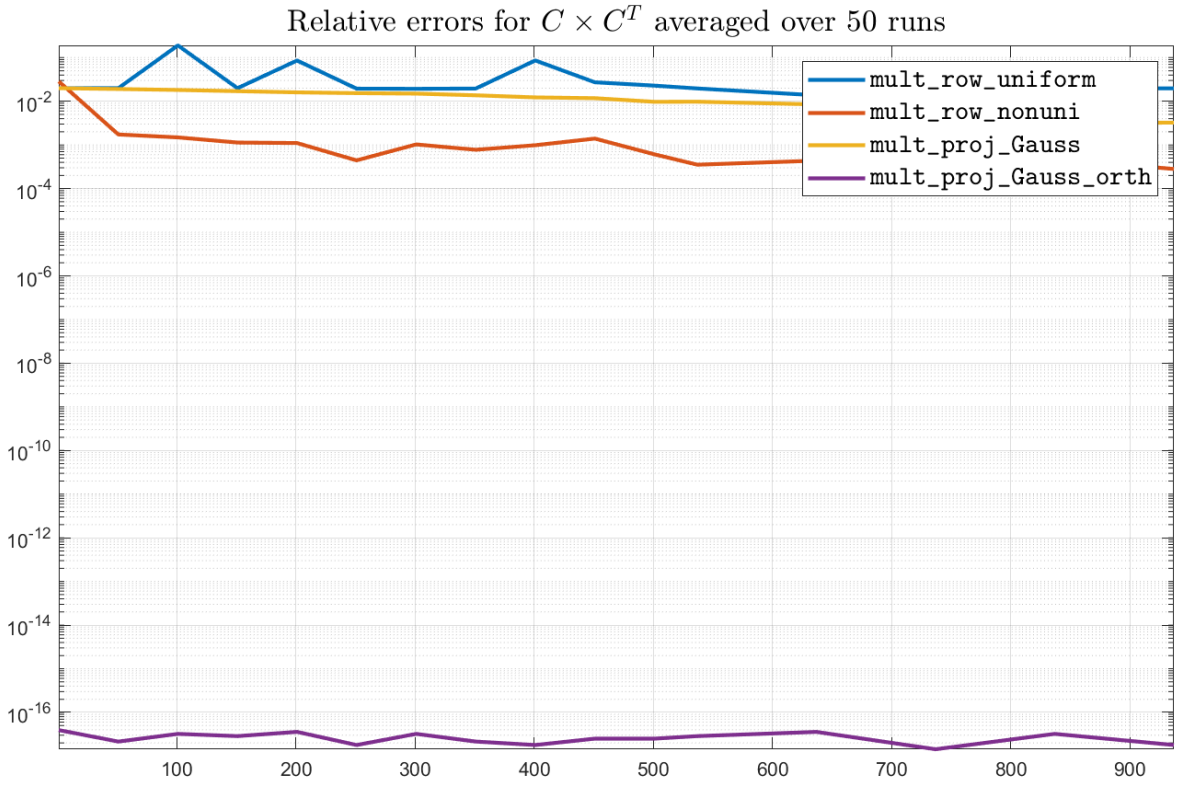

Figure 6: