CENG 371 - Scientific Computing
Spring 2022
Homework 2

Kargı, Bora
e2380566@ceng.metu.edu.tr

April 16, 2022

1 Answers to the Question 2

a) Run time and relative error of the algorithms are shown in the Figure 1. *my_lu* shows the result of LU factorization without pivotting, *my_lu_pp* shows the result of LU factorization with partial pivotting and lastly, *my_lu_cp* shows the result of LU factorization with complete pivotting.

b) Let's compare the result of the algorithms.

- For matrix A, the errors are 0. However there is a difference in run-time since factorization with complete pivotting is the most complex algorithm and naive LU factorization is the simplest one - so naive is faster than partial pivotting which is faster than complete pivotting in terms of speed.
- For matrix B, there is a little difference in terms of computed relative error that algorithm made. We see that partial partition algoritm performed slighlt better than other two whereas naive algorithm performed worst. However these differences are so small and we can not surely say that one performed better than other. In terms of speed, it is obvious that naive algorithm is the faster than complete partition. Partial partition is noticable faster than complete partition algorithm as well however it is similar with naive algorithm in terms of speed.
- For matrix C, difference between naive and partition algorithms is more obvious. We can see that with naive algorithm we calculated 0.0844 error. Both partial partition and complete partition algorithm made an error that is close to 10^{-17} which is very small so it is obvious that partition is better at reducing the error. The difference between partition algorithms are very small, and results show that complete partition algorithm performed better than partial partition algorithm in terms of error. If we compare their run-time, we can see that naive algorithm is still the winner where as complete partition still performs slowest.

Now let us compare with a matrix D which is 1000x1000 matrix with values that are coming from normal distribution ($randn(10000, 10000)$). Since our matrix B and C were a special matrix, what happens if we test algorithms on a randomly initialized big matrix? The result of this experiment is shown in Figure 2. By looking at the relative error and run time of the algorithms for matrix D, we can further support that **naive and partial pivotting algorithms are faster than the complete pivotting algorithm whereas algorithms with pivot is better in terms of how much error it makes compared to the naive one**.

```
>> my_lu
[Computed in 0.001350] Result of matrix [A] :      0

[Computed in 4.185248] Result of matrix [B] :    4.4153e-17

[Computed in 4.175284] Result of matrix [C] :     0.0844


>> my_lu_pp
[Computed in 0.002337] Result of matrix [A] :      0

[Computed in 4.255601] Result of matrix [B] :    3.6760e-17

[Computed in 4.226624] Result of matrix [C] :    4.3490e-17


>> my_lu_cp
[Computed in 0.003378] Result of matrix [A] :      0

[Computed in 5.392654] Result of matrix [B] :    4.0526e-17

[Computed in 5.073366] Result of matrix [C] :    3.5546e-17
```

Figure 1: Comparison of running time and relative error of the LU decomposition algorithms.

```
>> my_lu
[Computed in 4.426169] Result of matrix [D] :    6.0041e-12

>> my_lu_pp
[Computed in 4.186237] Result of matrix [D] :    8.5272e-15

>> my_lu_cp
[Computed in 5.456152] Result of matrix [D] :    4.7256e-15
```

Figure 2: Comparison of running time and relative error of the LU decomposition algorithms with respect to matrix D.