LU Factorization with Partial Pivoting

Sérgio Caldas* and Filipe Oliveira†
University of Minho
Computer Science Department
*Email: *a57779@alunos.uminho.pt, †a57816@alunos.uminho.pt*

***Abstract*—When Linear Systems appears, we need to resolve $Ax = b$ for very different b vectors. For solve a linear system, the Gaussian eliminations with pivoting is the most efficient and accurate way to solve, almost all the work falls on the matrix A. When A is big, and we need to solve linear systems with the same A and this linear systems are very different of each other, we must to avoid repeating the steps of Gaussian elimination on A for every different b, for that we use the LU factorization. This factorization records the steps of Gaussian elimination.**

In our case of study, we have to explore the LU factorization with partial pivoting, for that, it was provided to us two files that implements LU factorization without pivoting. The first file ("BLAS2LU.m") are for rectangular matrices.The second file ("BLAS3LU.m") is a block version for square matrices that use "BLAS2LU.m" to obtain the factorization of rectangular sub-matrices.

The objective of this work assignment it is implement the version of "BLAS2LU.m" with partial pivoting and implement a version of "BLAS3LU.m" with partial pivoting and use "BLAS2LU.m".

***Index Terms*—LU Factorization[1] - Gaussian Eliminations - Linear Systems - Pivoting**

1. Introduction

1.1. LU Factorization without Partial Pivoting

The LU Factorization[2] is a method used to solve a linear system of equations. This factorization use two matrices, one upper matrix (triangular superior matrix) and a lower matrix (triangular inferior matrix). The LU Factorization can be viewed as the matrix form of Gaussian elimination. A math definition for LU factorization is:

> Let A be a square matrix. An LU factorization refers to the factorization of A, with proper row and/or column orderings or permutations, into two factors, a lower triangular matrix L and an upper triangular matrix U.

The equation that translate this definition is:

$$A = LU \tag{1}$$

In the lower matrix, all elements above the diagonal are zeros, in the upper matrix all elements below the diagonal are zeros and in the upper matrix all elements of diagonal are 1.

1.2. LU Factorization with Partial Pivoting

In the LU Factorization with Partial Pivoting, the factorization refers often to the LU factorization with row permutations only. The LU Factorization can be defined for the following equation:

$$PA = LU \tag{2}$$

Where L and U, are the lower and the upper matrix respectively and P, refers to a permutation matrix. The permutation matrix, is a binary matrix where in each row and column only have one entry of 1, the others elements are zeros. When we multiply permutation matrix for other matrix, this produce the permutation in the rows or columns of the other matrix.

2. Hardware characterization

The computing platform, our team laptop, is a Intel® Core™ i7-3635QM (Ivy Bridge architecture). The system features 16 GB of DDR3 RAM, supported at a frequency of 1600MHz, divided in 2 memory channels.

The characteristics of the CPU on the computing platform is presented in the table 1.

1. Lecture 12 LU Decomposition
2. LU Decomposition

System	team laptop
# CPUs	1
CPU	Intel® Core™ i7-3635QM
Architecture	Ivy Bridge
# Cores per CPU	4
# Threads per CPU	8
Clock Freq.	2.4 GHz
L1 Cache	128KB 32KB per core
L2 Cache	1024KB 256KB per core
L3 Cache	6144KB shared
Inst. Set Ext.	SSE4.2 & AVX
#Memory Channels	2
Vendors Announced Peak Memory BW	25.6 GB/s
Measured Peak Memory BW	15.5 GB/s

Table 1: Architectural characteristics of the evaluation platform.

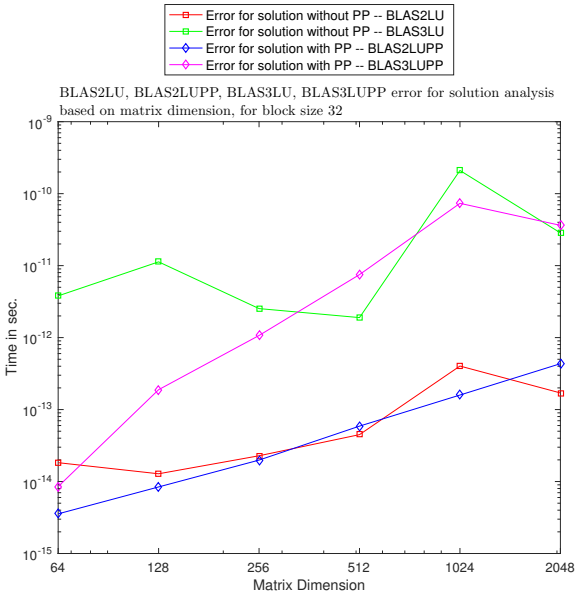

Figure 1: Error for solution analysis based on matrix dimension, for block size 32

3. Changes in the Provided Code

The first step of this work, was analyzing the provided code by the professor, and understand it. After that, we changed the code in order to allow rows permutations (this changes were made in the "BLAS2LU.m" file).

In order to be able to measure time and error for the several algorithms, changes were made to the return parameters of each function.

Every function, additionally to the returned A L U P, returns also vars time and error.

4. Analysing the solutions validity and error percentage

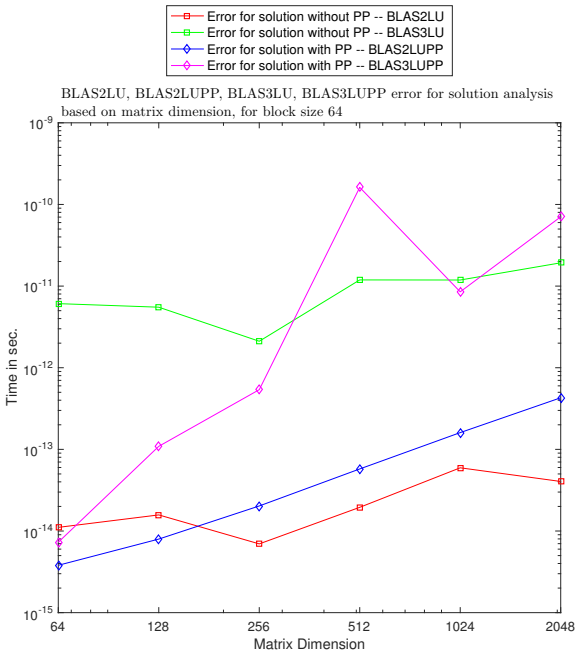

Figure 2: Error for solution analysis based on matrix dimension, for block size 64

Several auxiliar scripts were produced in order to relate both error and time for solution for the several matrix sizes and block dimensions. The corresponding error analyse results are shown in figures 1 and 2.

As predicted, the error value for algorithms that recur to Partial Pivot presents a value several orders of magnitude below the one presented by algorithms without Partial pivoting.

5. Analysing the time for solutions based on algorithm and block dimension

Proved the algorithm correctness, we should analyse the total time for solution based on both the algorithm and the block dimension.

Please denote that the block sizes were chosen based on the average CPU cache line size.

The corresponding time analyse is shown in figures 3 and 4. A relation between matrix block dimension and time for solution is presented on figure 5.

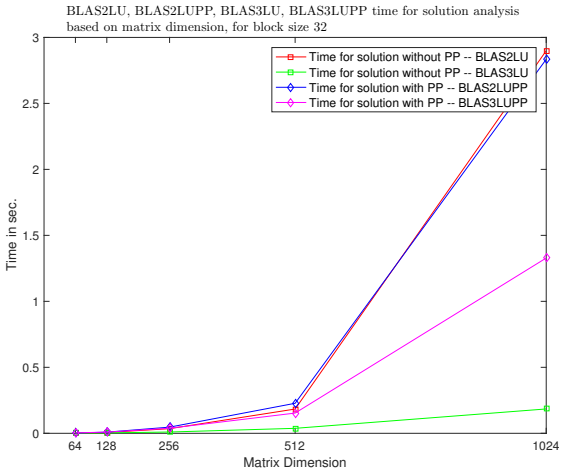

Figure 3: Time for solution analysis based on matrix dimension, for block size 32

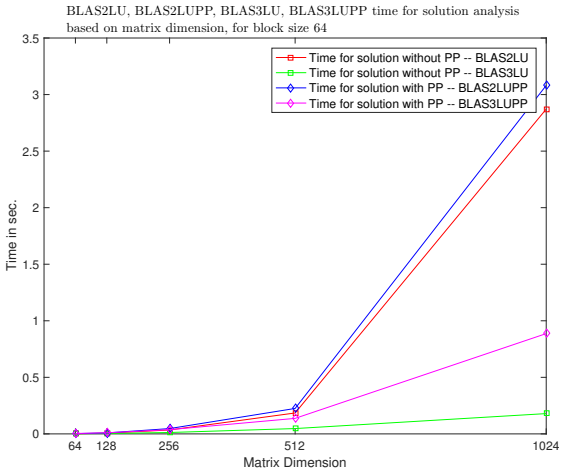

Figure 4: Time for solution analysis based on matrix dimension, for block size 64

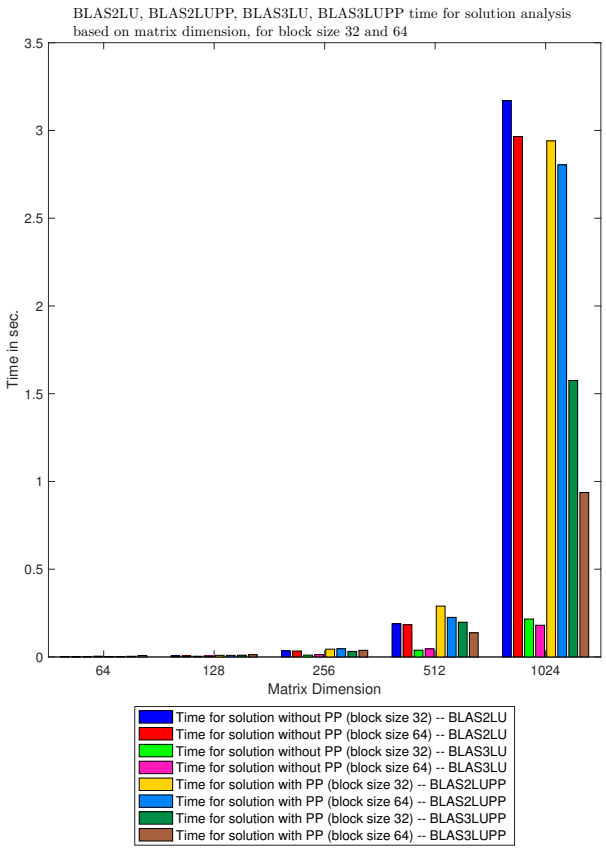

Figure 5: Time for solution analysis based on matrix dimension, for block size 32

As we can visualy infer, a block dimension of 32 produces better overall results in terms of time for solution. That was as expected result, since the computing platform used in the tests has as Cache line size of 32 Bytes.

6. Conclusion

As we can see, in the previous results, the relative error associated to the LU Factorization with partial pivoting is less then the LU Factorization without partial pivoting, because of this we can conclude that the LU Factorization with partial pivoting is more accurate then FU Factorization without partial pivoting.

In terms of computation time the most expensive algorithms are the ones with partial pivoting. The total time for solution LU Factorization with partial pivoting for BLAS2 has revealed to be the most expensive for block sizes of 32 and 64.

As predicted, the usage of blocks in the algorithms, implied a better time for solution.

During this work we have come across with some problems, essentially on the implementation of the BLAS3

LU factorization with partial pivoting. The bigger problem was the row permutations on this algorithm. We think that this difficulties was surpassed.

References

[1] J. W. Demmel, N. J. Higham, and R. S. Schreiber. Stability of block lu factorization. *Numerical linear algebra with applications*, 2(2):173–190, 1995.

[2] J. Dongarra, M. Faverge, H. Ltaief, and P. Luszczek. Achieving numerical accuracy and high performance using recursive tile lu factorization with partial pivoting. *Concurrency and Computation: Practice and Experience*, 26(7):1408–1431, 2014.

[3] A. George and E. Ng. An implementation of gaussian elimination with partial pivoting for sparse systems. *SIAM journal on scientific and statistical computing*, 6(2):390–409, 1985.

[4] E. S. Quintana-Ortí and R. A. Van De Geijn. Updating an lu factorization with pivoting. *ACM Transactions on Mathematical Software (TOMS)*, 35(2):11, 2008.