Communication Avoiding LU Factorization using Complete Pivoting

Implementation and Analysis[*]

Avinash Bhardwaj
Department of Industrial Engineering and Operations Research
University of California, Berkeley
Berkeley, CA 94704, USA

December 12, 2010

Abstract

Pivoting, in various applications is a challenge for minimizing communication, since straightforward implementations of well-known pivoting schemes do not seem to permit us to attain communication lower bounds. In the proposed work, we try to implement and analyze, communication avoiding LU decomposition with complete pivoting, the communication bounds for which have already been derived. We compare the stability and accuracy statistics with respect to existing pivoting schemes, GEPP and GECP.

Keywords: Communication avoiding, Complete pivoting, Pivoting schemes, LU decomposition.

1 Introduction

Solving linear systems of equations is one of the most common operation in scientific computing. These applications frequently lead to solving very large dense set of linear equations, often with millions of rows and columns, and solving these problems is very time consuming.

In this work we present a MATLAB© implementation of the communication-avoiding LU factorization (CALU) algorithm for computing the LU factorization of a dense matrix A distributed in a two dimensional layout, developed by [DG]. This version of CALU is based on a complete pivoting strategy, that we show is numerically stable in practice. Furthermore, we contrast the results thus obtained with classical pivoting schemes GEPP and GECP.

CALU has two main characteristics, first, it is latency avoiding, as it allows for a significant decrease in the number of messages exchanged during the factorization relative to conventional algorithms. Second, unlike conventional algorithms, CALU allows the usage of the best available sequential algorithms for computing the LU factorization of a block submatrix, as for example the recursive algorithms.

This work is structured as follows, in Section 2 we provide an overview of the algorithm, followed by a detailed summary of the implementation in Section 3. Section 4 illustrates the results obtained on the numerical and stability analysis of the algorithm. We finally conclude in Section 5, also providing directions for future extensions and research ideas.

2 The Algorithm

In this section, we provide an overview of the communication avoiding LU factorization algorithm with complete pivoting as developed by [DG].

Assume we start with an $m \times n$ matrix, and use panels consisting of b columns. The next b pivot columns can be chosen in communication-avoiding way by a reduction operation on panels, analogous to how CALU [GDX08] uses a reduction operation on blocks in a panel to choose the next b pivot rows.

Suppose we have a tree, with panels at the leaves. Each interior vertex of the tree will take b candidate pivot columns from its left child, b candidate pivot

[*]Implementation and Analysis based on the algorithm from a work in progress by Prof. James Demmel and Prof. Ming Gu.

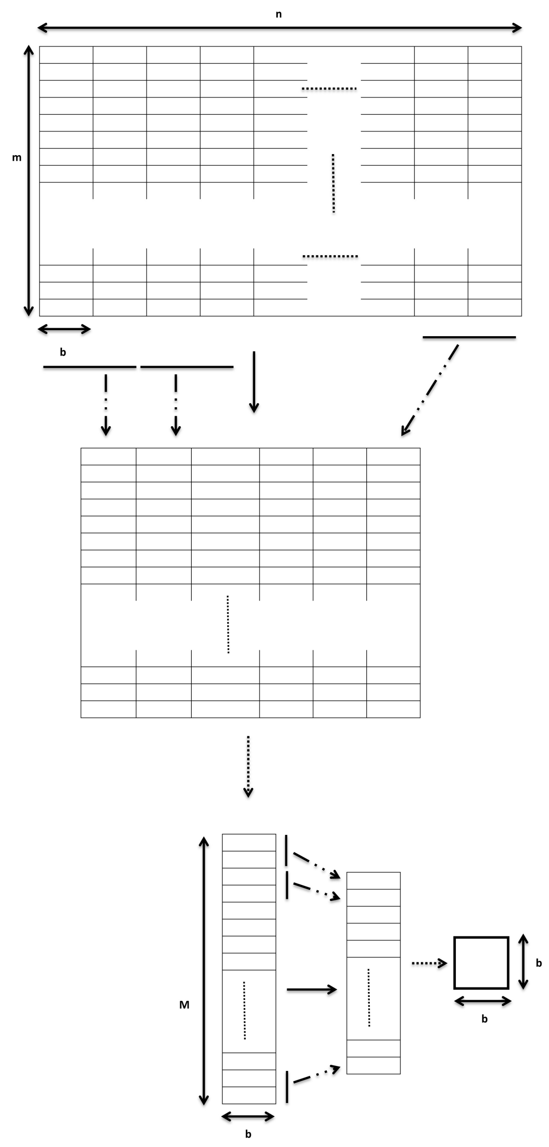


Figure 1: Choosing the best $b \times b$ submatrix

columns from its right child, do RRQR [DG] on the resulting $2b$ columns, and pass the (indices of the) first b columns it selects to its parent. The b columns chosen by the root of the tree are the actual pivot columns to use. The algorithm then uses TSLU [GDX08] on these best b columns to pick their best b rows. The resulting $b \times b$ submatrix is then permuted to the upper left corner of the matrix, and perform b steps of LU with no pivoting. Then the process repeats itself on the trailing submatrix or the Schur's complement. This same "tournament pivoting" (Figure 1) has been explored by [GDX08]

The intuition is that by picking the best b columns, and then the best b rows from these columns, we are in effect picking the best $b \times b$ submatrix overall, which is then permuted to the left corner. The communication costs are clearly minimal, since it just uses previously studied components [DG, GDX08]. To summarize, we present Algorithm 1.

Algorithm 1 To compute the LU factorization of a matrix A while avoiding communication using complete pivoting.

```
Input A dense $m \times n$ matrix $A$.
Output The LU decomposition of the matrix $A$,
    $PAQ = LDU$
.
I Partition the matrix $A$ into $m \times b$ column panels.
II Use tournament pivoting (with RRQR) to find the best $b$ column panel.
III For the $m \times b$ column obtained in Step II, use tournament pivoting (with TSLU) to obtain the best $b$ rows of this column panel.
IV Pivot this $b \times b$ submatrix thus obtained to the top-left corner.
V Perform a step of LU without pivoting on this submatrix.
VI Repeat steps I-V on the trailing matrix $A = SchurComplement(A)$
```

Algorithm 1 yields the LU decomposition of the input matrix A. The output of the algorithm gives the row permutation matrix P, column permutation Q, the strict lower and upper diagonal components L and U, and a diagonal matrix D, which should ideally contain the singular values of A.

3 Implementation

Algorithm 1 described in Section 2 was implemented in `MATLAB`© . `MATLAB`© 's inbuilt routine `qr()` was used in place of RRQR, as it operates in a similar communication avoiding manner and to enhance the computational performance for large test matrices. The tournament pivoting was implemented for both Steps II and III of the algorithm. `MATLAB`© 's inbuilt `lu()` was utilized for Step III instead of TSLU for a similar reason stated as above. The `MATLAB`© code is not included in this document, however is available on request.

4 Results and Analysis

We categorized the test matrices while collecting the statistics for contrasting stability measures for the three pivoting schemes, CALU-CP, GECP, and GEPP, into the following three categories.

I. Matrices with structures,

$$A = D_1 R D_2$$

where D_1 and D_2 are the diagonal matrices with large random entries and R is a random matrix whose entries follow a standard normal distribution.

II. Matrices with structures,

$$A = R_1 D R_2$$

where D is the diagonal matrix with large random entries and R_1 and R_2 are random matrices whose entries follow a standard normal distribution.

III. The "Bad Matrix": A matrix with 1's on the diagonal, -1's on the strict lower diagonal, 1's in the last column and zeros elsewhere.

The following statistics were collected for the above matrices, and were expected to behave as stated below.

i. The condition number of the strict lower diagonal matrix L, expected to be of $\mathcal{O}(n)$.

ii. The condition number of the strict upper diagonal matrix U, expected to be of $\mathcal{O}(n)$.

iii. Entries of the diagonal matrix D, should be sorted in the decreasing order, i.e.

$$\left|\frac{D_{i+1}}{D_i}\right| \leq 1 \quad \forall\ 1 \leq i \leq n$$

iv. D should ideally contain the singular values of the input matrix A, i.e.

$$\max_{1\leq i\leq n}\left\{\left|\frac{D_i}{\sigma_i}\right|, \left|\frac{\sigma_i}{D_i}\right|\right\} \leq 1$$

where σ_i is the i-th singular value of A.

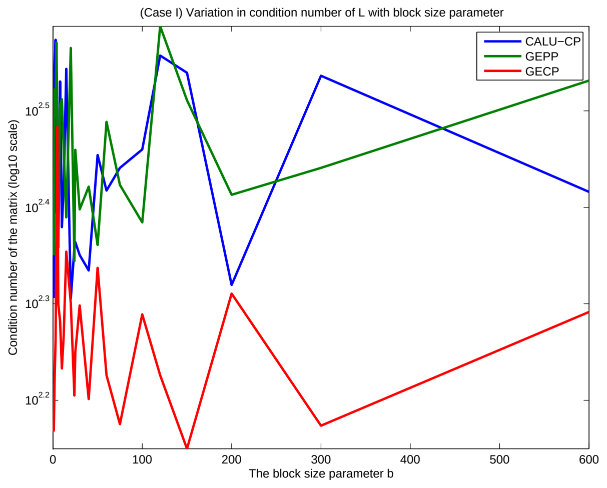
v. The algorithm should be backward stable, i.e. the relative backward error defined by,

$$\rho = \frac{||P * A * Q - L * D * U||_2}{||A||_2} \leq \epsilon$$

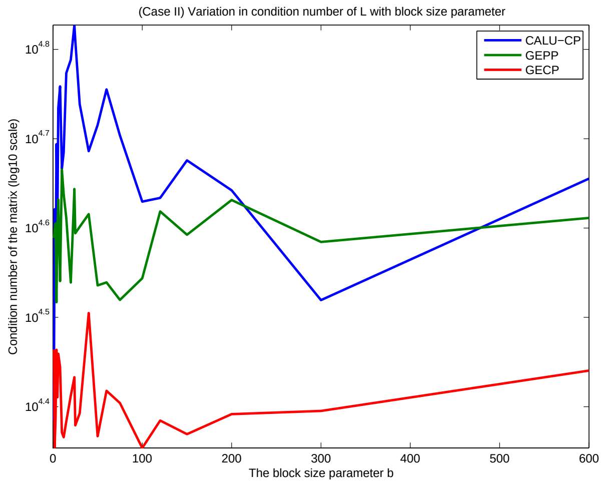
where ϵ denotes the machine epsilon (*macheps*) ($= 2.2204 \times 10^{-16}$ for `MATLAB`©)

The algorithm was tested for different values of n the dimension of the square matrix A, namely $n = \{200, 300, 500, 600\}$. The following plots illustrate the results obtained for $n = 600$[†]

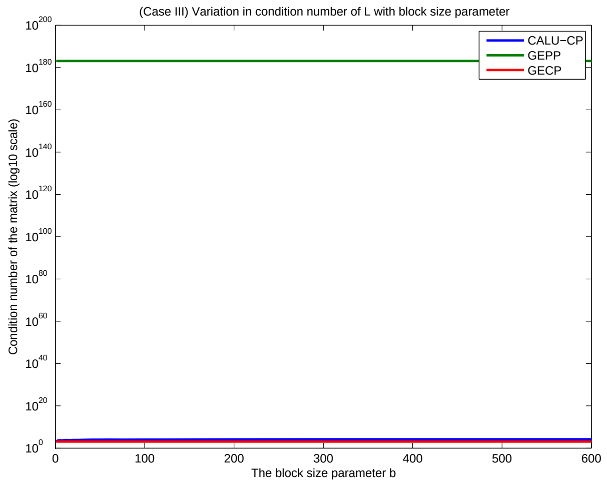
[†]The results obtained for other values of n are similar and are available on request.



(a) Case I

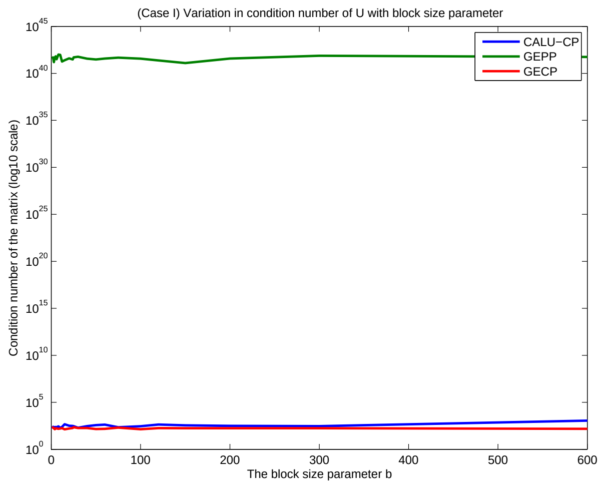


(b) Case II

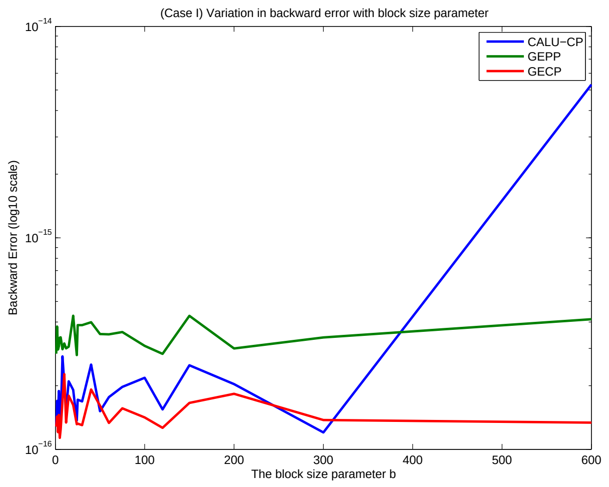


(c) Case III

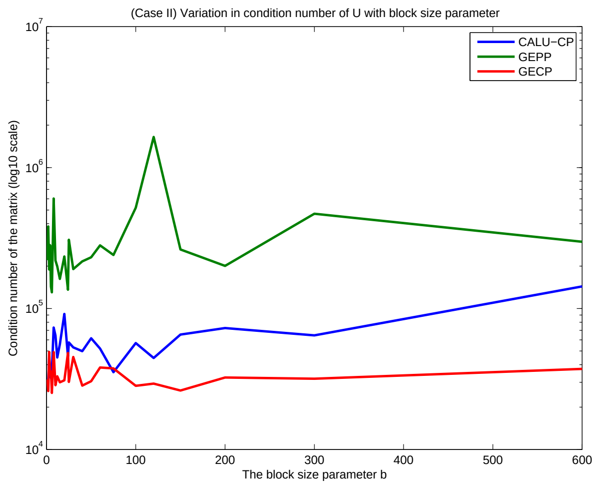
Figure 2: Variation in condition number of L with block size parameter b, $n = 600$



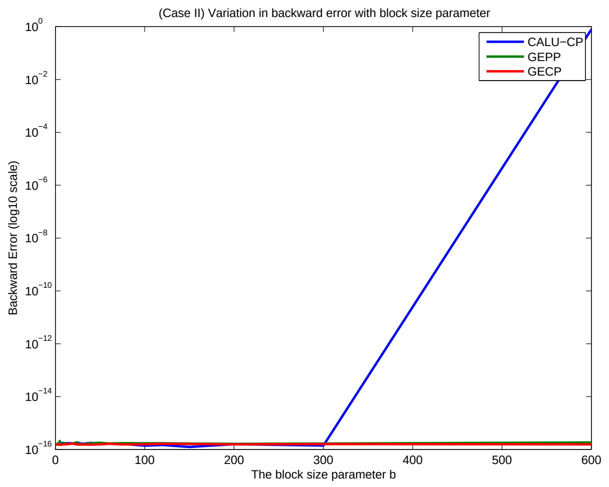
(a) Case I



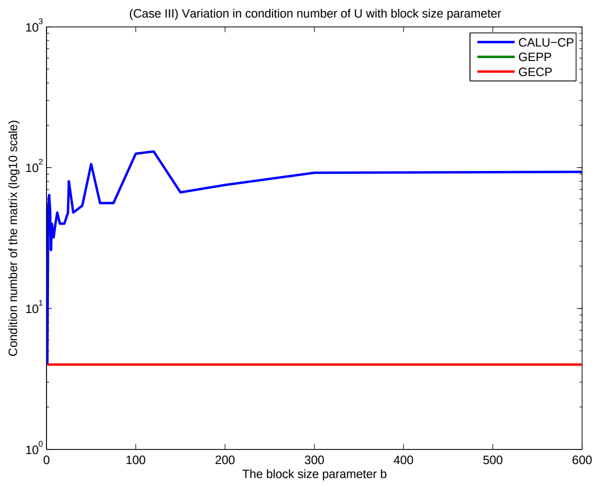
(a) Case I



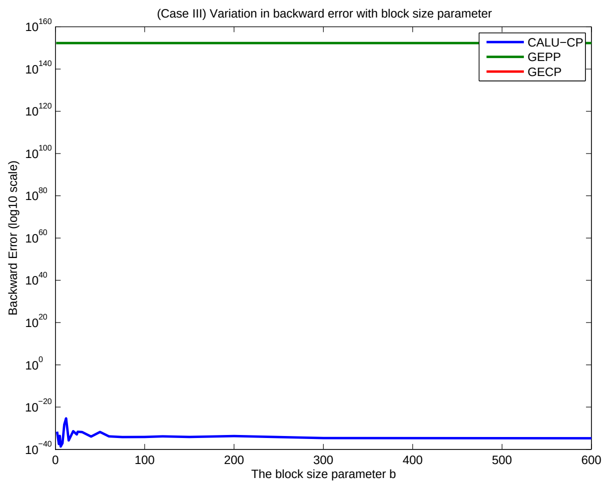
(b) Case II



(b) Case II



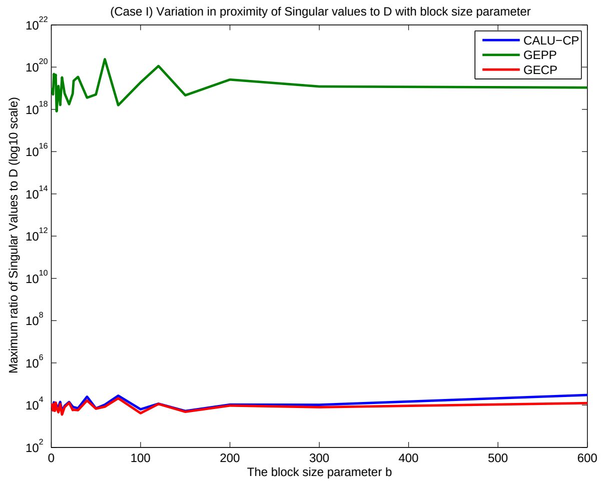
(c) Case III



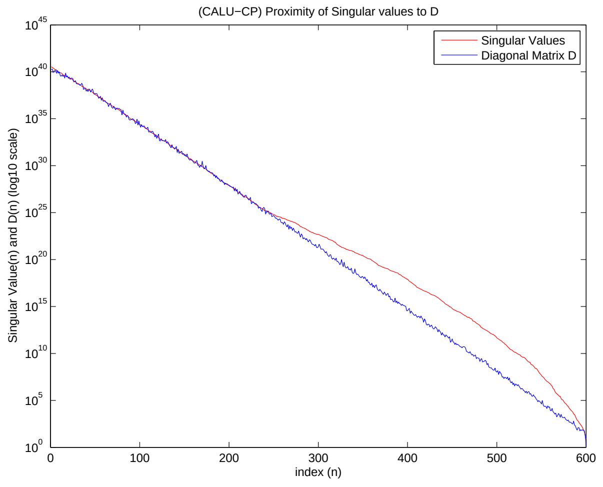
(c) Case III

Figure 3: Variation in condition number of U with block size parameter b, $n = 600$

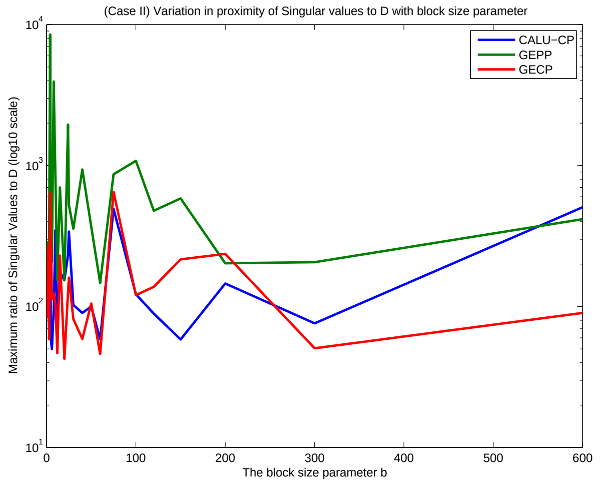
Figure 4: Variation in relative backward error with block size parameter b, $n = 600$



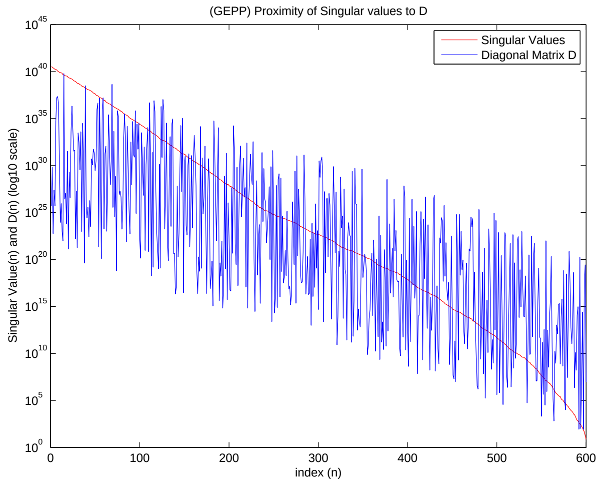
(a) Case I



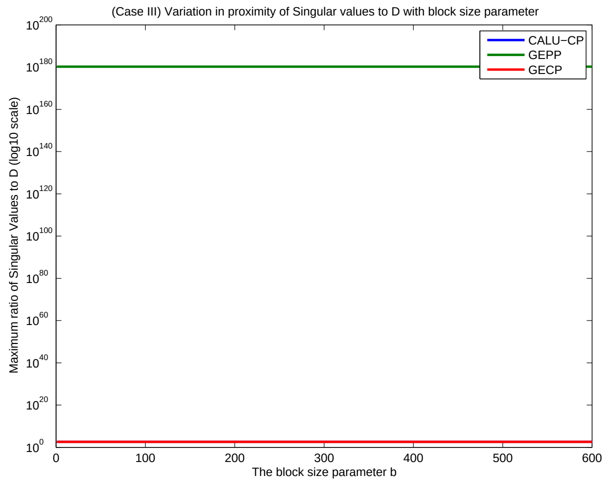
(a) CALU-CP



(b) Case II

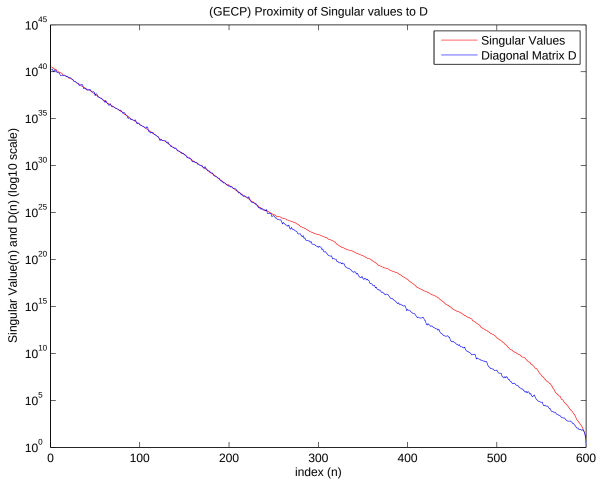


(b) GEPP



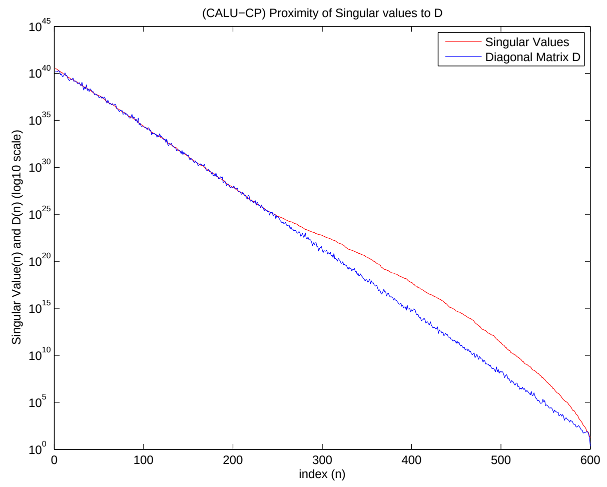
(c) Case III

Figure 5: Variation in proximity of singular values σ to D with block size parameter b, $n = 600$

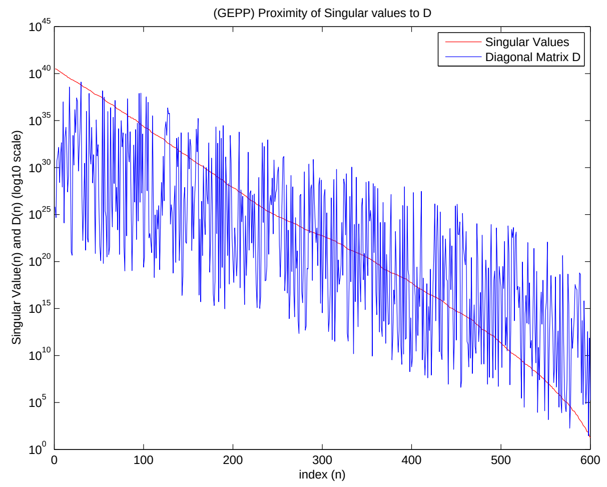


(c) GECP

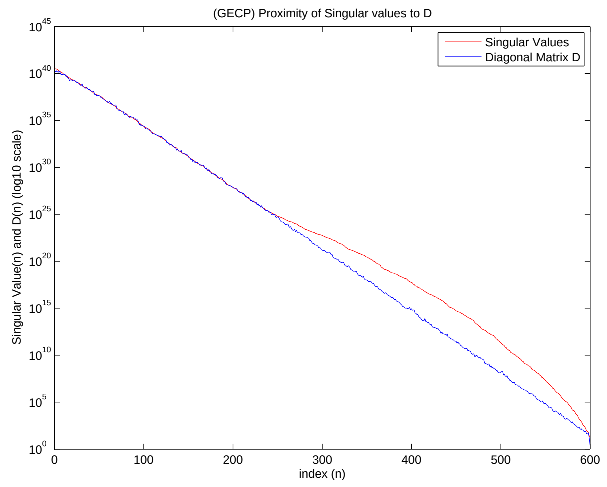
Figure 6: Proximity of singular values σ to D, $n = 600, b = 50$



(a) CALU-CP



(b) GEPP



(c) GECP

Figure 7: Proximity of singular values σ to D, $n = 600, b = 150$

Discussion of Results

The condition numbers of L and U in Case I and Case II are comparable and of linear order for all CALU-CP and GECP, wherein GEPP performs the worst of the three. In Case III, a little thought yields that partial pivoting will have a condition numbers of $\mathcal{O}(2^n)$, which explains the erratic behavior from GEPP in Case III, where condition numbers are $\sim 10^{180} \approx 2^{600}$.

The new algorithm is numerically stable as evident from the plots corresponding to the relative backward errors (Figure 4). The only inconsistent behavior is in Case II, for $b = 600$, i.e. when the entire matrix can be accommodated in the system cache. The reason this possibly can be attributed to might be that we do LU without pivoting in this case.

The accuracy of the results is further strengthened by the results obtained for the proximity ratio of the singular values σ and the diagonal matrix D. The results are consistent and as expected. Again, partial pivoting is the worst performer here. We also look closely at the individual singular values, in Figures 6 and 7 for $b = 50$ and $b = 150$, from which it is clear that CALU-CP and GECP more or less imitate the behavior of singular values, whilst GEPP's erratic behavior is unexplained.

5 Concluding Remarks

From the results obtained the following conclusions can be drawn:

i. The condition numbers for L and U obtained were observed to be of linear order for the tested cases ($n = 200, 300, 500, 600$).

ii. The Diagonal matrix D does proximate the singular values of the original matrices in the tested cases ($n = 200, 300, 500, 600$).

iii. The relative backward errors observed in for the test cases were of the order of machine epsilon (2.2204×10^{-16} in case of `MATLAB`©)

iv. The algorithm for Communication avoiding LU decomposition with complete pivoting was seen to be computationally and numerically stable for the tested cases ($n = 200, 300, 500, 600$).

v. The algorithm when compared to the classical GECP algorithm was stable at par, wherein the GEPP algorithm, was the worst of the lot, and behaved erratically for most of the test cases as expected.

Scope for future work

i. Theoretical bounds can be derived for the statistics observed in the work, namely the condition numbers of L and U, the relative backward error ρ, and the proximity ratio of the singular values to the diagonal matrix D.

ii. Parallel implementations can be done and tested for computational performance.

References

[DG] James Demmel and Ming Gu. Communication-avoiding pivoting.

[GDX08] Laura Grigori, James W. Demmel, and Hua Xiang. Communication avoiding gaussian elimination. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, SC '08, pages 29:1–29:12, Piscataway, NJ, USA, 2008. IEEE Press.