European Journal of Mathematics and Statistics
Vol 6 | Issue 1 | February 2025
ISSN 2736-5484

RESEARCH ARTICLE

Inverting Symmetric Positive Definite Matrices using Divide and Conquer Mathematical Technique with LU Factorization

Abdel Radi Abdel Rahman Abdel Gadir Abdel Rahman[1,*], Shady Seed El Okuer[2], and Musa Adam Abdullah[3]

ABSTRACT

We looked at the solution of a system of equations $Ax = b$ with symmetric positive definite coefficient matrix A that has singular and nearly singular values. Our technique, which is based on the Divide and Conquer strategy, combines the LU Factorization algorithm with the Divide-and-Conquer technique (D&C algorithm). The matrix was transformed into a product of the type LU using the LU Factorization, where L is a lower triangular matrix and U is an upper triangular matrix. MATLAB was used to implement the algorithm and simulate it as a user-subroutine. In order to reduce the round-off error, particularly for sensitive systems, the user-subroutine takes into account MATLAB characteristics. A non-singular matrix and an ill-conditioned matrix were both numerically demonstrated. Analysis was done on the impact of round-off error. We contrasted the results with those from earlier studies that employed LU factorization.

Keywords: Divide and conquer algorithm, LU algorithm, mathematical software, nearly singular matrix.

Submitted: December 07, 2024
Published: February 24, 2025

10.24018/ejmath.2025.6.1.387

[1] *Professor, Department of Mathematics, Faculty of Education, Omdurman Islamic University, Sudan.*
[2] *PhD, Gaza, Palestine.*
[3] *Department of Mathematics, College of Computer Science and Information Technology, University of Holy Quran and Tasssel of Science, Sudan.*

[*] ***Corresponding Author:***
e-mail: dibdelradi78@oiu.edu.sd

1. Introduction

One of the fundamental issues that arises frequently in the domains of science and engineering is the problem of matrix inversion. It is typically a crucial component of many solutions, such as the initial steps in robotic control, electromagnetic systems, statistics, physics, optimization, and single processing. [1], [2] contains a few efficient direct solution techniques that take advantage of displacement representation. In [3], alternative iterative techniques were put forth. The latter techniques can work well with well-conditioned inputs and nontrivially expand some earlier work for general input matrices [4]. A system of linear equations is typically produced when computational mathematics problems are solved using partial differential equations (PDE), integral equations, and boundary value issues in ordinary differential equations (ODE). A linear system of equations $Ax = b$ can be solved using the LU factorization method in the exceptional case where A is a symmetric positive definite matrix. [5]–[8]. At that point, all of A's eigenvalues are positive and $A^T = A$. These systems are found in many significant applications and are typically linked to the "minimum energy principle". The following outcome is necessary for the LU Factorization method to work. It is possible to factorize any symmetric positive definite matrix A into A = LU, where U is an upper triangular matrix and L is a lower triangular matrix. In this research, MATLAB is used for all computational results [9] utilizing several computational digits. The following formula is used to calculate the relative error:

$$res_{inv} = \frac{max\left[\parallel I - AA^{-1} \parallel, \parallel I - A^{-1}A \parallel\right]}{\parallel A \parallel}$$

We used the divide and conquer strategy to calculate the inverse of matrix A by factorizing the matrix into a product of lower and upper triangular matrices. This work's primary goal is to use the Divide and Conquer Algorithm to determine the inverse of almost singular positive definite matrices.

To do this, the system's sensitivity will be addressed by increasing the computational decimal digits to two. An overview of the Divide and Conquer and LU Factorization algorithms for determining the inverse of nearly singular matrices is provided in Section 2. Different examples of non-singular and nearly singular systems will be shown in Section 3 and solved for an increasing number of digits in the calculations. Our own subroutines in the MATLAB mathematical code will be used to simulate the calculations. Section Four will contain comments and concluding remarks.

2. Divide and Conquer Strategy for Matrix Inversion

One of the most crucial algorithms is divide and conquer (D&C). The way it operates is by breaking a problem down into two or more similar subproblems, which can then be addressed directly. A solution to the original problem is then obtained by combining the solutions to the sub-problems. Effective algorithms for a variety of issues are built on this methodology [5], [8], [10].

The divide and conquer strategy in three steps:

1. Break the issue up into two or fewer smaller issues,
2. Solve the subproblems recursively to overcome them,
3. Integrate the subproblems' solutions into the main problem's solutions.

Take the nxn lower triangular matrix **L** as an example:

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & 0 \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{bmatrix}$$

Divide L into chunks of size (n/2) × (n/2).

An upper- or lower-triangular matrix's inverse is also upper- or lower-triangular.

In the event when L is a lower triangular matrix divided as:

$$L = \begin{bmatrix} \mathrm{L}_{11} & 0 \\ \mathrm{L}_{21} & \mathrm{L}_{22} \end{bmatrix}$$

where:

$$\mathbf{L_{11}} = \begin{bmatrix} l_{11} & 0 & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & 0 \\ l_{(\frac{n}{2})1} & l_{(\frac{n}{2})2} & l_{(\frac{n}{2})3} & \cdots & l_{(\frac{n}{2})(\frac{n}{2})} \end{bmatrix}$$

and

$$\mathbf{L_{22}} = \begin{bmatrix} l_{(\frac{n}{2}+1)(\frac{n}{2}+1)} & 0 & 0 & 0 & 0 \\ l_{(\frac{n}{2}+2)(\frac{n}{2}+1)} & l_{(\frac{n}{2}+2)(\frac{n}{2}+2)} & 0 & 0 & 0 \\ l_{(\frac{n}{2}+3)(\frac{n}{2}+1)} & l_{(\frac{n}{2}+3)(\frac{n}{2}+2)} & l_{(\frac{n}{2}+3)(\frac{n}{2}+3)} & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & 0 \\ l_{n(\frac{n}{2}+1)} & l_{n(\frac{n}{2}+2)} & l_{n(\frac{n}{2}+3)} & \cdots & l_{nn} \end{bmatrix}$$

Next, let B be a matrix of equal dimensions that is divided as L:

$$B = \begin{bmatrix} \mathrm{B}_{11} & 0 \\ \mathrm{B}_{21} & \mathrm{B}_{22} \end{bmatrix}$$

Where:

$$\mathbf{B}_{11} = \begin{bmatrix} b_{11} & 0 & 0 & 0 & 0 \\ b_{21} & b_{22} & 0 & 0 & 0 \\ b_{31} & b_{32} & b_{33} & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & 0 \\ b_{(\frac{n}{2})1} & b_{(\frac{n}{2})2} & b_{(\frac{n}{2})3} & \cdots & b_{(\frac{n}{2})(\frac{n}{2})} \end{bmatrix}$$

$$\mathbf{B}_{21} = \begin{bmatrix} b_{(\frac{n}{2}+1)1} & b_{(\frac{n}{2}+1)2} & b_{(\frac{n}{2}+1)3} & \cdots & b_{(\frac{n}{2}+1)\frac{n}{2}} \\ b_{(\frac{n}{2}+2)1} & b_{(\frac{n}{2}+2)2} & b_{(\frac{n}{2}+2)3} & \cdots & b_{(\frac{n}{2}+2)\frac{n}{2}} \\ b_{(\frac{n}{2}+3)1} & b_{(\frac{n}{2}+3)2} & b_{(\frac{n}{2}+3)3} & \cdots & b_{(\frac{n}{2}+3)\frac{n}{2}} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ b_{n1} & b_{n2} & b_{n2} & \cdots & b_{n(\frac{n}{2})} \end{bmatrix}$$

$$\mathbf{B}_{22} = \begin{bmatrix} b_{(\frac{n}{2}+1)(\frac{n}{2}+1)} & 0 & 0 & 0 & 0 \\ b_{(\frac{n}{2}+2)(\frac{n}{2}+1)} & b_{(\frac{n}{2}+2)(\frac{n}{2}+2)} & 0 & 0 & 0 \\ b_{(\frac{n}{2}+3)(\frac{n}{2}+1)} & b_{(\frac{n}{2}+3)(\frac{n}{2}+2)} & b_{(\frac{n}{2}+3)(\frac{n}{2}+3)} & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & 0 \\ b_{n(\frac{n}{2}+1)} & b_{n(\frac{n}{2}+2)} & b_{n(\frac{n}{2}+3)} & \cdots & b_{nn} \end{bmatrix}$$

and

$$\mathbf{L}_{21} = \begin{bmatrix} l_{(\frac{n}{2}+1)1} & l_{(\frac{n}{2}+1)2} & l_{(\frac{n}{2}+1)3} & \cdots & l_{(\frac{n}{2}+1)\frac{n}{2}} \\ l_{(\frac{n}{2}+2)1} & l_{(\frac{n}{2}+2)2} & l_{(\frac{n}{2}+2)3} & \cdots & l_{(\frac{n}{2}+2)\frac{n}{2}} \\ l_{(\frac{n}{2}+3)1} & l_{(\frac{n}{2}+3)2} & l_{(\frac{n}{2}+3)3} & \cdots & l_{(\frac{n}{2}+3)\frac{n}{2}} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ l_{n1} & l_{n2} & l_{n2} & \cdots & l_{n(\frac{n}{2})} \end{bmatrix}$$

The equation $\mathbf{L}\ \mathbf{B} = \mathbf{I}$, If $\mathbf{L}$ is invertible and then $\mathbf{L}^{-1} = \mathbf{B}$
$\mathbf{L}_{11}\ \mathbf{B}_{11} = \mathbf{I}$ implies $\mathbf{B}_{11} = \mathbf{L}_{11}^{-1}$, and $\mathbf{L}_{22}\ \mathbf{B}_{22} = \mathbf{I}$
Implies $\mathbf{B}_{22} = \mathbf{L}_{22}^{-1}$, $\mathbf{L}_{21}\ \mathbf{B}_{11} + \mathbf{L}_{22}\ \mathbf{B}_{21} = \mathbf{0}$, implies that $\mathbf{B}_{21} = -\mathbf{L}_{22}^{-1}\ \mathbf{L}_{21}\ \mathbf{L}_{11}^{-1}$
Therefore, the inverse of the lower triangular matrix $\mathbf{L}$ is given by

$$L^{-1} = \begin{bmatrix} \mathrm{L}_{11}^{-1} & 0 \\ -\mathrm{L}_{22}^{-1}\mathrm{L}_{21}\mathrm{L}_{11}^{-1} & \mathrm{L}_{22}^{-1} \end{bmatrix}$$

Take the n × n upper triangular matrix U as an example:

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{bmatrix}$$

If U is a partitioned upper triangular matrix, then:

$$U = \begin{bmatrix} \mathrm{U}_{11} & \mathrm{U}_{12} \\ 0 & \mathrm{U}_{22} \end{bmatrix}$$

Next, let $\mathbf{B}$ be a matrix of equal dimensions that is divided as $\mathbf{L}$:

$$B = \begin{bmatrix} \mathrm{B}_{11} & \mathrm{B}_{12} \\ 0 & \mathrm{B}_{22} \end{bmatrix}$$

The equation $\mathbf{U}\ \mathbf{B} = \mathbf{I}$, If $\mathbf{U}$ is invertible and then $\mathbf{U}^{-1} = \mathbf{B}$
$\mathbf{U}_{11}\ \mathbf{B}_{11} = \mathbf{I}$ implies $\mathbf{B}_{11} = \mathbf{U}_{11}^{-1}$, and $\mathbf{U}_{22}\ \mathbf{B}_{22} = \mathbf{I}$, Implies $\mathbf{B}_{22} = \mathbf{U}_{22}^{-1}$,
$\mathbf{U}_{11}\ \mathbf{B}_{12} + \mathbf{U}_{12}\ \mathbf{B}_{21} = \mathbf{0}$, implies that $\mathbf{B}_{12} = -\mathbf{U}_{11}^{-1}\ \mathbf{U}_{12}\ \mathbf{U}_{22}^{-1}$
Therefore, the inverse of the lower triangular matrix $\mathbf{L}$ is given by:

$$U^{-1} = \begin{bmatrix} \mathrm{U}_{11}^{-1} & -\mathrm{U}_{11}^{-1}\mathrm{U}_{12}\mathrm{U}_{22}^{-1} \\ 0 & \mathrm{U}_{22}^{-1} \end{bmatrix}$$

3. Divide and Conquer Algorithm by LU-Factorization A^{-1}

1. Consider $n \times n$ s square matrix **A**.
2. Use the **LU** algorithm to factorize the matrix into a product of upper and lower triangular matrices $\mathbf{A} = \mathbf{L}\,\mathbf{U}$.
3. $\mathbf{A^{-1} = (LU)^{-1} = U^{-1}\,L^{-1}}$.

Computation of $\mathbf{U^{-1}}$

1. Divide **U** into four blocks $\mathbf{U_{11}, U_{12}, U_{22}}$ and a zero matrix.
2. Recursively compute inverses of $\mathbf{U_{11}}$ and $\mathbf{U_{22}}$.
3. Multiply $-\mathbf{U_{11}^{-1}}\,\mathbf{U}_{12}\,\mathbf{U_{22}^{-1}}$ and combine with $\mathbf{U_{11}^{-1}}$ and $\mathbf{U_{22}^{-1}}$ to get $\mathbf{U^{-1}}$.
4. $U^{-1} = \begin{bmatrix} \mathrm{U}_{11}^{-1} & -\mathrm{U}_{11}^{-1}\mathrm{U}_{12}\mathrm{U}_{22}^{-1} \\ 0 & \mathrm{U}_{22}^{-1} \end{bmatrix}$

Computation of $\mathbf{L^{-1}}$

1. Divide **L** into four blocks $\mathbf{L_{11}, L_{21}, L_{22}}$ and a zero matrix.
2. Recursively compute inverses of $\mathbf{L_{11}}$ and $\mathbf{L_{22}}$.
3. Multiply $-\mathbf{L_{11}^{-1}}\,\mathbf{L_{21}}\,\mathbf{L_{22}^{-1}}$ and combine with $\mathbf{L_{11}^{-1}}$ and $\mathbf{L_{22}^{-1}}$ to get $\mathbf{L^{-1}}$.
4. $L^{-1} = \begin{bmatrix} \mathrm{L}_{11}^{-1} & 0 \\ -\mathrm{L}_{22}^{-1}\mathrm{L}_{21}\mathrm{L}_{11}^{-1} & \mathrm{L}_{22}^{-1} \end{bmatrix}$

4. Numerical Examples and Results

Divide and Conquer MATLAB user subroutines are used to factor the matrix A via LU-factorization into a product of upper and lower triangular matrices, where L is a lower triangular matrix and U is an upper triangular matrix, with varying numbers of computational digits. The result is $A = LU$.

Problem (4.1)
Use the divide and conquer strategy to find the inverse matrix A when it is a structured matrix. Examine the 8x8 symmetric positive definite matrix, which is not unique:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 3 & 6 & 10 & 15 & 21 & 28 & 36 \\ 1 & 4 & 10 & 20 & 35 & 56 & 84 & 120 \\ 1 & 5 & 15 & 35 & 70 & 126 & 210 & 330 \\ 1 & 6 & 21 & 56 & 126 & 252 & 462 & 792 \\ 1 & 7 & 28 & 84 & 210 & 462 & 924 & 1716 \\ 1 & 8 & 36 & 120 & 330 & 792 & 1716 & 3432 \end{bmatrix}$$

First, we use the MATLAB software myLU11 (a) to determine the matrix A's LU factor.
We may express the matrix A as a product of two matrices, where L is a lower triangular matrix and U is an upper triangular matrix, as

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 3 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 1 & 5 & 10 & 10 & 5 & 1 & 0 & 0 \\ 1 & 6 & 15 & 20 & 15 & 6 & 1 & 0 \\ 1 & 7 & 21 & 35 & 35 & 21 & 7 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 0 & 1 & 3 & 6 & 10 & 15 & 21 \\ 0 & 0 & 0 & 1 & 4 & 10 & 20 & 35 \\ 0 & 0 & 0 & 0 & 1 & 5 & 15 & 35 \\ 0 & 0 & 0 & 0 & 0 & 1 & 6 & 21 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = L\,U$$

We now use the divide and conquer approach to calculate the inverse for the upper triangular matrix U.

$$U^{-1}=\begin{bmatrix}1&-1&-1&-1&-1&-1&-1&-1\\0&1&-2&3&-4&5&-6&7\\0&0&1&-3&6&-10&15&-21\\0&0&0&1&-4&10&-20&35\\0&0&0&0&1&-5&15&-35\\0&0&0&0&0&1&-6&21\\0&0&0&0&0&0&1&-7\\0&0&0&0&0&0&0&1\end{bmatrix}$$

Now we find the lower triangular matrix (L) by Divide and conquer method:

$$L^{-1}=\begin{bmatrix}1&0&0&0&0&0&0&0\\-1&1&0&0&0&0&0&0\\1&-2&1&0&0&0&0&0\\-1&3&-3&1&0&0&0&0\\1&-4&6&-4&1&0&0&0\\-1&5&-10&10&-5&1&0&0\\1&-6&15&-20&15&-6&1&0\\-1&7&-21&35&-35&21&-7&1\end{bmatrix}$$

Since $PA = LU$ then product both side by $(LU)^{-1}$ we have:
$(LU)^{-1}PA = (LU)^{-1}\ (LU)$
Hence $(U^{-1}\ L^{-1}\ P)\ A = I$
Therefore $A^{-1} = U^{-1}\ L^{-1}\ P$
Then

$$A^{-1}=\begin{bmatrix}8&-28&56&-70&56&-28&8&-1\\-28&140&-322&434&-364&188&-55&7\\65&-322&812&-1162&1016&-541&162&-21\\-70&434&-1162&1742&-1579&865&-265&35\\56&-364&1016&-1579&1476&-830&260&-35\\-28&188&-541&865&-830&478&-153&21\\8&-55&162&-265&260&-153&50&-7\\-1&7&-21&35&-35&21&-7&1\end{bmatrix}$$

Error in find the inverse is 0.

Problem (4.2)

We find the invers for Nearly-Singular 12 × 12 symmetric positive definite matrix "Hilbert matrix" by decompose the matrix A into triangular matrix by LU-factorization using Divide and Conquer of triangular lower and upper, and by the inverse of MATLAB inv(A), considered is n = 12 with Cond $(\mathbf{H_{12}}) = 1.7017e{+}016$, the det$(\mathbf{H_{12}})$ in single precision equal to zero. The results obtained are shown in Figs. 1 and 2.

Computational results of algorithm of Divide and Conquer for (H_{12}). The difference of res_{inv} by using the inv(A) in Matlab and using the inverse of myLU11 (A) in MATLAB are shown in Table I.

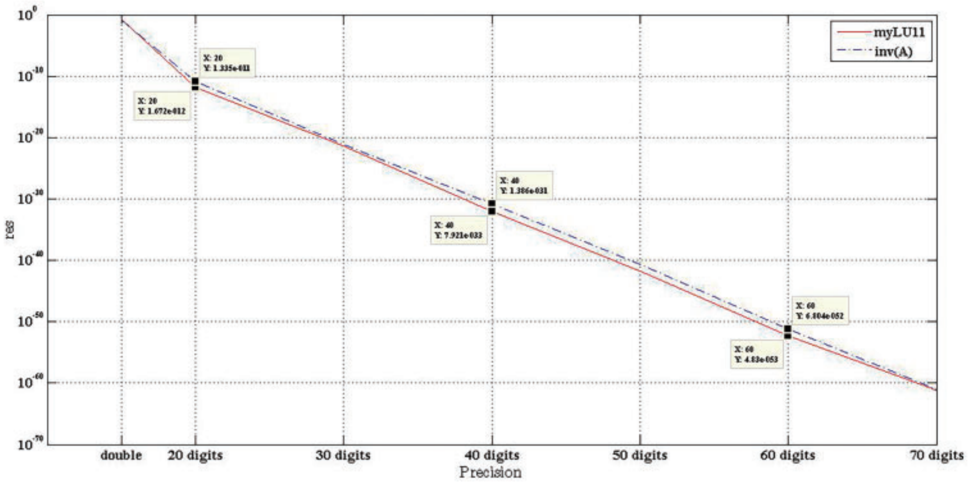

Fig. 1. Results showing that using myLU11 (A) in MATLAB is better than inv(A) in computing inverse of H_{12}.

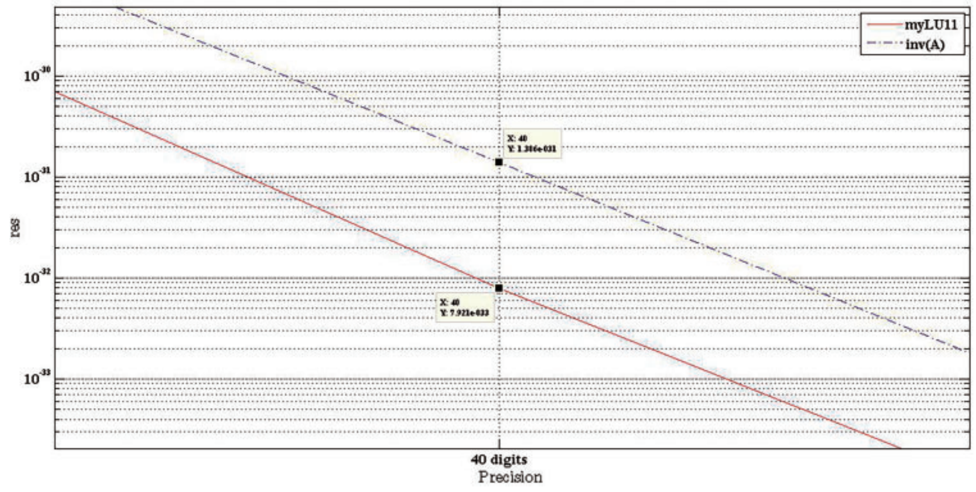

Fig. 2. The difference between the two lines.

TABLE I: Comparison of Inversion Accuray: MATLAB inv(A) vs. LU-factorization

Inverse of (A) By the program [inv(A)] in matlab		Inverse of (A) LU-factorization By the program [myLU11] in matlab	
Precision	res_{inv}	Precision	res_{inv}
Double	1.1588e-001	Double	1.6912e-001
20 digits	1.3347e-011	20 digits	1.6719e-012
30 digits	8.6546e-022	30 digits	3.7848e-022
40 digits	1.3856e-031	40 digits	7.9210e-033
50 digits	2.4168e-041	50 digits	1.8346e-042
60 digits	6.8037e-052	60 digits	4.8295e-053
70 digits	8.1132e-062	70 digits	6.5380e-062

5. Results and Conclusion

We introduced our own alternative MATLAB user-subroutine based on the Divide-and-Conquer strategy with LU factorization after recalling built-in MATLAB functions for the inversion of symmetric positive definite matrices. However, there are a few obstacles to overcome in order to fully utilize our algorithm's capabilities. To lower the round-off error, the number of digits taken into account in the computations is first raised. Secondly, the matrices are scaled before to factorization. Thirdly, there are fewer surgeries. Our user subroutine in MATLAB appears to produce far better and more accurate results than the built-in inverse functions in MATLAB. This suggests that a smart way to reduce errors and, of course, obtain a good approximation of the inverse of a nearly unique system is to use the Divide and Conquer Algorithm in conjunction with increasing the number of computational digits. The results of Rump's Gaussian elimination with double precision calculations for the same matrix in [4] appear to be less precise than the computed inverse of the matrix in example one.

Conflict of Interest

The authors declare that they do not have any conflict of interest.

References

[1] Pan VY, Zheng A, Huang X, Dias O. Newton's iteration for inversion of Cauchy-like and other structured matrices. *J Complexity*. 1997;13(1):108–24. doi: 10.1006/jcom.1997.0431.
[2] Söderström T, Stewart G. On the numerical properties of an iterative method for computing the Moore-Penrose generalized inverse. *SIAM J Numer Anal*. 1974;11(1):61–74. doi: 10.1137/0711008.
[3] Bini D, Pan V. Polynomial and matrix computations. In *Fundamental Algorithms*, vol. 1, Boston: Birkhäuser, 1994.
[4] Rump SM. Approximate inverses of almost singular matrices still contain useful information. In *Technical Report 90.1. Forschungsschwerpunkt Informations- und Kommunikationstechnik*. TU Hamburg-Harburg, 1990.
[5] Björck Å. *Numerical Mathematics and Scientific Computation*, vol. 2 and 3. 1999.
[6] Lay DC. *Linear Algebra and Its Applications*. 2nd ed. Boston: Addison-Wesley; 1997.

[7] Watkins DS. *Fundamentals of Matrix Computations*. 2nd ed. Hoboken: John Wiley & Sons; 2002.
[8] Higham NJ. *Accuracy and Stability of Numerical Algorithms*. 2nd ed. Philadelphia: SIAM; 2002.
[9] The Mathworks Inc. *MATLAB User's Guide, Version 7.9.0*. The MathWorks; 2009.
[10] Horowitz E, Sahni S, Rajasekaran S. *Computer Algorithms*. New York: W.H. Freeman; 1998.