World Journal of Advanced Engineering Technology and Sciences
eISSN: 2582-8266
Cross Ref DOI: 10.30574/wjaets
Journal homepage: https://wjaets.com/

(RESEARCH ARTICLE)

Check for updates

Automating Upper Triangular Matrix Neutralisation for Minimal Error

Henry Samambgwa * and Thomas Musora

Department of Mathematics and Statistics, School of Natural Sciences and Mathematics, Chinhoyi University of Technology, 78 Magamba Way, Off Chirundu Road, Chinhoyi, Zimbabwe.

World Journal of Advanced Engineering Technology and Sciences, 2025, 17(03), 350-356

Publication history: Received 06 November 2025; revised on 16 December 2025; accepted on 18 December 2025

Article DOI: https://doi.org/10.30574/wjaets.2025.17.3.1561

Abstract

This study develops and tests a MATLAB program for neutralising systems with upper triangular coefficient matrices. Augmented matrices for systems of linear equations are reduced to produce a neutral (identity matrix) left hand side, at which point the right hand side has the solutions for the system. A strategy for minimal error is pursued, where elimination operations must be completed in one step for each number. The strategy was expressed in pseudocode, a flowchart developed, and then the program was coded in MATLAB. The program was tested for four sample augmented matrices of varying sizes. The results demonstrated that the program was 100% accurate. The technique can be applied to any square matrix system, since any square matrix can be expressed as an upper triangular matrix. The automation of matrix operations and achievement of 100% accuracy allows for the use of the program for very large data sets, extending the potential for research with accurate finding.

Keywords: Upper Triangular Matrix; MATLAB Programing; Algorithm; Source Code.

1. Introduction

Upper triangular matrices occur often in research and applications [1]. They have useful properties that make them a powerful tool in mathematics [2]. Upper triangular matrices are closed under addition and multiplication [3]. The determinant of an upper triangular matrix is equal to its trace [4]. Upper triangular matrices are easy to work with as the systems they represent can be solved by backward substitution [5]. Expressing matrices in triangular form reduces the amount of storage space used [6]. Triangular matrices are sparse (have a large number of zero elements) matrices that occur often in application and analysis [5]. Any square matrix can be expressed in Hermitic normal form [7].

Continuous improvement of computational algorithms is essential for improving computational efficiency in speed, storage and accuracy [8]. Immense research has been carried out to optimise matrix operations. Mouha et al [9] compared matrix multiplication algorithms on computational complexity, prioritising speed and memory usage. Ray [10] focused on algorithm optimisation for scalability to large data sets. Herrero [11] optimised algorithms to reduce computer processor overheads. Dumasa et al [12] sought to optimise matrix algorithms for computational speed. Alman and Yu [13] developed algorithm search methods to reduce the magnitude of the leading constant in matrix multiplication. Dehghankar et al [14] optimised matrix processing algorithms for neural network computations. Their method improved matrix multiplication speeds by up to 29 times and used six times less memory.

Matrices are critical to artificial intelligence research [15]. Neural networks are trained from empirical data [16]. After training, their characteristics are stored as weights and biases. Matrices are a convenient way to store characteristics of neural network models in machine learning, a subfield of artificial intelligence [17]. Numerous methods have been developed to improve matrix algorithms, with far reaching impact in research and applications [18]. Developments in upper triangular matrix operations have a widespread impact in mathematical analysis and application. Upper

* Corresponding author: Henry Samambgwa. Email: henrysamambgwa@gmail.com

triangular matrix methods can be extended to all square matrices since any square matrix can be expressed as an upper triangular matrix [19].

Aim

To develop an algorithm and programming code that neutralises upper triangular matrices with minimal error.

Objectives

1. Develop pseudocode for neutralising any upper triangular matrix with minimal error.
2. Develop the flow chart for the upper triangular matrix neutralisation algorithm.
3. Develop MATLAB programming code for the upper triangular matrix neutralisation algorithm.

1.1. Target programing environment

This study sought to develop an upper triangular matrix neutralisation program that runs in the MATLAB 22 [20] programing and analysis environment.

2. Methodology

2.1. Pseudocode

Consider a system of n linear equations expressed in an upper triangular coefficient matrix. **Figure 1** illustrates the augmented matrix for such a system. In solving the system computationally, considerations are made to minimise errors.

$$\left[\begin{array}{ccccccc|c} a_{11} & a_{12} & a_{13} & . & . & . & a_{1n} & r_1 \\ 0 & a_{22} & a_{23} & . & . & . & a_{2n} & r_2 \\ 0 & 0 & a_{33} & . & . & . & a_{3n} & r_3 \\ 0 & 0 & 0 & . & . & . & . & . \\ 0 & 0 & 0 & 0 & . & . & . & . \\ 0 & 0 & 0 & 0 & 0 & . & . & . \\ 0 & 0 & 0 & 0 & 0 & 0 & a_{nn} & r_n \end{array}\right]$$

Figure 1 Augmented upper triangular coefficient matrix

The matrix is said to be neutralised if the left side is the nxn identity matrix. The strategy is to minimise the number of operations necessary to neutralise the matrix. Thus each eliminated term must be eliminated in a single step to minimise concatenation and round off errors. The algorithm is thus designed such that on each elimination, a single row operation is performed with appropriate parameters to eliminate the target element. The steps are outlined in the Preamble, Figure 2 and the pseudo code in Figure 3.

Preamble: Define the pivot of a row as the element, a_p, on the main diagonal when it is the only non-zero element in the row.

Define the targets, the elements in the column above the pivot.

Define the victim, a_v, the target to be eliminated at a particular operation step.

R_v is the row with the victim.

R_p is the row with the pivot.

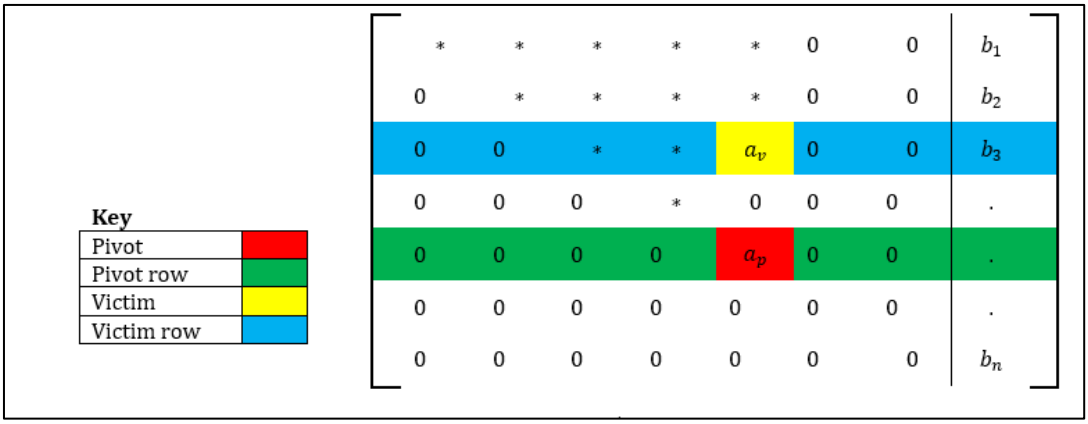

Figure 2 Augmented matrix at a typical step in eliminating targets.

The Pseudocode in **Figure 3** outlines the steps to eliminate targets, one by one while changing the pivot from the bottom row to the top.

Step 1: Set bottom row as pivot row.

Step 2: Repeat sub-steps 1-3 until top row = pivot row.

Sub-step 1: Set the row above the pivot row as the victim row.

Sub-step 2: Repeat sub-sub-steps 1 and 2 until victim row = top row.

Sub-sub-step 1: Perform the operation $R_v \rightarrow R_v - \frac{a_v}{a_p} R_p$.

Sub-sub-step 2: Release the current victim row and set the next row above as the victim row.

Sub-step 3: Release the current pivot row and set the next row above as the pivot row.

Figure 3 Pseudo code for upper triangular matrix neutralisation.

Figure 5 shows the flowchart of the upper triangular matrix neutralisation algorithm. The algorithm is composed of two main loops. The major loop terminates when the pivot is at the top row, since no further elimination will be possible. The minor loop terminates when the row operation completes with the victim at the top row.

2.2. MATLAB source code

Figure 4 shows the programing code that implements upper triangular matrix neutralisation. Comments are included after the '%' symbol at each line to help explain the code.

```
%User must have saved the augmented matrix as M
s=size(M);   % s is a two dimensional vector giving the number rows and columns
respectively
n=s(1);      % n is set to the first value of s, number of rows

j=0;         % here j is the row counter, causing the major loop to happen as many
times as
while j<n-1  % the number of rows minus 1
    k=n-j;   % k, set to n-j, is the pivot row (and column) index
    i=1;     % here i is the row counter, used for looping through victim rows as
many times
    while i<k      % as the pivot index minus 1 for each pivot.
        pivot=M(k,k);     % pivot value is set at each iteration
        victim=M(k-i,k);  % victim value is set at each iteration
        M(k-i,:)=M(k-i,:)-(victim/pivot)*M(k,:);    %the elimination row operation
is done here
        i=i+1;            %victim row counter is incremented
    end                   % minor loop ends
j=j+1;                    %pivot row counter is incremented
end                       %major loop ends
i=n;                      % at the end of the algorithm
while i>0                 % each pivot row is divided by
    pivot=M(i,i);         % by a value equal to the pivot
    M(i,:)=M(i,:)/pivot;  % so that the result is an
    i=i-1;                % identity matrix.
end                       % Hence the upper triangular matrix is neutralised
M                         % The resulting matrix is displayed
```

Figure 4 MATLAB source code for the upper triangular matrix neutralisation algorithm**.**

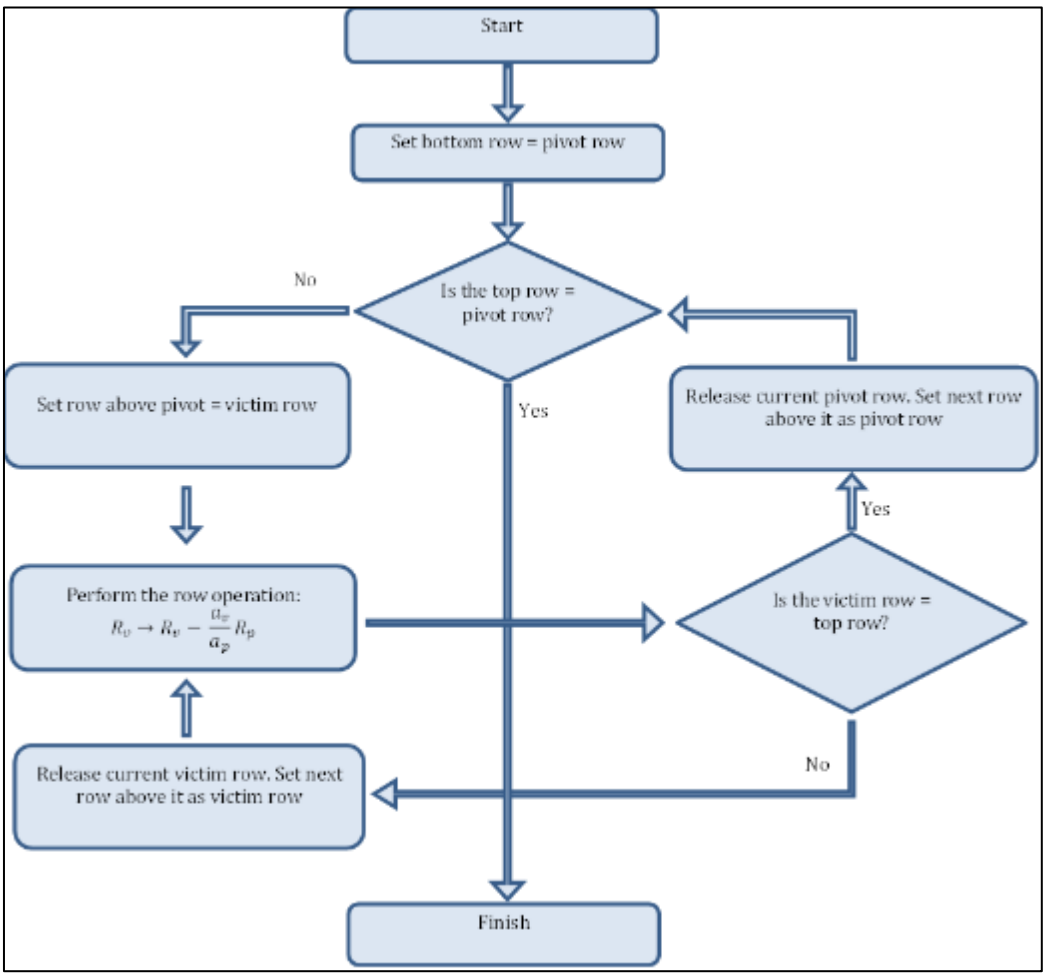

Figure 5 Flow chart of upper triangular matrix neutralisation algorithm

3. Analysis

The upper triangular matrix neutralisation program was tested for four cases:

- A system with a 3 by 3 upper triangular matrix,
- A system with a 5 by 5 upper triangular matrix,
- A system with an 8 by 8 upper triangular matrix, and
- A system with a 10 by 10 upper triangular matrix.

Table 1 shows the four sample augmented upper triangular matrices (on the left) and the neutralised matrices (one the right) that resulted when the neutralisation program in Figure 4 was used.

Table 1 Sample augmented matrices (left) and the corresponding neutralised matrices (right).

Augmented Matrix	**Neutralised Matrix**
$\left[\begin{array}{ccc\|c} 3 & 5 & 2 & 36 \\ 0 & 1 & 4 & 15 \\ 0 & 0 & -2 & -6 \end{array}\right]$	$\left[\begin{array}{ccc\|c} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 3 \end{array}\right]$
$\left[\begin{array}{ccccc\|c} 2 & -2 & 8 & 1 & 5 & 53 \\ 0 & -3 & 9 & 0 & 2 & 30 \\ 0 & 0 & 1 & 6 & 4 & 29 \\ 0 & 0 & 0 & 4 & 7 & 29 \\ 0 & 0 & 0 & 0 & 2 & 6 \end{array}\right]$	$\left[\begin{array}{ccccc\|c} 1 & 0 & 0 & 0 & 0 & 5 \\ 0 & 1 & 0 & 0 & 0 & 7 \\ 0 & 0 & 1 & 0 & 0 & 5 \\ 0 & 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 & 3 \end{array}\right]$
$\left[\begin{array}{cccccccc\|c} -4 & 3 & 0 & 3 & 2 & 1 & 4 & 5 & 64 \\ 0 & 3 & 7 & 3 & 7 & 3 & 2 & 1 & 174 \\ 0 & 0 & 2 & 1 & 8 & 1 & 8 & 0 & 138 \\ 0 & 0 & 0 & 8 & 5 & 9 & 10 & 9 & 225 \\ 0 & 0 & 0 & 0 & 5 & 7 & 8 & 9 & 151 \\ 0 & 0 & 0 & 0 & 0 & 3 & 5 & 2 & 57 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8 & 2 & 60 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 4 \end{array}\right]$	$\left[\begin{array}{cccccccc\|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 7 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 7 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \end{array}\right]$
$\left[\begin{array}{cccccccccc\|c} 4 & 5 & 8 & 8 & 5 & 2 & 8 & 7 & 4 & 5 & 129 \\ 0 & 6 & 1 & 9 & 5 & 10 & 1 & 7 & 10 & 3 & 210 \\ 0 & 0 & 9 & 6 & 7 & 4 & 2 & 4 & 7 & 2 & 111 \\ 0 & 0 & 0 & 6 & 9 & 7 & 5 & 4 & 3 & 2 & 138 \\ 0 & 0 & 0 & 0 & 6 & 7 & 0 & 1 & 1 & 5 & 86 \\ 0 & 0 & 0 & 0 & 0 & 9 & 6 & 9 & 3 & 4 & 123 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 4 & 2 & 1 & 49 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 5 & 2 & 71 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7 & 2 & 69 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 12 \end{array}\right]$	$\left[\begin{array}{cccccccccc\|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 \end{array}\right]$

4. Discussion

When the left hand sides (LHS) of the augmented matrices were multiplied by the right hand sides (RHS) of the neutralised matrices, the results were equal to the RHS of the original augmented matrices. This indicated that indeed the values on the RHS of the neutralised matrices were suitable solutions for the systems. The upper triangular matrix neutralisation program generated the correct values on the right hand side (RHS) of the neutralised matrices.

The matrix neutralisation program developed in this study thus demonstrated 100% accuracy. It can be used for arbitrary sizes of upper triangular matrices.

5. Conclusion

This research successfully developed and tested a 100% accurate upper triangular matrix neutralisation program. This can be used in the numerous occurrences of upper triangular systems in research and application. Automation of essential matrix operations allows for quicker, more accurate solutions to mathematical problems. The level of accuracy and speed allow for analysis of very large data sets to produce reliable results. Further research may be conducted to automate useful matrix operations such finding eigenvalues and eigenvectors.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Sarajlija, N. (2024). Upper Triangular Operator Matrices and Stability of Their Various Spectra. Results in Mathematics. Springer.

[2] Vel, O. D., Mallet, Y., Coomans, D. (2000). Data handling in science and technology. Elsevier.

[3] Schneider, P. J., Eberly, D. H. (2003). Geometric tools for computer graphics. Elsevier.

[4] Britanac, V., Yip, P. C., Rao, K. R. (2007). Discrete cosine and sine transforms. Elsevier.

[5] Zalizniak, V. (2008). Essentials of scientific computing. Horwood Publishing Limited.

[6] Pan, V. (2003). Encyclopedi of Physical Science and Technology, third edition. Elsevier Science Limited.

[7] Chen, Y., Huizhou, Zhao, X., Nanchang, Liu, Z. (2015). On upper triangle non-negative matrices. Czechoslovak Mathematical Journal.

[8] Fawzi, A., Balog, M., Huang, A., Hubert, T., Romera-Paredes, B., Barekatain, M., Novikov, A., Ruiz, F. J. R., Schrittwieser, J., Swirszcz, G., Silver, D., Hassabis, D., Kohli, P. (2022). Discovering faster matrix multiplication algorithms with reinforcement learning. Springer Nature.

[9] Mouha, K., Faiz, H., Bourhnane, S. (2023). Large matrix multiplication algorithms: Analysis and comparison. The 7th International conference on Algorithms, Computing and Systems. ICAS.

[10] Ray, A. (2024). Sublinear algorithms for matrices: Theory and applications. Robert and Donna Manning College of Information and Computer Sciences. University of Massachusetts Amherst.

[11] Herrero, J. R. (2006). A Framework for Efficient Execution of Matrix Computations. Departament D'Arquitectura de Computadors. Universitat Polit` Ecnica De Catalunya.

[12] Dumasa, J. G., Perneta, C., Sedoglavic, A. (2025). Towards automated generation of fast and accurate algorithms for recursive matrix multiplication. Journal of Symbolic Computation.

[13] Alman, J., Yu, H. (2025). Improving the Leading Constant of Matrix Multiplication. Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).

[14] Dehghankar, M., Erfanian, M., Asudeh, A. (2025). An Efficient Matrix Multiplication Algorithm for Accelerating Inference in Binary and Ternary Neural Networks. International Conference on Machine Learning (ICML).

[15] Balappa, R. D., Rajput, G. K. (2024). The Role and Application of Matrices in Artificial Intelligence: Foundations, Methods, and Advancements. Volume. 9 Issue.11, November-2024 International Journal of Innovative Science and Research Technology.

[16] Musora, T. (2023). Application of Panel Data Analysis and Modelling to Economic Data: A Case of Determinants of Economic Growth for SADC and Zimbabwe. Chinhoyi University of Technology.

[17] Samambgwa, H., Musora, T., Kamusha, J. (2025). Deriving mathematical models from neural networks: A method for deducing individual effects of factors on a response variable. World Journal for Advanced Engineering Technology and Sciences.

[18] Liu, J. (2023). Analysis for faster matrix multiplication. Proceedings of the 2023 International Conference on Mathematical Physics and Computational Simulation. UC Berkeley.

[19] Pai, D., Arya, R., Sahai, A. (2023). Designing Information Systems and Devices II. Note 14: Upper Triangularization, Spectral Theorem. UC Berkeley.

[20] MATLAB Online environment. (2025). Accessed at: matlab.mathworks.com