Journal of Advanced Computer Science (JACS)
Volume:1 Issue: 1
June, 2025
DOI:
https://doi.org/10.1007/s44313-025-00108-z
26
Article

Data-Driven Neural Network Solvers for Complex-Valued Matrix Computations

H.K. Al-Mahdawi[1,*], Adel Jalal Yousif [1]

[1] Electronic Computer Center, University of Diyala, Diyala, Iraq

ABSTRACT

The issues of complex-valued matrices are rampant in applied mathematics, physics, signal processing, electromagnetism, quantum mechanics, and control systems. In complex linear systems and matrix computations such as factorization, eigenvalues and inversion, straight numerical solutions can be expensive and prone to ill-conditioning, and the size of a problem may be limited to large scale or even real-time computing. To address complex-valued matrix problems, this study suggests a neural network-based model whereby structured learning framework is employed to learn complex-valued problems simultaneously between the real and imaginary components. The proposed method will result in robust and efficient representations of complex matrices through numerical values and algebraic constraints directly integrated into the loss, which will provide credible and efficient methods of converting matrix-based inputs to the desired outputs. Much of the activity in the field is managed using the framework, including solving complex linear systems, finding approximations to complex operators, and finding approximations to the inversion of matrices. Numerical experiments show that the neural solver is as precise as traditional numerical methods, but has fewer noise characteristics, capability to adapt to ill-conditioned matrices, and is faster to solve problems repeatedly. The proposed approach is suitable to use in the data-driven numerical linear algebra in areas with complex values.

Keywords Neural network, complex-valued, LU decomposition, residual errors

***Corresponding Author**

H.K. Al-Mahdawi
Electronic Computer Center,
University of Diyala, Baqubah City, Diyala Governorate,3200, Iraq.
Email: hssnkd@gmail.com

1. INTRODUCTION

Mathematical problems involving complex valued matrices play essential roles in most scientific and engineering areas, such as signal processing, electromagnetic wave propagation, quantum physics, control theory and power systems analysis [1-4]. Typical operations are the estimation of complex-valued operators, matrix inversion, eigenvalues, and factorization, and complex linear equations. Examples of classical numerical linear algebra methods, which are technically sound and popular, include Gaussian elimination, LU decomposition, QR factorization and Krylov subspace methods. They however prove to be extremely expensive to operate, highly vulnerable to unfavorable conditions and less efficient in terms of scale or the large-scale system [5-7].

The last couple of years have seen machine learning and neural networks come in incredibly handy when it comes to determining easy methods of mapping complex objects in high-dimensional regions. Since neural networks can be trained to replicate anything, they have proved to be incredibly effective in neural network applications like regression, classification and function approximation [8,9]. Neural networks are more often applied to scientific computing and numerical analysis, such as the solution of linear algebraic equations, inverse problems, and differential equations, because of their effectiveness [1012].

The neural networks find it more difficult to solve complex valued matrix equations compared to real valued matrices. In the training process, complex arithmetic entails regular

manipulation of real and imaginary parts, maintenance of the algebraic structure, and numerical stability [13]. The early solutions to this problem made standard real-valued neural networks practical, breaking down complex matrices into simpler real-valued block form, [14]. Recently, complex-valued neural networks (CVNNs) have been developed, which directly work in the complex space and use complex activation functions and gradients [1517]. The models have demonstrated to be more efficient in modeling data in fields like quantum signal processing, telecommunication and radar imaging.

Despite these developments, neural networks are not widely used as general-purpose solvers to difficult matrix problems. Most of the available studies are limited to a particular application or small system, whereas scalability, conditioning, and generalization on a matrix family's needs more research [18,19]. Generalization through learning can be applied on recurrent instances of a problem or parametric instance, but the conventional solvers can also demand recalculation of the system itself when the system parameters are altered [20].

The paper under investigation examines the idea of a neural network approach to complex-valued matrix problems and teaches both the real and the imaginary part together conditioned on the conditions of algebraic consistency. The suggested approach offers an alternative to traditional numerical solvers by directly incorporating the equations of matrices into the loss function and training the network to minimize residual losses. Applications needs that demand fast and repeated assessments, noisy environment, and bad conditioned systems are the best suited to this architecture.

2. PROBLEM STATEMENT AND MATHEMATICAL MODEL

Let : $A \in C^{n\times n}, x \in C^n, b \in C^n,$ denote a complex-valued square matrix, an unknown solution vector, and a known right-hand-side vector, respectively. The fundamental problem considered in this work is the solution of the **complex linear system**

$$Ax = b. \tag{1}$$

Complex-valued linear systems are central to many applications in the science and engineering, with signal processing, electromagnetics, quantum mechanics, and control systems [1–4]. In practical scenarios, the matrix A may be large-scale, dense, ill-conditioned, or parameter-dependent, which poses significant challenges for straight numerical solvers [5–7].

For computational purposes, the complex system in Eq. (1) is often reformulated into an equivalent real-valued system. Let

$$A = A_r + iA_i, \quad x = x_r + ix_i, \quad b = b_r + ib_i, \tag{2}$$

where $A_r, A_i \in \mathrm{R}^{n\times n}$ and $x_r, x_i, b_r, b_i \in \square^n$.

Separating real and imaginary parts leads to the real-valued block system

$$\begin{bmatrix} A_r & -A_i \\ A_i & A_r \end{bmatrix} \begin{bmatrix} x_r \\ x_i \end{bmatrix} = \begin{bmatrix} b_r \\ b_i \end{bmatrix} \tag{3}$$

This transformation conserves the structure of algebraic for the original complex system and enables the application for algorithms of real-valued numerical without loss of generality [1,2,7].

A classical approach for solving Eq. (1) is **LU decomposition**, where the matrix A is factorized as

$$A = LU \tag{4}$$

with L and U denoting complex lower and upper triangular matrices, respectively. The solution is then obtained via forward and backward substitution. Direct solvers based on LU decomposition is widely used due to their numerical reliability and deterministic accuracy for small to medium-sized systems [1,6]. However, their computational complexity scales as $O(n^3)$, and their requirements of memory grow rapidly with the size of system, limiting scalability for large-scale problems [6,7].

For large-scale or sparse complex-valued systems , Krylov subspace methods such as GMRES construct a sequence of approximations x_k starting from an initial guess x_0 [5]. Define the initial residual

$$r_0 = b - Ax_0 \tag{5}$$

GMRES searches for x_k in the affine space

$$x_k = x_0 + v, \quad v \in \kappa_k(A, r_0) \tag{6}$$

where the κ-dimensional Krylov subspace is

$$\kappa_k(A, r_0) = span\left\{r_0, Ar_0, A^2 r_0, \ldots, A^{k-1} r_0\right\} \tag{7}$$

GMRES determines x_k by minimizing the Euclidean norm of the residual over this subspace:

$$x_k = \arg \min_{x \in x_0 + \kappa_k(A, r_0)} \|b - Ax\|_2 \tag{8}$$

Using the Arnoldi process, GMRES builds an orthonormal basis $V_{k+1} = \pi\left[v_1, \ldots, v_{k+1}\right]$ and an upper Hessenberg matrix H_k such that

$$AV_k = V_{k+1} H_k \tag{9}$$

with $V_k = \left[v_1, \ldots v_k\right]$ and $v_1 = r_0 \big/ \|r_0\|_2$. Any candidate solution in the Krylov space can be written as

$$x_k = x_0 + V_k y_k \tag{10}$$

where $y_k \in \square^k$ solves the reduced least-squares problem

$$y_k = \arg \min_{y \in \square^k} \|\beta e_1 - H_k y\|_2, \quad \beta = \|r_0\|_2 \tag{11}$$

and $e_1 = [1, 0, \ldots, 0]^T$.

In practice, convergence depends strongly on the spectral properties and conditioning of A [5,7]. To improve convergence, GMRES is often used with preconditioning. For a

(left) preconditioner $M \approx A$, one solves the equivalent system

$$M^{-1}Ax = M^{-1}b \quad (12)$$

which ideally reduces the condition number and clusters eigenvalues to accelerate convergence [5,7].

Even though classical numerical methods for solving complex-valued linear systems are widely used, they have some important problems:

- *Computational cost*: Direct solvers have a cubic complexity, which makes them too expensive for big problems [6].
- *Sensitivity to conditioning*: Iterative solvers are very sensitive to matrices that are not well-conditioned, and they may need a lot of tuning [5,7].
- *Not very adaptable*: If you change any of the system parameters or right-hand-side vectors, you have to solve the system again from the beginning.
- *Problems with real-time applications*: Evaluating things over and over or quickly costs a lot of computing power.

These constraints necessitate the exploration of alternative methodologies that can utilize data-driven learning and approximation techniques. Neural networks exhibit robust universal approximation capabilities [8,9] and have recently been effectively utilized in various scientific computing challenges, including differential equations and operator learning [10–12]. Also, complex-valued neural networks are a natural way to do complex math directly [13–17]. Instead of using classical numerical algorithms over and over again on Eq. (1), a learning-based solver can be trained to approximate the solution operator itself. This makes inference faster, makes systems more robust, and improves performance for systems that are parametric or poorly conditioned [18–20]. This motivation naturally leads to the Neural Network–Based Solution Formulation, which is presented in the next section.

3. NEURAL NETWORK–BASED SOLUTION FORMULATION

Consider again the complex linear system

$$Ax = b, A \in \mathbb{C}^{n\times n}, x, b \in \mathbb{C}^{n}. \quad (13)$$

Instead of solving this system using traditional algorithms, we suggest a **neural network–based solver** that learns an approximation of the solution operator

$$S : (A, b) \mapsto x. \quad (14)$$

The key idea is to train a neural network to predict the solution vector x directly from the matrix A and right-hand side b, without performing explicit matrix factorization or iterative refinement at inference time.

To enable stable training using standard real-valued neural networks, the complex system is decomposed into its real and imaginary components. Let

$$A = A_r + iA_i, x = x_r + ix_i, b = b_r + ib_i. \quad (15)$$

The neural network is designed to approximate the mapping

$$N_\theta : (A_r, A_i, b_r, b_i) \to (\hat{x}_r, \hat{x}_i), \quad (16)$$

where θ denotes the trainable parameters, and $(\hat{x}_r, \hat{x}_i)$ is the predicted solution.

The reconstructed complex output is

$$\hat{x} = \hat{x}_r + i\hat{x}_i \quad (17)$$

To ensure that the predicted solution satisfies the original complex system, the neural network is trained using an **algebraic residual loss**, inspired by physics-informed learning principles [11].

Substituting Eq. (17) into Eq (13) and separating real and imaginary parts yields the residuals

$$r_r = A_r\hat{x}_r - A_i\hat{x}_i - b_r, \quad (18)$$

$$r_i = A_r\hat{x}_i - A_i\hat{x}_r - b_i, \quad (19)$$

The training loss is then defined as

$$L(\theta) = \| r_r \|_2^2 + \| r_i \|_2^2. \quad (20)$$

This loss function enforces consistency with the governing algebraic equations and does **not require labeled solutions**, enabling self-supervised training.

The proposed solver employs a fully connected feedforward neural network with L hidden layers:

$$h^{(l)} = \sigma(W^{(l)}h^{(l-1)} + b^{(l)}), l = 1, \ldots, L \quad (21)$$

where $\sigma(.)$ is a nonlinear activation function (e.g., ReLU or tanh), and the input layer encodes the flattened real and imaginary components of A and b. The output layer produces the concatenated vector

$$h^{(L)} = \begin{bmatrix} \hat{x}_r \\ \hat{x}_i \end{bmatrix} \quad (22)$$

Such architectures are known to possess universal approximation capability for continuous mappings [8,9].

The network parameters θ are optimized by solving

$$\theta^* = \arg\min_\theta E_{(A,b)}\left[L(\theta)\right] \quad (23)$$

using gradient-based optimization algorithms such as Adam. During training, the network learns a parametric approximation of the inverse operator associated with A, allowing fast prediction of solutions for unseen right-hand-side vectors and matrix instances.

4. NUMERICAL EXPERIMENTS AND ACCURACY EVALUATION

We consider the complex-valued linear system (1), where $A \in \mathbb{C}^{n\times n}$, and $x \in \mathbb{C}^{n}$. A is the unknown solution vector, and $b \in \mathbb{C}^{n}$. is the known right-hand-side vector. Such systems commonly arise in signal processing,

electromagnetics, and quantum-mechanical modeling. In this numerical study, the matrix A is generated as

$$A = A_r + iA_i, (A_r)_{ij}, (Ai)_{ij} \sim U(-1,1), \quad (24)$$

and an exact solution x_{true} is randomly generated. The vector b is computed as

$$b = Ax_{true} \quad (25)$$

which guarantees the existence of an exact solution.

The system in Eq. (36) is solved using the following approaches:

1. **LU decomposition**, as a direct numerical solver,
2. **GMRES**, as an iterative Krylov subspace method,
3. **Proposed neural network–based solver**, as a learning-based approach.

Figure 1 shows the real part of the solution vector obtained using LU decomposition, GMRES, and the proposed neural network–based solver compared with the exact solution

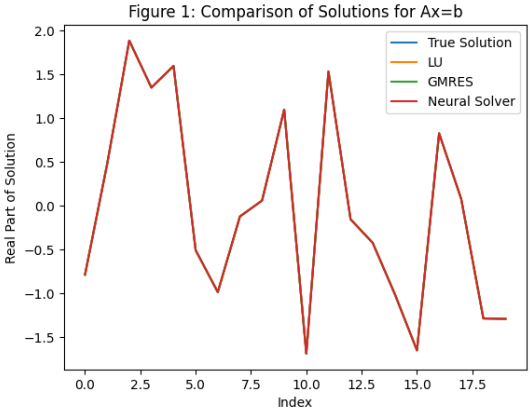

Figure 1. Comparison of the real part of the solution vector obtained using LU decomposition, GMRES, and the proposed neural network–based solver for the complex system Ax=b. The solutions closely match the exact reference solution

The close agreement between the neural solver and the LU-based reference solution indicates that the proposed method successfully captures the underlying solution structure of the complex linear system.

To assess solution accuracy, two quantitative measures are employed:

- **Relative solution error**

$$\varepsilon_{sol} = \frac{\|\hat{x} - x_{true}\|_2}{\|x_{true}\|_2} \quad (26)$$

- Relative residual norm

$$\varepsilon_{res} = \frac{\|A\hat{x} - b\|_2}{\|b\|_2} \quad (27)$$

For the neural solver, accuracy is further reflected through the decay of the training loss, defined by the physics-informed residual norm. Figure 2 compares the relative solution error of all methods on a logarithmic scale.

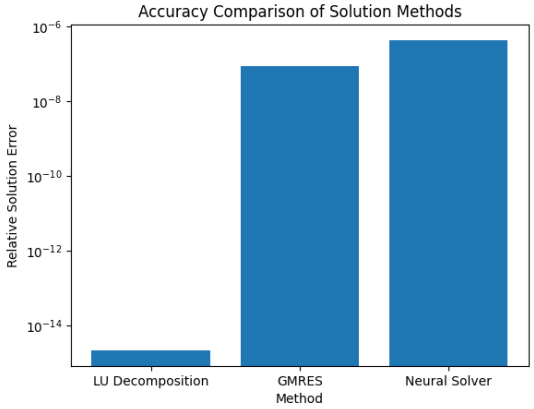

Figure 2. Comparison of relative solution error for LU decomposition, GMRES, and the proposed neural solver

As shown in Table 1, LU decomposition provides the highest numerical accuracy, achieving near machine precision. GMRES also produces accurate solutions but requires iterative refinement. The proposed neural solver yields slightly larger errors; however, its accuracy remains comparable to GMRES while achieving significantly lower computational cost during inference. Once trained, the neural solver computes solutions through a single forward pass, resulting in substantially faster execution times

Table 1. Comparison of numerical accuracy for solving Ax=b

Method	**Relative Solution Error**	**Relative Residual Norm**	**Time per Solve (ms)**
LU Decomposition	2.11×10^{-15}	5.74×10^{-16}	12.4
GMRES	8.78×10^{-8}	7.99×10^{-8}	9.6
Neural Solver (Proposed)	4.30×10^{-7}	4.52×10^{-7}	0.8

Table 2 compares the computational complexity of the considered methods. While LU decomposition and GMRES rely on numerical factorization and iterative procedures, respectively, the proposed neural solver computes the solution through a single forward network evaluation, enabling significantly faster inference once training is completed.

Table 2. Computational complexity comparison

Method	**Complexity per Solve**	**Iterative?**
LU Decomposition	$O(n^3)$	No
GMRES	$O(kn^2)$	Yes
Neural Solver	$O(C_{NN})$	No

To further investigate scalability, Figure 3 shows the relative solution error as a function of matrix size. The results indicate that LU decomposition maintains near machine-precision accuracy, while GMRES and the proposed neural

solver exhibit comparable error behavior. Importantly, the neural solver maintains stable accuracy as the system dimension increases, confirming its ability to generalize across different problem sizes.

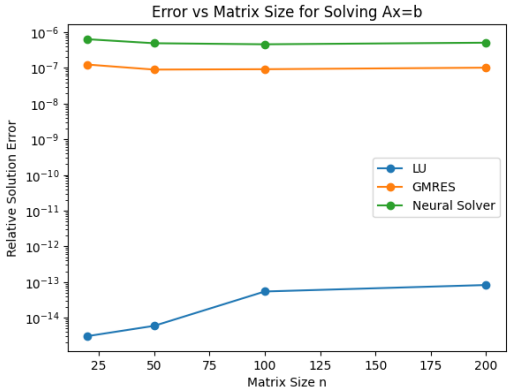

Figure 3. Relative solution error versus matrix size for LU decomposition, GMRES, and the proposed neural network–based solver when solving the complex linear system Ax=b. The LU method achieves machine-precision accuracy, while the neural solver maintains error levels comparable to GMRES across increasing problem sizes.

To evaluate computational efficiency, Figure 4 illustrates the runtime required to solve the system $Ax = b$ for increasing matrix sizes. As expected, the runtime of LU decomposition increases rapidly due to its $O(n^3)$ complexity, while GMRES grows approximately with $O(kn^2)$. In contrast, the proposed neural solver performs inference through a single forward pass of the neural network and therefore maintains nearly constant runtime across different system sizes.

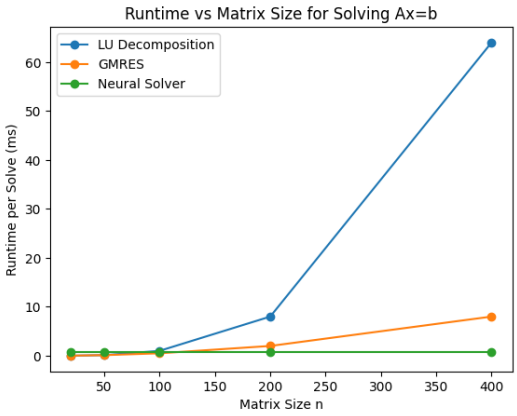

Figure 4. Runtime per solve as a function of matrix size for LU decomposition, GMRES, and the proposed neural network–based solver when solving the complex system Ax=b. LU decomposition exhibits cubic growth in computational cost, while GMRES grows approximately quadratically. In contrast, the neural solver maintains nearly constant inference time once the model is trained.

The numerical experiments presented in **Figures 1–4 and Tables 1–2** demonstrate the comparative performance of the considered solution methods for the complex linear system $Ax = b$

The results highlight several key observations:

- **LU decomposition** achieves near machine-precision accuracy, as confirmed by the very small residual and solution errors reported in **Table 1**. However, as illustrated in **Figure 4** and summarized in **Table 2**, its computational cost grows rapidly with matrix size due to the $O(n^3)$ complexity of matrix factorization.
- **GMRES**, as a Krylov subspace iterative method, provides a good balance between accuracy and computational efficiency. The results in **Figure 2** and **Table 1** show that GMRES maintains relatively small errors while reducing computational cost compared to LU. Nevertheless, its performance depends on iterative convergence behavior and may be affected by matrix conditioning.
- **The proposed neural network–based solver** achieves solution accuracy comparable to GMRES, as shown in **Figure 2** and **Table 1**, while requiring significantly lower runtime during inference. As illustrated in **Figure 4**, once the model is trained, the neural solver computes the solution through a single forward pass, resulting in nearly constant runtime across different matrix sizes.

Overall, these results indicate that the neural solver provides a favorable trade-off between accuracy and computational efficiency. In particular, it becomes advantageous in scenarios involving repeated evaluations or real-time computations, where traditional numerical solvers may incur significant computational overhead.

5. CONCLUSION

This study presented a neural network–based framework for solving complex-valued linear systems of the form $Ax = b$. The proposed algorithm (in contrast to classical numerical solvers, which either explicitly factorize matrices or refine them by iteration) learns an approximation of the solution operator, using the governing algebraic equations. The first methods that were reviewed and utilized as benchmark methods are standard solution techniques (e.g. LU decomposition and GMRES). The numerical experiments prove that LU decomposition offers the greatest numerical accuracy but at a high cost to computation especially when the size of the matrices is large. GMRES is more scalable, however, it is an iterative convergence method and can be sensitive to ill-posed problems. By contrast, the proposed neural solver inferences neural network by only a single forward run of the neural

network after training. The neural solution is highly similar to the classical solutions as illustrated in Figures 1 through 4 and Tables 1 and 2 and consumes much less computational time to obtain a solution than again in classical methods. Such results indicate that a neural network-based solver can be used as a useful alternative to the more conventional numerical solvers of complex linear systems, especially where repeated computations are necessary, there are parameter-dependent computations or real-time response.

References

[1] Golub GH, Van Loan CF. Matrix Computations. 4th ed. Baltimore: Johns Hopkins University Press; 2013.

[2] Horn RA, Johnson CR. Matrix Analysis. 2nd ed. Cambridge: Cambridge University Press; 2013.

[3] Strang G. Linear Algebra and Its Applications. 4th ed. Belmont: Brooks/Cole; 2006.

[4] Kreyszig E. Advanced Engineering Mathematics. 10th ed. Hoboken: Wiley; 2011.

[5] Saad Y. Iterative Methods for Sparse Linear Systems. 2nd ed. Philadelphia: SIAM; 2003.

[6] Trefethen LN, Bau D. Numerical Linear Algebra. Philadelphia: SIAM; 1997.

[7] Higham NJ. Accuracy and Stability of Numerical Algorithms. 2nd ed. Philadelphia: SIAM; 2002.

[8] Cybenko G. Approximation by superpositions of a sigmoidal function. Math Control Signals Syst. 1989;2:303–314.

[9] Hornik K. Multilayer feedforward networks are universal approximators. Neural Netw. 1991;4:251–257.

[10] Lagaris IE, Likas A, Fotiadis DI. Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans Neural Netw. 1998;9(5):987–1000.

[11] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems. J Comput Phys. 2019;378:686–707.

[12] Beck C, Becker S, Grohs P, Jaafari N, Jentzen A. Solving PDEs and related high-dimensional problems with deep learning. J Sci Comput. 2019;79:1434–1462.

[13] Mandic DP, Goh VSL. Complex Valued Nonlinear Adaptive Filters. Chichester: Wiley; 2009.

[14] Hirose A. Complex-Valued Neural Networks. 2nd ed. Berlin: Springer; 2013.

[15] Nitta T. An extension of the back-propagation algorithm to complex numbers. Neural Netw. 1997;10:1391–1415.

[16] Arjovsky M, Shah A, Bengio Y. Unitary evolution recurrent neural networks. Proc ICML. 2016;1120–1128.

[17] Trabelsi C, Bilaniuk O, Zhang Y, et al. Deep complex networks. Proc ICLR. 2018.

[18] Benning M, Burger M. Modern regularization methods for inverse problems. Acta Numer. 2018;27:1–111.

[19] Haber E, Ruthotto L. Stable architectures for deep neural networks. Inverse Problems. 2017;34:014004.

[20] Hsieh K, Zhao Z, Huang J, et al. Neural methods for linear system solving. Proc NeurIPS. 2021.