

Comparative Analysis of Cramer's Rule and Machine-Learned Solvers for Small-Scale Linear Systems under Clean and Noisy Conditions

Md. Sifuzzaman^{1,*}, Md. Ayon Mia², Md. Anisur Rahman Pramanik¹, Rabeya Sarker³

¹Department of Computer Science & Engineering, Dhaka International University (DIU), Satarkul, Badda, Dhaka-1212, Bangladesh

²Department of Computer Science & Engineering, Southeast University, 252, Tejgaon Industrial Area, Dhaka - 1208, Bangladesh

³Department of Business Administration, Dhaka International University (DIU), Satarkul, Badda, Dhaka-1212, Bangladesh

*Corresponding Author. E-mail address: zaman001217@gmail.com

Received: 06.03.2026; Accepted: 18.06.2026; Published online: 31.08.2026

DOI: <https://doi.org/10.54280/jse.260802>

Abstract: In this work, a comprehensive comparison of classical analytical methods and traditional machine learning (ML) techniques has been made in terms of solving systems of small linear equations in dimensions $n = 2, 3$, and 4 where the exact analytical solutions are still feasible to compute. The established ML models included the Cramer's Rule for performance comparison and analysis. The seven regression models were tested on the projected experimental data set. For evaluation and performance score, different metrics such as the Mean Absolute Error and the Root Mean Square Error were implemented, as well as the relative error and the computation time. The Cramer's rule was executed solely in the noise-free environment due to the basic feature of accuracy loss in numerical perturbation. Our findings demonstrate that although Cramer's Rule retains close-to-optimum accuracy in the ideal condition, the numerical stability significantly deteriorates in the noisy cases; in contrast, the ensemble-based results, and specifically Random Forest and XGBoost, demonstrate higher predictability, better perturbation resilience, and relatively fast computational times. Therefore, data-driven solvers may present viable and viable option to conventional numerical computations when the measurements are uncertain, observations are noisy, or variables are unobservable or when the computational resources are limited. In this respect, our work adds to the literature on machine learning applications in numerical analysis and provides data for hybrid computational methods to be applied in practice.

Key words: Cramer's Rule, Small-scale linear systems, Numerical robustness, Noisy data analysis, Supervised machine learning.

1 Introduction

Linear systems of equations are foundational in numerous fields of science and engineering, with applications ranging from control theory and structural mechanics to data fitting, signal processing, and machine learning [1]. The general form of a linear system is: $\mathbf{A} \mathbf{x} = \mathbf{b}$, where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a coefficient matrix, $\mathbf{x} \in \mathbb{R}^n$ is the unknown vector, and $\mathbf{b} \in \mathbb{R}^n$ is the known output vector. This formulation serves as the basis for a wide spec-

trum of analytical and numerical methodologies. A number of traditional “exact” solvers, such as Gaussian elimination, LU decomposition, and Cramer's Rule, have been used for an extended period of time to compute solutions to these systems [2]. Among them, Cramer's Rule is of particular interest because of the elementary manner of its derivation: it is based on determinants and is both theoretically elegant and educationally clear. It guarantees exact solutions when the coefficient matrix is non-singular and well-

conditioned. However, its practical applicability is limited by factorial time complexity ($O(n!)$) and numerical instability in the presence of near-singular matrices or floating-point round-off errors [3]. As system size increases, the cost of computing determinants renders this method inefficient for real-world problems, particularly when compared with more scalable direct and iterative methods. With the proliferation of high-performance computing and access to large datasets, machine learning (ML) techniques have emerged as viable alternatives to traditional numerical solvers, especially in cases where model generalization and data-driven inference are advantageous [4-5]. In particular, supervised regression methods can be trained to approximate the inverse mapping: $f : (A, B) \rightarrow x$, learning from examples rather than computing explicit algebraic transformations.

Despite these advances, the utility of ML-based solvers for low-dimensional linear systems remains largely unexplored. Most existing research focuses on large-scale, sparse, or time-dependent systems [6-7], overlooking practical scenarios where systems of size $n = \{2, 3, 4\}$ arise, such as embedded systems, sensor networks, or robotic controllers [8-9]. In such applications, trade-offs between precision, robustness to noise, and runtime become paramount, especially under resource constraints. However, systematic comparative evaluations between ML models and classical exact solvers like Cramer's Rule under these constraints are still lacking.

This paper presents a comprehensive comparative study between Cramer's Rule and a suite of supervised machine-learned regression models, including Linear Regression, Ridge Regression, Random Forest, XGBoost, Support Vector Regressor (SVR), K-Nearest Neighbors (KNN), and Multi-Layer Perceptron (MLP). The evaluation spans both clean and noisy data scenarios across system sizes $n = 2, 3, 4$, aiming to quantify how well data-driven models approx-

imate classical solvers under varying degrees of perturbation. To our knowledge, this is the first study that systematically benchmarks learned solvers in such low-dimensional regimes against exact analytical baselines.

The contributions of this work are as follows:

- We develop a unified framework for both exact and learned solvers on small-scale linear systems, ensuring consistency in metric computation and data generation.
- We introduce Gaussian noise in both the coefficient matrix and the output vector to model real-world imperfections and evaluate solver robustness under a noisy environment.
- We perform a detailed profiling of performance across error metrics (MAE, RMSE, relative error) and runtime, revealing the advantages and challenges of different ML models in the context of deterministic solvers.

2 Related Works

The use of machine learning (ML) models as alternatives or complements to traditional linear solvers has gained notable attention in recent years, particularly in scientific computing domains. Several studies have demonstrated the efficacy of deep neural networks (DNNs) in solving systems of linear equations derived from discretized PDEs or physics-based models. For instance, Jiang et al. [10] developed a residual-based DNN for PDE-driven systems, achieving solution errors below 10^{-7} while offering improved scalability over classical solvers. Similarly, Gu and Ng [11] proposed DNNs with sublinear parameter complexity to approximate solutions to large sparse systems. Antony et al. [12] introduced Line2D-Net for linear elasticity problems using U-Net architectures, and Xiao et al. [13] demonstrated the effectiveness of CNN-based solvers in fluid simulation scenarios. These works confirm

that deep learning architectures can serve as competent solvers in high dimensional and spatially structured systems. However, most studies focus on large-scale applications and do not explore the behavior of ML solvers on low-dimensional systems, where analytical methods like Cramer's Rule offer exact and efficient solutions.

A growing body of research explores the hybridization of ML and classical methods to achieve faster convergence or improved stability. Cui et al. [14] presented a neural augmented iterative solver using spectral corrections learned by a lightweight network, resulting in convergence rates independent of grid scale. Markidis [15] benchmarked Physics Informed Neural Networks (PINNs) against multigrid solvers and concluded that PINNs are better suited as initializers or surrogate models than full solvers. In dynamic settings, zeroing neural networks (ZNNs) have emerged as robust solvers for time-varying systems. Notably, Tavakkoli et al. [9] proposed a noise-tolerant RNN-based solver for time-varying matrix inversion, while Zhao et al. [16] and Li et al. [17] extended ZNN frameworks with fixed-time and segmented activation dynamics to enhance noise resilience and convergence speed. Other variants, such as fuzzy logic-integrated ZNNs Guo et al. [18] show promising robustness under measurement noise. These developments collectively support the viability of recurrent and hybrid neural solvers in dynamic, uncertain environments, particularly where classical methods degrade in stability.

Despite these advances, a clear gap remains in the literature concerning small-scale linear systems (e.g., $n = 2, 3, 4$). While machine learning has been applied to large sparse systems and nonlinear PDEs, few works benchmark ML regressors, such as Random Forests, Support Vector Regression (SVR), Ridge Regression, or XGBoost as direct solvers for compact linear systems. Arslan and Bozyigit [19] surveyed ML models for solver recommendation but did not evaluate them

for direct prediction tasks. Related studies Chen et al. [20] and Weymouth [21] emphasized acceleration in large systems but excluded the trivial-size regime. Furthermore, traditional error metrics such as MAE, RMSE, and inference time are inconsistently reported across methods. Our work addresses this underexplored niche by rigorously evaluating classical regression models, tree-based ensembles, kernel methods, and neural networks on low-dimensional linear systems under both clean and noisy conditions. By comparing their performance to Cramer's Rule, a ground truth exact solver, we provide new insights into the efficacy, robustness, and practical limitations of ML solvers in fundamental linear algebra tasks.

3 Methodology

3.1 Problem Formulation

This study investigates the comparative performance of traditional analytical and modern machine-learned solvers in solving small-scale linear systems of the form:

$$\mathbf{Ax}=\mathbf{B} \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a square coefficient matrix, $\mathbf{x} \in \mathbb{R}^n$ is the unknown vector, and $\mathbf{b} \in \mathbb{R}^n$ is the known right-hand side. The focus is restricted to low-dimensional systems with $n \in \{2, 3, 4\}$, representing the practical boundary where **Cramer's Rule** remains computationally feasible.

The central objective is to examine whether machine-learned models can offer comparable or superior accuracy, robustness, and speed in solving these systems under both ideal and noisy conditions.

3.2 Dataset Generation

For each value of n , two datasets were synthetically generated:

- A clean dataset containing exact solutions.
- A noisy dataset with Gaussian perturbations applied to both $\mathbf{A}$ and $\mathbf{b}$.

Each sample was created by:

1. Sampling a random matrix $\mathbf{A}$ from the standard normal distribution.
2. Generating a random ground-truth vector $\mathbf{x}_{\text{true}}$.
3. Computing $\mathbf{b} = \mathbf{A}\mathbf{x}_{\text{true}}$.
4. For noisy data, inject Gaussian noise:

$$\mathbf{A}' = \mathbf{A} + \mathbf{N}(0, \sigma^2), \quad (2)$$

$$\mathbf{b}' = \mathbf{b} + \mathbf{N}(0, \sigma^2), \quad (3)$$

where $\sigma = 0.05$. Each configuration consisted of 5000 training samples and 1000 test samples, with a fixed random seed to ensure reproducibility.

3.3 Cramer's Rule Solver

Cramer's Rule solves each component of the solution vector using the expression:

$$\mathbf{x}_i = \frac{\det(\mathbf{A}_i)}{\det(\mathbf{A})} \quad (4)$$

where $\mathbf{A}_i$ is obtained by replacing the i -th column of $\mathbf{A}$ with $\mathbf{b}$. The method is exact in the absence of round-off error, but becomes unstable when $\det(\mathbf{A})$ is near zero. To ensure valid evaluation, any matrix $\mathbf{A}$ with $|\det(\mathbf{A})| < 10^{-12}$ was considered nearly singular and excluded from Cramer's Rule experiments.

3.4 Machine-Learned Solvers

To approximate the mapping from the pair $[\text{vec}(\mathbf{A}), \mathbf{b}]$ to the solution vector $\mathbf{x}$, multiple supervised regression models were employed. The following models were implemented and compared:

- **Linear Regression (LR):** Ordinary least squares without regularization.
- **Ridge Regression:** Linear model with L2 regularization to improve stability under noise.
- **Random Forest:** An ensemble of 100 decision trees trained with bootstrap aggregation.

- **XGBoost:** Gradient-boosted decision trees optimized for accuracy and regularization.
- **Support Vector Regressor (SVR):** A kernel-based model using the RBF kernel for non-linear regression.
- **K-Nearest Neighbors (KNN):** A non-parametric model using Euclidean distance in the input space.
- **Multi-Layer Perceptron (MLP):** A feedforward neural network with two hidden layers (64 units each), *ReLU* activation, and *Adam* optimizer for 300 epochs.

All models were trained on clean data and evaluated on both clean and noisy test sets to assess generalization ability and robustness to perturbations.

4 Results and Discussion

This section evaluates the performance of analytical and machine-learned solvers across small-scale linear systems for both clean and noisy datasets. The experiments focus on accuracy, robustness, and computational efficiency, providing a comparative analysis for $n \in \{2, 3, 4\}$.

4.1 Evaluation Metrics

The following metrics are employed to assess the performance of each solver in terms of accuracy and computational efficiency.

Mean Absolute Error (MAE).

Quantifies the average magnitude of deviation between the predicted and true solution components:

$$MAE = \frac{1}{n} \sum_{i=1}^n |x_i - \hat{x}_i| \quad (5)$$

It represents the average absolute difference between the true solution x_i and the estimated value $\hat{x}_i$, providing a straightforward measure of accuracy that is less sensitive to outliers.

Root Mean Square Error (RMSE).

Defined as:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \hat{x}_i)^2} \quad (6)$$

This metric penalizes larger deviations more heavily, making it suitable for evaluating models where large errors are particularly undesirable.

Relative Error (RelErr).

Measures the normalized deviation of the estimated solution from the true solution:

$$RelErr = \frac{\|\hat{x} - x\|_2}{\|x\|_2} \quad (7)$$

It provides a scale-invariant measure of accuracy, enabling fair comparison across systems with different magnitudes of $\mathbf{x}$.

Time Per Solve (ms).

The average computational time required to solve one instance of a linear system is measured in milliseconds:

$$T_{avg} = \frac{1}{N} \sum_{j=1}^N t_j \quad (8)$$

where t_j denotes the wall-clock time taken for the j -th test case and N is the total number of test samples. This metric captures the runtime efficiency of each solver.

4.2 Results Analysis**Results on clean data.**

The comparative performance of machine-learned and exact solutions on clean linear

systems spanning dimensions $n = 2, 3, 4$ is summarized in Table 1. While Cramer's Rule is deterministic and depends on precise arithmetic, it produces almost zero error across all assessed metrics, including MAE, RMSE, and Relative Error. However, due to the factorial complexity involved in calculating matrix determinants and adjugates, its time per solve increases with dimensionality. Among the machine-learned approaches, XGBoost and Random Forest exhibit notably lower error rates than their linear counterparts, indicating their superior capacity to model complex patterns, even in small systems. While Linear and Ridge Regression are extremely fast and stable, their error values remain consistently higher, revealing limitations in approximating systems not well aligned with their assumptions of linearity and feature independence. KNN and SVR methods offer mixed results; although they show some ability to capture structure, their runtime and relative errors suggest lower scalability and sensitivity to input configurations. MLP models perform reasonably well in balancing accuracy and efficiency but are slightly less effective than ensemble methods. Overall, these observations confirm that, in the absence of noise, ensemble learners, especially XGBoost, provide the best trade-off between accuracy and computational cost when exact solutions like Cramer's Rule are infeasible for higher dimensions.

Results on noisy data.

The findings for noisy linear systems are shown in Table 2, where each solver's performance has been evaluated under relatively small input data perturbations. In light of its numerical instability, Cramer's Rule is not appropriate in this situation; even slight variations in input values can substantially affect the determinant, making solutions unreliable. All error metrics across models and dimensions generally rise when noise is present. However, ensemble-based techniques Random

Forest and XGBoost in particular, continue to show great robustness and display comparatively slower accuracy decline than other techniques. This robustness is attributed to their inherent averaging mechanisms and ability to capture feature interactions. Although Ridge Regression and Linear Regression perform almost identically with short inference intervals, they frequently result in systematic underfitting because they are unable to sufficiently account for the added noise. Particularly for SVR, which suffers from costly

Table 1: Performance comparison of exact and machine-learned solvers on clean linear systems of size $n=2,3,4$. Metrics include MAE, RMSE, Relative Error, and Time per Solve (ms). Bold values indicate the best performance per metric.

n	Metric	Method							
		Cramer	Linear	Ridge	RF	SVR	KNN	MLP	XGB
2	MAE	2.40e-16	0.791	0.791	0.171	0.399	0.356	0.265	0.178
	RMSE	8.40e-16	0.993	0.993	0.240	0.579	0.506	0.381	0.255
	RelErr	3.47e-16	0.997	0.997	0.227	0.518	0.534	0.431	0.278
	Time (ms)	0.0322	0.00023	0.00020	0.0405	0.378	0.0383	0.0012	0.0155
3	MAE	9.34e-16	0.799	0.799	0.223	0.443	0.471	0.402	0.174
	RMSE	1.18e-14	1.008	1.008	0.292	0.622	0.618	0.518	0.241
	RelErr	1.04e-15	1.001	1.001	0.273	0.562	0.618	0.553	0.244
	Time (ms)	0.0370	0.00026	0.00024	0.0405	0.724	0.249	0.0014	0.0226
2	MAE	8.66e-16	0.799	0.799	0.245	0.407	0.529	0.433	0.141
	RMSE	3.75e-15	1.002	1.002	0.314	0.583	0.680	0.554	0.194
	RelErr	1.15e-15	0.999	0.999	0.305	0.536	0.673	0.573	0.198
	Time (ms)	0.0448	0.00025	0.00021	0.0400	1.010	0.163	0.0012	0.0311

Table 2: Quantitative evaluation of machine-learned solvers on noisy linear systems of size $n = 2, 3, 4$. Metrics reported are Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Relative Error (RelErr), and average Time per Solve (ms). Cramer's Rule is omitted due to its numerical instability under noise. Bold values highlight the best-performing method for each metric.

n	Metric	Method						
		Linear	Ridge	RF	SVR	KNN	MLP	XGB
2	MAE	0.793	0.793	0.999	0.996	1.033	1.095	1.052
	RMSE	0.996	0.996	1.250	1.239	1.291	1.368	1.313
	RelErr	1.006	1.006	1.679	1.662	1.767	1.901	1.819
	Time (ms)	0.00027	0.00024	0.0466	0.388	0.0410	0.0012	0.0121
3	MAE	0.798	0.798	0.958	0.968	0.964	1.051	1.034
	RMSE	1.008	1.008	1.203	1.220	1.217	1.328	1.306
	RelErr	0.999	0.999	1.310	1.328	1.332	1.485	1.446
	Time (ms)	0.00027	0.00027	0.0490	0.742	0.252	0.0013	0.0170
4	MAE	0.803	0.803	0.923	0.953	0.920	1.011	1.015
	RMSE	1.008	1.008	1.163	1.204	1.161	1.275	1.282
	RelErr	1.003	1.003	1.207	1.262	1.209	1.349	1.357
	Time (ms)	0.00032	0.00028	0.0479	1.024	0.112	0.0012	0.0632

kernel evaluations, models such as KNN and SVR suffer from substantial efficiency drops, elevated error values, and increased time costs. MLP models show sensitivity to both hyperparameter adjustment and data scale, however they are still somewhat competitive but less consistent across di-

mensions. These findings collectively reveal that ensemble learners offer the best empirical trade-offs in terms of error tolerance and scalability for small-scale, perturbed systems, even though no approach is completely impervious to noisy inputs.

Visual Insights into Model Performance. The variation in Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for all analyzed approaches across various system sizes is shown in Figure 1 and Figure 2, respectively. The illustrations provide a better grasp of each model's performance trajectory and support the tabular data. Cramer's Rule routinely exhibits almost zero error, which is consistent with its deterministic accuracy in clean linear systems. As dimensionality increases, XGBoost and Random

Forest exhibit the most stable and advantageous error profiles among machine-learned solvers. Specifically, XGBoost shows a little decrease in MAE of about 20% from $n=2$ to $n=4$, indicating its resilience and flexibility even as system complexity increases. In contrast, linear models such as Linear and Ridge Regression maintain nearly flat error curves, indicating stable yet suboptimal approximation capacity due to their inherent assumptions about feature linearity and independence.

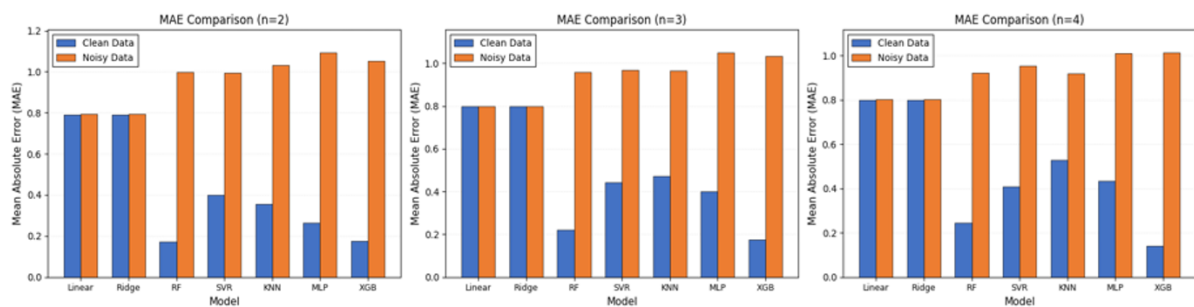


Figure 1: MAE comparison of solvers across system sizes $n = 2, 3, 4$.

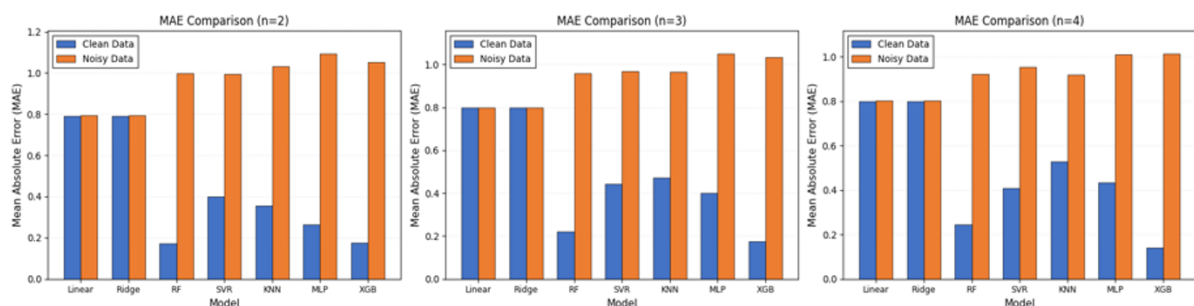


Figure 2: RMSE comparison of solvers across system sizes $n = 2, 3, 4$.

SVR and KNN show more erratic tendencies in all dimensions; when n increases from 2 to 4, SVR's RMSE increases by more than 60% and KNN's MAE increases

by more than 48%. These trends point to deteriorating generalization performance for techniques that are sensitive to high-dimensional distributions and data shape.

Despite having competitive early performance, the MLP model gradually deteriorates in both MAE and RMSE metrics, underscoring its reliance on architectural adjustment and data volume. Importantly, RMSE trends magnify these distinctions by emphasizing larger deviations in prediction error, where ensemble methods again maintain lower and more consistent values. Overall, these visual insights affirm that ensemble learners, particularly XGBoost offer a favorable trade-off between scalability and precision, making them suitable candidates for solving small to moderate-sized linear systems when exact methods are impractical.

5 Conclusion

This study compared traditional analytical techniques with advanced machine learning techniques in order to solve small scale linear systems under both ideal and noisy settings. Experiments for systems where $n = 2, 3, 4$ showed that Cramer's rule works remarkably well for clean, noise-free data. Its accuracy is almost unmatched in these situations. But the study also showed how quickly even little disruptions or computational errors cause its performance to decline. These flaws make it less appropriate for real-world applications involving uncertainty or scarce resources, in addition to its high computing cost.

On the other hand, a number of supervised learning algorithms, like Random Forest and XGBoost, demonstrated robust and reliable outcomes. They solved problems far more quickly than conventional analytical techniques and maintained an acceptable accuracy in both clean and noisy situations. They might be sensible options for real-time or low-resource systems when an exact but slower solution is not required because of this balancing between speed and dependability. Overall, this work shows how integrating concepts from machine learning and classical computation may present new opportuni-

ties for effective problem solving. Future studies could investigate broader or more complicated systems or develop hybrid solutions that combine the adaptability of data-driven models with the clarity of analytical techniques.

Acknowledgement

The authors are thankful to Dhaka International University, Dhaka, Bangladesh, for all kinds of research support.

Conflict of Interest

There is no conflict of interest.

References

- [1] Y. Saad, "Iterative Methods for Sparse Linear Systems Second Edition", 2003.
- [2] L. N. Trefethen and D. Bau, Numerical Linear Algebra. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1997.
- [3] G. H. Golub and C. F. Van Loan, Matrix Computations. Johns Hopkins University Press, 2013.
- [4] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J Comput Phys*, vol. 378, pp. 686–707, 2019.
- [5] Z. Jiang, J. Jiang, Q. Yao, and G. Yang, "A neural network-based PDE solving algorithm with high precision," *Sci Rep*, vol. 13, no. 1, p. 4479, 2023.
- [6] G. D. Weymouth, "Data-driven multi-grid solver for accelerated pressure projection," *Comput Fluids*, vol. 246, p. 105620, 2022.
- [7] R. Zhao, M. Li, T. Liu, and S.-X. Miao, "A noise-tolerant zeroing neural network with fixed-time convergence for solving multi-linear systems with nonsingular M-tensors," *AIMS Mathematics*, vol. 10, no. 6, pp. 13974–13995, 2025.
- [8] R. Li, J. Jin, D. Zhang, and C. Chen, "A Segmented Activation Function-Based Zeroing Neural Network Model for Dynamic Sylvester Equation Solving and Robotic Manipulator Control," *Concurr Comput*, vol. 37, no. 21–22, p. e70243, 2025.

- [9] V. Tavakkoli, J. C. Chedjou, and K. Kyamakya, "A novel recurrent neural network-based ultra-fast, robust, and scalable solver for inverting a 'time-varying matrix,'" *Sensors*, vol. 19, no. 18, p. 4002, 2019.
 - [10] Z. Jiang, J. Jiang, Q. Yao, and G. Yang, "A neural network-based PDE solving algorithm with high precision," *Sci Rep*, vol. 13, no. 1, p. 4479, 2023.
 - [11] Y. Gu and M. K. Ng, "Deep neural networks for solving large linear systems arising from high-dimensional problems," *SIAM Journal on Scientific Computing*, vol. 45, no. 5, pp. A2356–A2381, 2023.
 - [12] A. Antony, M. Mayur, S. Mukherjee, and A. Dasgupta, "Linell2D-Net: A Deep Learning Approach to Solving 2D Linear Elastic Boundary Value Problems on Image Domains," *Comput Methods Appl Mech Eng*, 2024.
 - [13] X. Xiao, Y. Zhou, H. Wang, and X. Yang, "A Novel CNN-Based Poisson Solver for Fluid Simulation," *IEEE Trans Vis Comput Graph*, 2020.
 - [14] C. Cui, K. Jiang, Y. Liu, and S. Shu, "A Hybrid Iterative Neural Solver Based on Spectral Analysis for Parametric PDEs," *J Comput Phys*, p. 114165, 2025.
 - [15] S. Markidis, "The old and the new: Can physics-informed deep-learning replace traditional linear solvers?," *Front Big Data*, vol. 4, p. 669097, 2021.
 - [16] R. Zhao, M. Li, T. Liu, and S.-X. Miao, "A noise-tolerant zeroing neural network with fixed-time convergence for solving multi-linear systems with nonsingular M-tensors".
 - [17] Q. Li, Z. Wang, S.-S. Wang, and X. Chen, "A Segmented Activation Function-Based Zeroing Neural Network Model for Dynamic Sylvester Equation Solving and Robotic Manipulator Control," *IEEE Trans Neural Netw Learn Syst*, 2025.
 - [18] Y. Guo, T. Huang, and W. Zhang, "New Fuzzy Zeroing Neural Network with Noise Suppression Capability for Time-Varying Linear Equation Solving," *IEEE Trans Cybern*, 2025.
 - [19] H. Arslan and F. Bozyiğit, "Machine Learning Algorithms for Solving Linear Systems of Equations," in *Handbook of Research on New Investigations in Artificial Life, AI, and Machine Learning*, IGI Global, 2021.
 - [20] X. Chen, G. Yang, Q. Yao, Z. Nie, and Z. Jiang, "A compressed lattice Boltzmann method based on ConvLSTM and ResNet," *Computers & Mathematics with Applications*, 2021.
 - [21] G. D. Weymouth, "Data-driven Multi-Grid solver for accelerated pressure projection," *Comput Fluids*, 2022.
-