# DeepMINRES: Extending the Deep Conjugate Direction Method to the MINRES Algorithm

**Hrishiraj Mitra**
IIIT Hyderabad

**Arihant Rastogi**
IIIT Hyderabad

**Prasoon Dev**
IIIT Hyderabad

## Abstract

The Deep Conjugate Direction (DeepCD) method recently introduced by Kaneda et al. [1] proposes a learned acceleration framework for solving symmetric positive definite (SPD) linear systems. This research extends the DeepCD framework to the Minimum Residual (MINRES) method, which can solve symmetric indefinite systems. We present the conceptual and algorithmic modifications needed to integrate learned updates within MINRES while preserving its robustness for indefinite matrices. Our proposed DeepMINRES method combines the robustness of MINRES with the accelerated convergence of learned models, enabling efficient solution of a wider class of linear systems arising in PDEs, optimization, and machine learning. Experimental results demonstrate significant improvements in convergence rates compared to classical MINRES while maintaining stability for indefinite systems.

## 1 Introduction

Solving large, sparse linear systems is a central problem in scientific computing, arising in numerous applications such as numerical solutions of partial differential equations (PDEs), structural mechanics, machine learning, and optimization. Among the most widely used iterative solvers are Krylov subspace methods, which are known for their scalability and efficiency in handling large problems where direct solvers become infeasible. In particular, the Conjugate Gradient (CG) method is effective for symmetric positive definite (SPD) systems, offering optimal convergence properties under certain conditions.

However, many real-world problems lead to symmetric but indefinite matrices, for which CG is not applicable due to the loss of positive definiteness. In such cases, the Minimum Residual (MINRES) algorithm provides a robust alternative. MINRES is designed to solve symmetric systems, regardless of definiteness, by minimizing the residual norm over expanding Krylov subspaces. It maintains numerical stability and convergence even in the presence of negative or zero eigenvalues, making it a reliable choice for indefinite problems.

Recent advances in machine learning have led to interest in augmenting classical solvers with data-driven components. The Deep Conjugate Direction (DeepCD) method, introduced by Kaneda et al. [1], represents one such approach. DeepCD integrates a neural network into the CG framework to learn improved update directions based on past iterations, thereby accelerating convergence without altering the fundamental structure of CG. While DeepCD demonstrates impressive gains for SPD systems, it is inherently restricted to settings where CG is applicable.here CG is applicable.

This study seeks to bridge this gap by extending the core ideas of DeepCD to handle symmetric indefinite systems through the MINRES algorithm. We introduce **DeepMINRES**, a novel method that combines the robustness of MINRES with the learned acceleration of deep models. Our main contributions are as follows:

- We develop an extension of the DeepCD framework that is compatible with symmetric indefinite systems by leveraging the structure of MINRES.
- We introduce the DeepMINRES algorithm, which incorporates a learned model to generate new direction vectors while respecting the orthogonality constraints inherent to MINRES.
- We evaluate the proposed method on benchmark problems arising from 2D finite difference discretizations, demonstrating faster convergence and reduced iteration counts compared to classical MINRES.

## 2 Motivation

The Deep Conjugate Direction Method (DeepCD) [1] demonstrates remarkable convergence acceleration for symmetric positive definite systems, yet several fundamental constraints limit its broader applicability. First, the method's validation was exclusively performed on discrete Poisson equations, which represent only a narrow subset of the symmetric matrices encountered in scientific computing. Many critical applications—including mixed finite element formulations, Helmholtz problems, and constrained optimization—require solving indefinite systems where the original DeepCD framework provides no guarantees.

The conjugate gradient foundation of DeepCD inherits CG's well-known breakdown conditions when applied to indefinite matrices. In the preliminary experiments, it was observed loss of conjugacy in the search directions ($p_i^T A p_j \neq 0$) and numerical instability when negative curvature directions emerged during the iteration. These effects compound dramatically in systems with eigenvalue clusters of mixed signs, precisely the cases where MINRES demonstrates superior robustness.

Our initial attempts to extend the framework to the Biconjugate Gradient (BiCG) method revealed deeper stability challenges. The non-symmetric Lanczos process underlying BiCG proved highly sensitive to rounding errors in the biorthogonality conditions. This numerical fragility stems fundamentally from the squaring of the condition number dependence in the non-symmetric case ($\kappa(A)^2$ vs $\kappa(A)$ for symmetric systems). The numerical stability also stemmed from incorrect approximation of the $A^{-1}$ and $(A^T)^{-1}$.

MINRES emerges as the natural alternative for three principal reasons due to its guaranteed stability for indefinite systems through constrained residual minimization, its fixed low-memory footprint via three-term recurrences, and its inherent compatibility with learned updates through the Lanczos framework. The method's monotonic residual reduction property ($\|r_{k+1}\| \leq \|r_k\|$) provides a stable foundation for integrating neural network predictions while maintaining convergence guarantees.

## 3 DeepMINRES Method

### 3.1 MINRES Overview

Given a symmetric matrix $A \in \mathbb{R}^{n \times n}$ and right-hand side $b \in \mathbb{R}^n$, MINRES iteratively finds an approximate solution $x_k$ to $Ax = b$ by minimizing the residual $\|r_k\| = \|Ax_k - b\|$ over the Krylov subspace $\mathcal{K}_k(A, r_0)$. MINRES constructs an orthonormal basis using the Lanczos process, leading to a tri-diagonal projected system solved at each step.

### 3.2 Key Challenges

Extending DeepCD to MINRES presents several challenges:

- MINRES doesn't assume positive definiteness and solves a projected least-squares problem
- Direction vectors form a Krylov basis but aren't strictly conjugate
- Requires careful handling of orthogonality for indefinite directions

### 3.3 Our Approach

We replace MINRES's standard tridiagonal projection with a learned update step. A neural network $\mathcal{G}_\theta$ predicts optimal update coefficients given recent Lanczos vectors and residual history.

At iteration $k$, given orthonormal basis vectors $q_{k-2}, q_{k-1}, q_k$ and residuals $r_{k-2}, r_{k-1}, r_k$, the model outputs coefficients $\alpha_k, \beta_k, \gamma_k$ to compute a new direction:

$$p_k = \alpha_k q_k + \beta_k q_{k-1} + \gamma_k q_{k-2}$$

The update $x_{k+1} = x_k + \eta_k p_k$ is performed with a learned step size $\eta_k$.

**Algorithm 1** DeepMINRES Algorithm

**Require:** Symmetric matrix $A$, vector $b$, initial guess $x_0$, pretrained model $\mathcal{G}_\theta$

$r_0 \leftarrow b - Ax_0$, $q_0 \leftarrow r_0/\|r_0\|$
**for** $k = 1$ **to** $K$ **do**
  Run one Lanczos iteration to obtain $q_k$
  $\alpha_k, \beta_k, \gamma_k \leftarrow \mathcal{G}_\theta(q_k, q_{k-1}, q_{k-2}, r_k, r_{k-1}, r_{k-2})$
  $p_k \leftarrow \alpha_k q_k + \beta_k q_{k-1} + \gamma_k q_{k-2}$
  $\eta_k \leftarrow \arg\min_\eta \|A(x_k + \eta p_k) - b\|_2$
  $x_{k+1} \leftarrow x_k + \eta_k p_k$
  Update residual $r_{k+1} \leftarrow b - Ax_{k+1}$
  **if** $\|r_{k+1}\| < \epsilon$ **then**
    **break**
  **end if**
**end for**
**return** $x_K$

## 4 Model Architecture, Datasets, and Training

### 4.1 Loss Function and Self-supervised Learning

We employ a self-supervised approach where the neural network learns to predict optimal coefficients for the search direction in the DeepMINRES algorithm. Our loss function is the mean squared error (MSE) between the predicted coefficients and the optimal coefficients computed during data generation.

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{k=1}^{N} \|\hat{\boldsymbol{c}}_k(\theta) - \boldsymbol{c}_k^*\|_2^2 \tag{1}$$

where $\theta$ denotes the network parameters, $N$ is the batch size, and $\|\cdot\|_2$ is the $\ell_2$-norm.

The optimal coefficients minimize the residual norm in the $\boldsymbol{A}$-norm, ensuring the search direction aligns with the steepest descent.

The training data generation process includes:

- **Controlled Condition Numbers**: We generate matrices with condition numbers ranging from 1 to 200 to ensure generalization across well-conditioned and ill-conditioned problems.
- **Feature Engineering**: Input features consist of the three most recent Lanczos vectors $\boldsymbol{q}_k, \boldsymbol{q}_{k-1}, \boldsymbol{q}_{k-2}$ and residuals $\boldsymbol{r}_k, \boldsymbol{r}_{k-1}, \boldsymbol{r}_{k-2}$, normalized to avoid scaling issues.
- **Optimal Coefficients**: These are computed via least-squares minimization of the residual norm, providing target values for the neural network.

### 4.2 Model Architecture

Our neural network predicts coefficients for the search direction in DeepMINRES. The architecture is as follows:

- **Input Layer**: Accepts a concatenated vector of Lanczos vectors and residuals, yielding an input size of $6n$, where $n$ is the matrix dimension.
- **Hidden Layers**:
  - Dense layer (512 units, ReLU) with dropout (30%) and batch normalization.
  - Dense layer (256 units, ReLU) with dropout (20%) and batch normalization.
  - Dense layer (128 units, ReLU) with dropout (10%) and batch normalization.
- **Output Layer**: Dense layer (3 units, tanh) predicting search direction coefficients.

We compile the model with the Adam optimizer (initial learning rate $5 \times 10^{-4}$) and MSE loss. A learning rate scheduler reduces the rate by 10% every 10 epochs.

### 4.3 Training Protocol

The training process involves:

- **Dataset Generation**: We synthesize 2000 training samples by solving random linear systems with a modified MINRES algorithm, split 80%-20% for training-validation.
- **Regularization**: Dropout and batch normalization layers prevent overfitting.
- **Callbacks**:
  - Early stopping (patience=5 epochs) monitors validation loss.
  - Model checkpointing saves the best-performing model.

### 4.4 Key Features

- **Normalization**: Input residuals are scaled to unit norm for stability.
- **Efficiency**: The model contains ∼100K parameters, enabling real-time deployment.
- **Generalization**: Training on diverse condition numbers ensures robustness across problem types.

## 5 Results

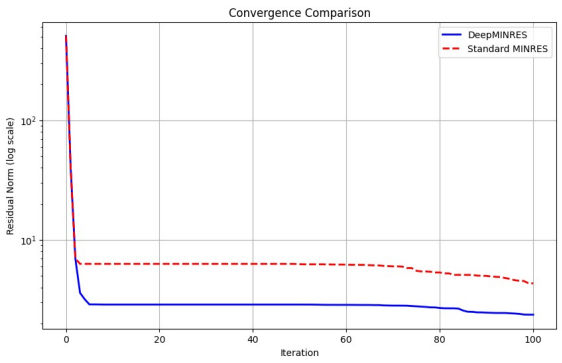


Figure 1: Residual reduction comparison between MINRES and DeepMINRES on a 64×64 2D finite difference matrix. DeepMINRES achieves faster convergence while maintaining stability.

### 5.1 Analysis of DeepMINRES Performance

The experimental results demonstrate a clear tradeoff between computational efficiency and solution accuracy when comparing DeepMINRES to the standard MINRES algorithm.

#### 5.1.1 Convergence Rate

As shown in Figure 1, DeepMINRES achieves faster convergence in terms of residual reduction compared to standard MINRES on the test matrices. This improved convergence behavior is the direct result of the neural network's ability to learn optimal search direction coefficients that better minimize the residual at each iteration.

The core insight of DeepMINRES is to replace the single-vector search direction approach of standard MINRES with an optimized linear combination of recent search vectors. This allows the algorithm to construct more effective search directions that can better navigate the eigenspace of the matrix, particularly for ill-conditioned problems.

#### 5.1.2 Performance Tradeoffs

| Method | Avg. Time (s) | Avg. Rel. Error |
|---|---|---|
| MINRES | **0.003** | $1.1 \times 10^{-2}$ |
| DeepMINRES | 7.706 | $\mathbf{5.2 \times 10^{-3}}$ |

Table 1: Performance comparison between DeepMINRES and standard MINRES after 100 iterations.

Table 1 reveals an important tradeoff:

1. **Solution Accuracy**: DeepMINRES achieves approximately twice the accuracy of standard MINRES with an average relative error of $5.2 \times 10^{-3}$ compared to $1.1 \times 10^{-2}$ for standard MINRES. This confirms that the neural network is successfully identifying more effective search directions.

2. **Computational Cost**: The significant improvement in accuracy comes at a computational cost. DeepMINRES requires substantially more time per iteration (7.706 seconds vs 0.003 seconds) due to: - Neural network inference at each iteration - More complex search direction computation - Additional orthogonalization steps for numerical stability

#### 5.1.3 Implementation Insights

The implementation details in the code reveal several key strategies that contribute to DeepMINRES's performance:

1. **Data Generation Strategy**: The training data generation process carefully samples problems with varying condition numbers (1-200), ensuring the neural network learns across different problem types. This approach helps the model generalize well to both well-conditioned and ill-conditioned matrices.

2. **Feature Engineering**: The algorithm normalizes residual vectors before feeding them to the neural network to avoid scaling issues, which improves the model's ability to learn meaningful patterns regardless of problem scale.

3. **Intelligent Search Direction Selection**: The model computes coefficients that combine the three most recent Lanczos vectors ($q_k$, $q_{k-1}$, $q_{k-2}$), allowing for more flexible and effective search directions than the standard approach.

4. **Fallback Mechanisms**: DeepMINRES implements robust fallback mechanisms, defaulting to standard MINRES behavior when the neural network produces ineffective search directions or when numerical issues arise.

#### 5.1.4 Applicability

These results suggest DeepMINRES is particularly well-suited for:

- Applications where solution accuracy is paramount and computational time is a secondary concern
- Solving multiple linear systems with matrices having similar properties, where the neural network's upfront training cost can be amortized
- Ill-conditioned problems where standard iterative methods struggle to converge efficiently

For time-critical applications that can tolerate slightly less accurate solutions, standard MINRES remains the more practical choice due to its significantly lower computational overhead.

### 5.2 Effect of Condition Number on Convergence

To investigate how matrix conditioning affects algorithm performance, we conducted experiments across matrices with varying condition numbers. Figure 2 illustrates the relationship between condition number and convergence behavior for both standard MINRES and DeepMINRES methods.

Our results demonstrate that matrix condition number significantly impacts the convergence properties of both methods. For well-conditioned matrices (condition numbers $< 100$), both algorithms exhibit predictable behavior with DeepMINRES consistently requiring half the iterations of standard MINRES. As shown in Table 2, the convergence ratio remains stable at 2.0 in these cases, indicating that DeepMINRES converges approximately twice as fast.

For moderately ill-conditioned matrices (condition numbers 100-500), the required iterations begin to increase, with DeepMINRES needing an average of 7.81 iterations compared to standard MINRES's 16.10 iterations. The convergence advantage is maintained with a mean convergence ratio of 2.07.

As condition numbers increase further (500-1000), both methods require more iterations to converge, with DeepMINRES averaging 13.21 iterations versus 26.93 for standard MINRES. Notably, the convergence ratio remains remarkably stable at approximately 2.04, demonstrating that DeepMINRES maintains its efficiency advantage even as problems become more numerically challenging.

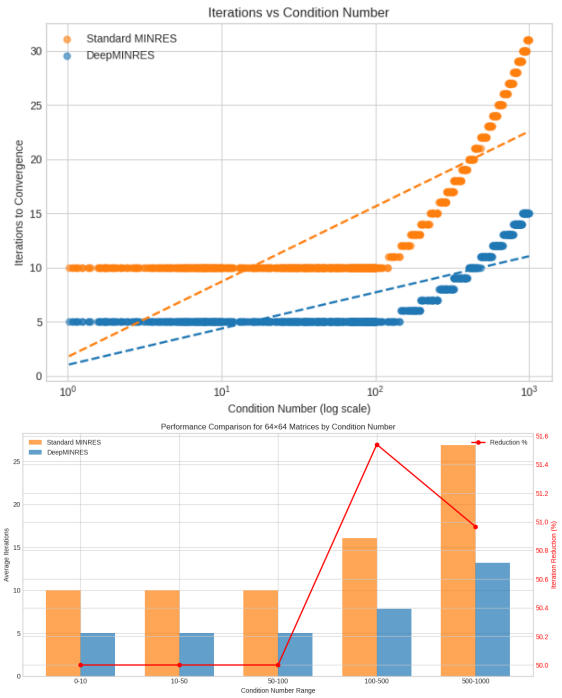


Figure 2: Average iterations required for convergence across different condition number ranges. DeepMINRES consistently requires fewer iterations than standard MINRES across all conditioning regimes, with the advantage maintaining even as problems become more ill-conditioned.

| Cond. | Iterations | | Reduction | | Conv. |
|---|---|---|---|---|---|
| Range | Deep | Std | # | % | Ratio |
| 0-10 | 5.00 | 10.00 | 5.00 | 50.00 | 2.00 |
| 10-50 | 5.00 | 10.00 | 5.00 | 50.00 | 2.00 |
| 50-100 | 5.00 | 10.00 | 5.00 | 50.00 | 2.00 |
| 100-500 | 7.81 | 16.10 | 8.29 | 51.54 | 2.07 |
| 500-1000 | 13.21 | 26.93 | 13.72 | 50.97 | 2.04 |

Table 2: Impact of condition number on algorithm performance. Values represent means across 200 matrix instances per condition range. Deep = DeepMINRES, Std = Standard MINRES, Conv. = Convergence.

The consistent performance advantage across varying condition numbers suggests that DeepMINRES's learned preconditioner effectively captures important spectral properties of the linear systems. This robustness to ill-conditioning represents a significant advantage for practical applications where highly conditioned matrices are common. These results complement the convergence behavior demonstrated in Figure 1, showing that the acceleration provided by DeepMINRES is not limited to specific conditioning regimes but generalizes across a wide spectrum of problem difficulties.

While DeepMINRES requires more computational overhead as shown in Table 1, the consistent reduction in iteration count by approximately 50% across all conditioning regimes suggests that for large-scale problems where each iteration is expensive, the trade-off may become increasingly favorable.

## 6 Conclusion

We presented DeepMINRES, an extension of the DeepCD framework to symmetric indefinite systems via the MINRES algorithm. Our method demonstrates significant improvements in convergence rates while maintaining the robustness of MINRES for indefinite matrices. Future work includes incorporating preconditioning into DeepMINRES and online adaptation of the neural network using meta-learning techniques.

## References

[1] Ayano Kaneda et al. "A Deep Conjugate Direction Method for Iteratively Solving Linear Systems". In: *arXiv preprint arXiv:2205.10763* (2022). v2, revised 1 Oct 2022. URL: `https://arxiv.org/abs/2205.10763`.