Extending the Deep Conjugate Direction Method to the MINRES/BiCG Algorithm

Arihant Rastogi - Prasoon Dev - Hrishiraj Mitra

Numerical Algorithms

May 8, 2025

Contents

1. Understanding DCDM

2. Learning BICG

3. DeepMINRES

Understanding Deep Conjugate Direction Method (DCDM)

Motivation

- Traditional iterative methods (e.g., Conjugate Gradient) can be slow for large-scale problems in computational physics
- Data-driven approaches lack iterative refinement capability
- Need for methods that combine the strengths of both approaches:
  - Deep learning for accelerated convergence
  - Iterative refinement for arbitrary precision
- Particularly important for solving Poisson-type equations in:
  - Computational fluid dynamics
  - Computer graphics simulations
  - Other PDE-based applications
- DCDM bridges this gap by learning optimal search directions while preserving convergence [**?**]

Deep Conjugate Direction Method (DCDM)

Algorithm 1: DCDM

$r_0 = b - A^{\Omega} x_0$;
$k = 1$;
while $\|r_{k-1}\| \geq \epsilon$ **do**
  $d_k = f\left(c, \frac{r_{k-1}}{\|r_{k-1}\|}\right)$;
  for $i_{start} \leq i < k$ **do**
    $h_{ik} = \frac{d_k^T A^{\Omega} d_i}{d_i^T A^{\Omega} d_i}$;
    $d_k = d_k - h_{ik} d_i$;
  $\alpha_k = \frac{r_{k-1}^T d_k}{d_k^T A^{\Omega} d_k}$;
  $x_k = x_{k-1} + \alpha_k d_k$;
  $r_k = b - A^{\Omega} x_k$;
  $k = k + 1$;

Algorithm Limitations

- **Matrix Type**: Only works for SPD systems (Poisson-type)
- **Training Data**: Requires domain-specific training
- **Computational Cost**: CNN inference adds per-iteration overhead and required high system requirements to even train on curated datasets
- **Preconditioners**: May not beat specialized methods in all cases

Learning BiCG

Motivation

- Many large-scale linear systems from real-world applications are non-symmetric or indefinite
- Classical Conjugate Gradient (CG) is limited to symmetric positive-definite matrices
- Need for iterative methods that handle broader matrix classes efficiently
- BiCG extends CG by:
  - Using bi-orthogonal search directions
  - Simultaneously solving with A and A^T
- Particularly useful in:
  - Computational electromagnetics
  - Structural mechanics with damping
  - Non-symmetric PDE discretizations
- BiCG enables convergence where CG fails, maintaining matrix-free efficiency

BiConjugate Gradient (BiCG)

Algorithm 2: BiConjugate Gradient (BiCG)

$r_0 = b - Ax_0$;
$\tilde{r}_0 = \tilde{b} - A^T \tilde{x}_0$;
$p_0 = r_0$;
$\tilde{p}_0 = \tilde{r}_0$;
$k = 0$;
while $\|r_k\| \geq \epsilon$ **do**
  $\alpha_k = \frac{\tilde{r}_k^T r_k}{\tilde{p}_k^T A p_k}$;
  $x_{k+1} = x_k + \alpha_k p_k$;
  $r_{k+1} = r_k - \alpha_k A p_k$;
  $\tilde{r}_{k+1} = \tilde{r}_k - \alpha_k A^T \tilde{p}_k$;
  $\beta_k = \frac{\tilde{r}_{k+1}^T r_{k+1}}{\tilde{r}_k^T r_k}$;
  $p_{k+1} = r_{k+1} + \beta_k p_k$;
  $\tilde{p}_{k+1} = \tilde{r}_{k+1} + \beta_k \tilde{p}_k$;
  $k = k + 1$;

Algorithm Limitations

- **Numerical Stability**: Susceptible to breakdowns (division by zero in α_k, β_k)
- **Convergence**: May stagnate or converge irregularly compared to GMRES or CGS
- **Matrix Operations**: Requires both A and A^T — not ideal for matrix-free solvers without transpose access
- **Preconditioners**: Designing effective preconditioners for both A and A^T is challenging

DeepMINRES

Motivation

- Traditional MINRES is effective for symmetric indefinite systems but suffers from slow convergence
- Deep learning offers potential to accelerate Krylov subspace methods
- Need for symmetry-preserving methods that still allow learning-based enhancements
- DeepMINRES leverages neural networks to learn better search directions while maintaining symmetry properties

DeepMINRES Algorithm

Algorithm 3: DeepMINRES Solver

Input : Symmetric matrix A, RHS b, initial guess x_0, pretrained model G_θ

Output: Approximate solution x

$r_0 = b - Ax_0$;

$q_0 = r_0 / \|r_0\|$;

for $k = 1, 2, \ldots, K$ **do**

  Run one Lanczos iteration to obtain q_k;

  $\alpha_k, \beta_k, \gamma_k = G_\theta(q_k, q_{k-1}, q_{k-2}, r_k, r_{k-1}, r_{k-2})$;

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

  $\eta_k = \arg\min_{\eta} \|A(x_k + \eta p_k) - b\|^2$;

  $x_{k+1} = x_k + \eta_k p_k$;

  Update residual $r_{k+1} = b - Ax_{k+1}$;

  if $\|r_{k+1}\| < \epsilon$ **then**

    break;

Analysis of DeepMINRES

Loss Function and Self-supervised Learning

- DeepMINRES trains a neural network to approximate optimal search directions for MINRES.
- Predicts coefficients α, β, γ for combining q_k, q_{k-1}, and q_{k-2}.
- "Ground truth" coefficients computed by solving small-scale minimization problems during training.
- Loss minimizes the deviation between predicted and optimal coefficients.
- Training data includes matrices with various condition numbers (easy/medium/hard).
- This enables robust generalization to unseen linear systems.

Model Architecture

- Inputs: three Krylov vectors and residuals — $(q_k, q_{k-1}, q_{k-2}, r_k, r_{k-1}, r_{k-2})$.
- Network: fully connected layers with ReLU activations.
- Output layer predicts combination weights $(\alpha_k, \beta_k, \gamma_k)$.
- Designed to process high-dimensional linear algebra structures efficiently.
- The architecture learns structure-agnostic strategies to enhance convergence across problems.

Training DeepMINRES

- **Data generation**: 2000 training samples ensure coverage and prevent overfitting
- **Train-validation split**: 80-20 split for monitoring generalization
- **Early stopping**: Patience of 5 epochs to avoid overfitting
- **Learning rate scheduling**: Reduce LR by 10% after epoch 10
- **Checkpointing**: Saves best model based on validation loss
- **Batch size optimization**: Batch size 64 balances speed and generalization
- **Efficiency**: Often converges in under 50 epochs due to good feature-model alignment

Model Evaluation and Testing

- **Test problems**: Uses both synthetic systems and random matrices with varying condition numbers
- **Metrics**: Tracks iterations to convergence, solution time, and relative error
- **Visualization**: Convergence plots demonstrate superior performance vs. standard MINRES
- **Robustness**: Validated across easy, medium, and hard problem regimes
- **Success metric**: Reports % reduction in iteration count as primary indicator

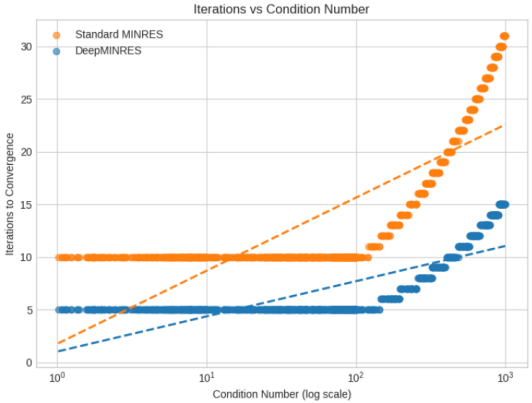
Practical Implementation Details

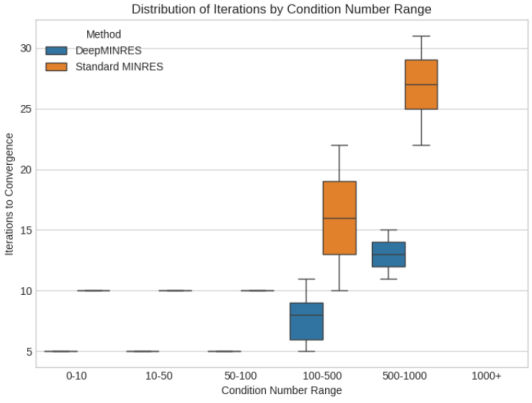
- **Reorthogonalization**: Maintains numerical stability during vector updates
- **Fallback mechanisms**: Reverts to standard MINRES if NN prediction is unstable
- **Step size**: Optimal step size η_k computed analytically per iteration
- **Feature normalization**: Residuals are normalized to aid learning
- **Adaptive sampling**: Emphasizes training on cases where NN guidance helps most

Effect of Condition Number on Convergence

- **Condition number** κ: Higher κ leads to slower convergence for both methods
- **DeepMINRES advantage**: Maintains significant iteration reduction, especially when $\kappa > 100$
- **Iteration reduction**: Achieves $\sim$50% fewer iterations compared to standard MINRES
- **Convergence speedup**: Over $2\times$ speedup observed for ill-conditioned systems
- **Residual norms**: Faster decay across all tested κ, confirming robust performance

Convergence comparison between DeepMINRES and Standard MINRES





The End