Fast Linear Solvers via AI-Tuned Markov Chain Monte Carlo-based Matrix Inversion

Anton Lebedev*
Won Kyung Lee*
anton.lebedev@stfc.ac.uk
wonkyung.lee@stfc.ac.uk
STFC Hartree Centre
Warrington, United Kingdom

Soumyadip Ghosh
IBM Research
Yorktown Heights, New York, USA
ghoshs@us.ibm.com

Olha I. Yaman
STFC Hartree Centre
Warrington, United Kingdom
olha.ivanyshyn-yaman@stfc.ac.uk

Vassilis Kalantzis
IBM Research
Yorktown Heights, New York, USA
vkal@ibm.com

Yingdong Lu
IBM Research
Yorktown Heights, New York, USA
yingdong@us.ibm.com

Tomasz Nowicki
IBM Research
Yorktown Heights, New York, USA
tnowicki@us.ibm.com

Shashanka Ubaru
IBM Research
Yorktown Heights, New York, USA
shashanka.ubaru@ibm.com

Lior Horesh
IBM Research
Yorktown Heights, New York, USA
lhoresh@us.ibm.com

Vassil Alexandrov
STFC Hartree Centre
Warrington, United Kingdom
vassil.alexandrov@stfc.ac.uk

Abstract

Large, sparse linear systems are pervasive in modern science and engineering, and Krylov subspace solvers are an established means of solving them. Yet convergence can be slow for ill-conditioned matrices, so practical deployments usually require preconditioners. Markov chain Monte Carlo (MCMC)-based inversion can generate such preconditioners and accelerate Krylov iterations, but its effectiveness depends on parameters whose optima vary across matrices; manual or grid search is costly. We present an AI-driven framework recommending MCMC parameters for a given linear system. A graph neural surrogate predicts preconditioning speed from A and MCMC parameters. A Bayesian acquisition function then chooses the parameter sets most likely to minimise iterations. On a previously unseen ill-conditioned system, the framework achieves better preconditioning with 50% of the search budget of conventional methods, yielding about a 10% reduction in iterations to convergence. These results suggest a route for incorporating MCMC-based preconditioners into large-scale systems.

CCS Concepts

• **Computing methodologies** → *Massively parallel and high-performance simulations*; *Neural networks*; • **Theory of computation** → *Numeric approximation algorithms.*

*Both authors contributed equally to this research.

SC '25, St Louis, MO, USA

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Keywords

Markov Chain Monte Carlo, MCMCMI, Numerical Linear Algebra, AI, Recommendation Systems

ACM Reference Format:
Anton Lebedev, Won Kyung Lee, Soumyadip Ghosh, Olha I. Yaman, Vassilis Kalantzis, Yingdong Lu, Tomasz Nowicki, Shashanka Ubaru, Lior Horesh, and Vassil Alexandrov. 2025. Fast Linear Solvers via AI-Tuned Markov Chain Monte Carlo-based Matrix Inversion. In *Proceedings of Research Posters of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '25).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Large and sparse linear systems $Ax = b$ are foundational across science and engineering, and Krylov solvers are typically paired with a preconditioner. MCMC matrix-inversion (MI) can construct an approximate inverse preserving the sparsity of the matrix while offering a high degree of embarrassing parallelism. However, performance is sensitive to a few MCMC parameters, and exhaustive searches are computationally expensive. Thus, we propose an AI-assisted framework that recommends MCMC parameters for a given sparse system. We use Bayesian Optimisation (BO) driven by a Graph Neural Network (GNN) surrogate that predicts the MCMC preconditioning performance, given a matrix and candidate parameter vector. A Bayesian acquisition function then proposes MCMC parameters that are both promising and informative.

2 Methodology

We consider $Ax = b$ with real, square A. An MCMC-based MI method takes an MCMC parameter vector x_{M} and returns a preconditioner $P \approx A^{-1}$. We then solve $PAx = Pb$, where the Krylov solver will typically converge faster due to the lower condition number of PA.

Algorithm 1 Bayesian tuning loop for MCMC parameter selection

Require: evaluated matrix set $\mathcal{A}_{\text{train}}$, total budget B, batch size k
  Initialise $\mathcal{D}_0$ with coarse grid-search records $(A, x_{\text{M}}, \bar{y}, s)$
  for $t = 0, 1, \ldots$ **until** $|\mathcal{D}_t| = B$ **do**
    Fit surrogate f_θ on $\mathcal{D}_t$
    for all $A \in \mathcal{A}_{\text{train}}$ **do**
      for $j = 1$ **to** k **do**
        draw initial $x_{\text{M}}^{(j,\text{init})}$
        $x_{\text{M}}^{(j)} \leftarrow$ **L–BFGS–B** maximise $\text{EI}(x_{\text{M}}; A)$ starting from $x_{\text{M}}^{(j,\text{init})}$
        Run MCMC + Krylov solver (e.g., GMRES) with $x_{\text{M}}^{(j)}$
        Record $(A, x_{\text{M}}^{(j)}, \bar{y}, s)$ and append to $\mathcal{D}_{t+1}$
      end for
    end for
  end for
  return $x_{\text{M}}^{\star}(A) = \arg\max_{x_{\text{M}}} \text{EI}(x_{\text{M}}; A)$ given $A \in \mathcal{A}$

Our surrogate receives three inputs (G, x_A, x_{M}). We form a directed, weighted graph G from A: nodes index rows, edges represent nonzeros with weight A_{ij}, and node degree is included as a node attribute. From G we compute a fixed-dimensional graph embedding via node-wise message passing with shared weights followed by permutation-invariant pooling, allowing the model to address matrices of varying size. We augment the embedding with low-cost matrix features x_A: 1-, Frobenius-, and infinity-norms; largest and smallest non-zero element magnitudes; sparsity; trace; and a symmetry flag. Separate Multi-Layer Perceptrons encoders embed x_A and x_{M}. The three embeddings are fused and passed to two linear heads that produce the predictive mean $\widehat{\mu}$ and the standard deviation $\widehat{\sigma}$ of the MCMC preconditioning performance metric

$$\text{y}(A, x_{\text{M}}) = \frac{\#\text{ of steps with preconditioner}}{\#\text{ of steps without preconditioner}}. \tag{1}$$

The surrogate is trained by minimising Mean-Squared Error to the sample mean and standard deviation from repeated solver runs.

For MCMC parameter selection we use BO with Expected Improvement (EI) to balance exploration and exploitation [2]. Because EI is differentiable with respect to x_{M}, we optimise it with L-BFGS-B from multiple random initialisations to produce a batch of recommendations. Algorithm 1 summarises the workflow.

3 Experiments

We benchmark the MCMC–based MI preconditioner of [1, 3] with tunables $x_{\text{M}} = (\alpha, \varepsilon, \delta)$: α diagonal scaling; ε sampling tolerance; δ truncation tolerance. The Krylov method (GMRES, BiCGStab; CG for SPD) is included as a categorical input but not optimised. We fix preconditioner fill $2\phi(A)$, where $\phi(A)$ is that of the matrix A, and drop tolerance 10^{-9}. The dataset comprises 11 sparse matrices, each run on a 64-point grid over x_{M} with 10 replicates; data are split 80/20 into training/validation and an unseen, ill-conditioned higher-order advection–diffusion test matrix. After one EI round (32 evaluations) and retraining, uncertainty calibration improves (Figure 1), with statistically meaningful coverage gains on the test matrix, and BO-enhanced recommendations yield ∼ 10% fewer iterations than grid search with 50% of the budget (Figure 2).

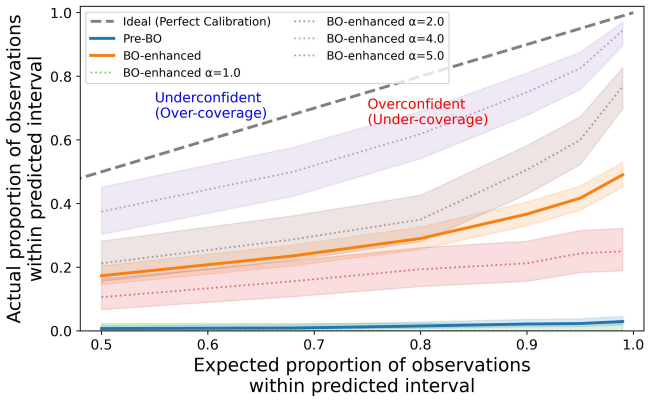

Figure 1: Calibration on the test matrix. Prediction intervals use $= \hat{\mu} \pm z_{(1+\tau)/2}\hat{\sigma}$ where τ indicates confidence levels in $\{0.50, 0.68, 0.80, 0.90, 0.95, 0.99\}$. Dashed grey: ideal; bands: Wilson 95% Confidence Itervals. Pre-BO under-covers; BO-enhanced approaches the diagonal, especially at larger α.

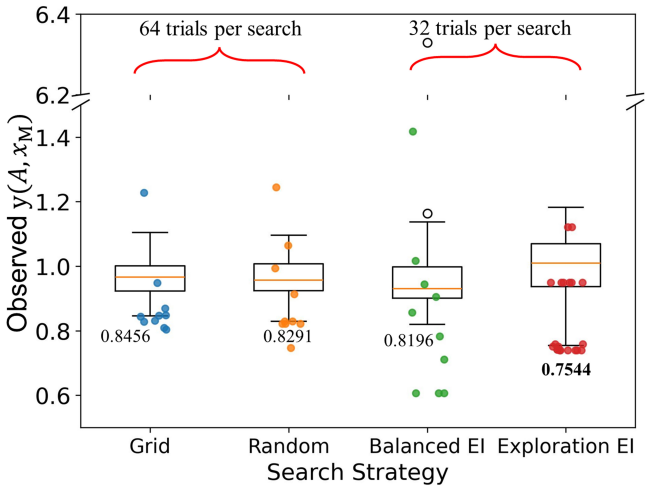

Figure 2: Box plot of sample median of $\text{y}(A, x_{\text{M}})$ over the explored x_{M}, including the minimum. Coloured circle points represent the distribution of the observed $\text{y}(A, x_{\text{M}}^*)$ over 10 replications, where x_{M}^* indicates the parameter yielding the minimum sample median.

4 Conclusions

We presented a BO framework coupling a GNN surrogate with EI to tune MCMC preconditioners. On a new ill-conditioned matrix it delivered 10% fewer Krylov iterations with 50% of the search budget. Next steps include jointly recommending the Krylov solver alongside MCMC parameters and optimising a time-to-solution objective in realistic HPC environments—GPU-accelerated and multi-node systems—by modelling latency and communication/memory overhead. We will also strengthen the surrogate's uncertainty estimates via deep kernels or scalable GPs, and develop acquisition scheme (cost-aware, batch, constrained).

Acknowledgments

This work was supported by the Hartree National Centre for Digital Innovation, a UK Government-funded collaboration between STFC and IBM.

References

[1] Anton Lebedev and Vassil Alexandrov. 2018. On advanced Monte Carlo methods for linear algebra on advanced accelerator architectures. In *2018 IEEE/ACM 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA)*. IEEE, 81–90.

[2] Jonas Mockus. 1998. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization* 2 (1998), 117.

[3] Emre Sahin, Anton Lebedev, Maksims Abaļenkovs, and Vassil Alexandrov. 2021. Usability of Markov chain Monte Carlo preconditioners in practical problems. In *2021 12th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA)*. IEEE, 44–49.

accepted 10 September 2025