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

Anton Lebedev[*1], Won Kyung Lee[*1], Soumyadip Ghosh[2], Olha I. Yaman[1], Vassilis Kalantz[2], Yingdong Lu[2], Tomasz Nowicki[2], Shashanka Ubaru[2], Lior Horesh[2], Vassil Alexandrov[1]

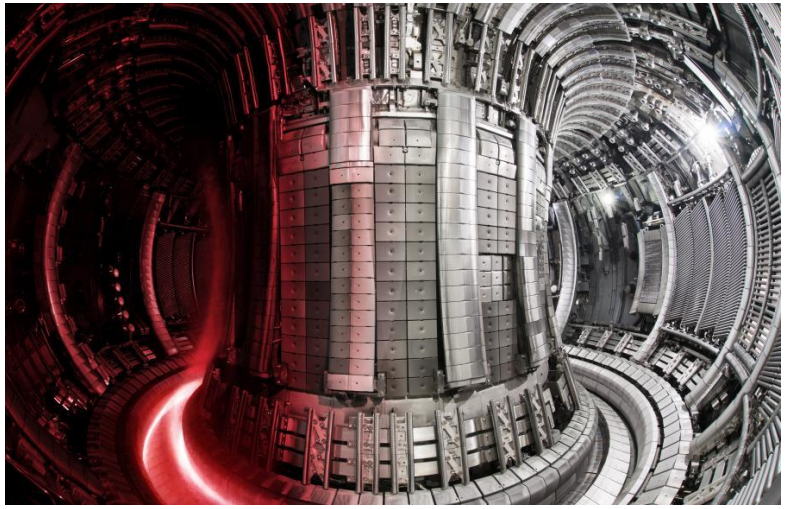
***: Equal contributions**

1: STFC Hartree Centre, UK

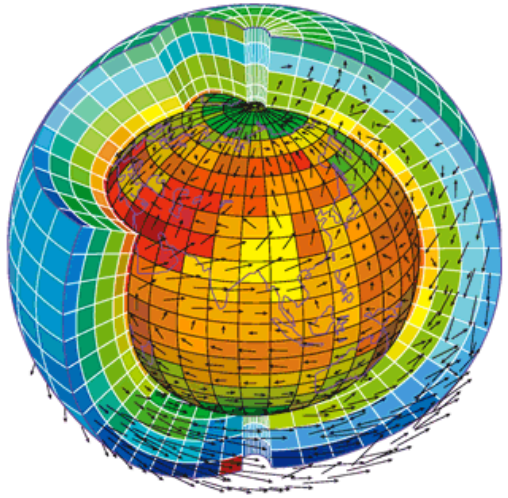
2: IBM Research, Yorktown Heights

ScalAH'25 Workshop 2025-11-16

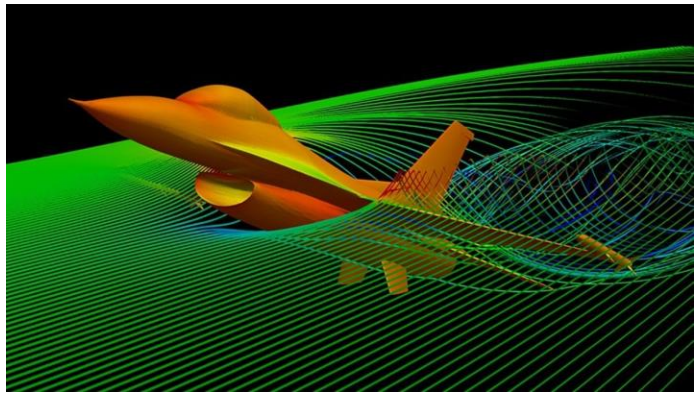
Linear Systems Ax=b are everywhere



Plasma simulation



Climate simulation



Fluid dynamics simulation

- Preconditioner P: Ax=b -> P Ax = P b $\text{s.t.}\, cond(PA) \ll cond(A)$
- Pain points

1) Huge system size
2) Numerically fragile: A tiny numerical change -> large change in x
   (ill-conditioned system)

- They make iterative methods like Krylov methods(BiCGstab, GMRES) very slow

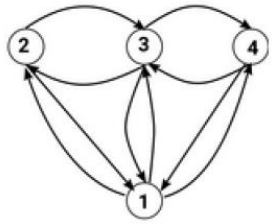
Preconditioning

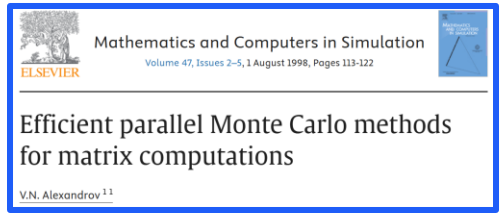
- Markov Chain Monte Carlo-based Matrix Inversion (MCMCMI)
  Let A=I-B, then Ax=b -> $x - Bx = \varphi, \quad \|B\| < 1$

$$(g,(I-B)^{-1}\varphi) = \underbrace{\sum_{r=0}^{n} g_r \left(\sum_{j=0}^{\infty} B^j \varphi \right)_r}_{\mathbb{E}(\Theta^*[g])} \approx \mathbb{E}(\Theta^i[g]) = \sum_{r=0}^{n} g_r \sum_{j=0}^{i} \sum_{k_1,\dots,k_j=1}^{n} p_{rk_1} \frac{b_{rk_1}}{p_{rk_1}} \dots p_{k_{j-1}k_j} \frac{b_{k_{j-1}k_j}}{p_{k_{j-1}k_j}} \varphi_{k_j}$$

Markov chain:
$k_0 \to k_1 \to \cdots \to k_j \to \cdots \to k_i$

$$\begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \end{pmatrix}$$





$$\left\| (I-B)^{-1} - \left(\sum_{j=0}^{i} B^j \right) \right\| \le \frac{\|B^{i+1}\|}{1-\|B\|} \le \boldsymbol{\delta}$$

$truncation\ error \implies$
$\boldsymbol{i - length\ of\ each\ trajectory}$

$$\mathbb{P}\left(\left| \frac{1}{N} \sum_{s=1}^{N} \Theta_s^i[g] - \mathbb{E}(\Theta^*[g]) \right| < \boldsymbol{\varepsilon} \right) \approx 1/2,$$

$stochastic\ error \implies$
$\boldsymbol{N - number\ of\ trajectories}$

$\alpha > 0$: diagonal shift ensuring Neumann–series convergence

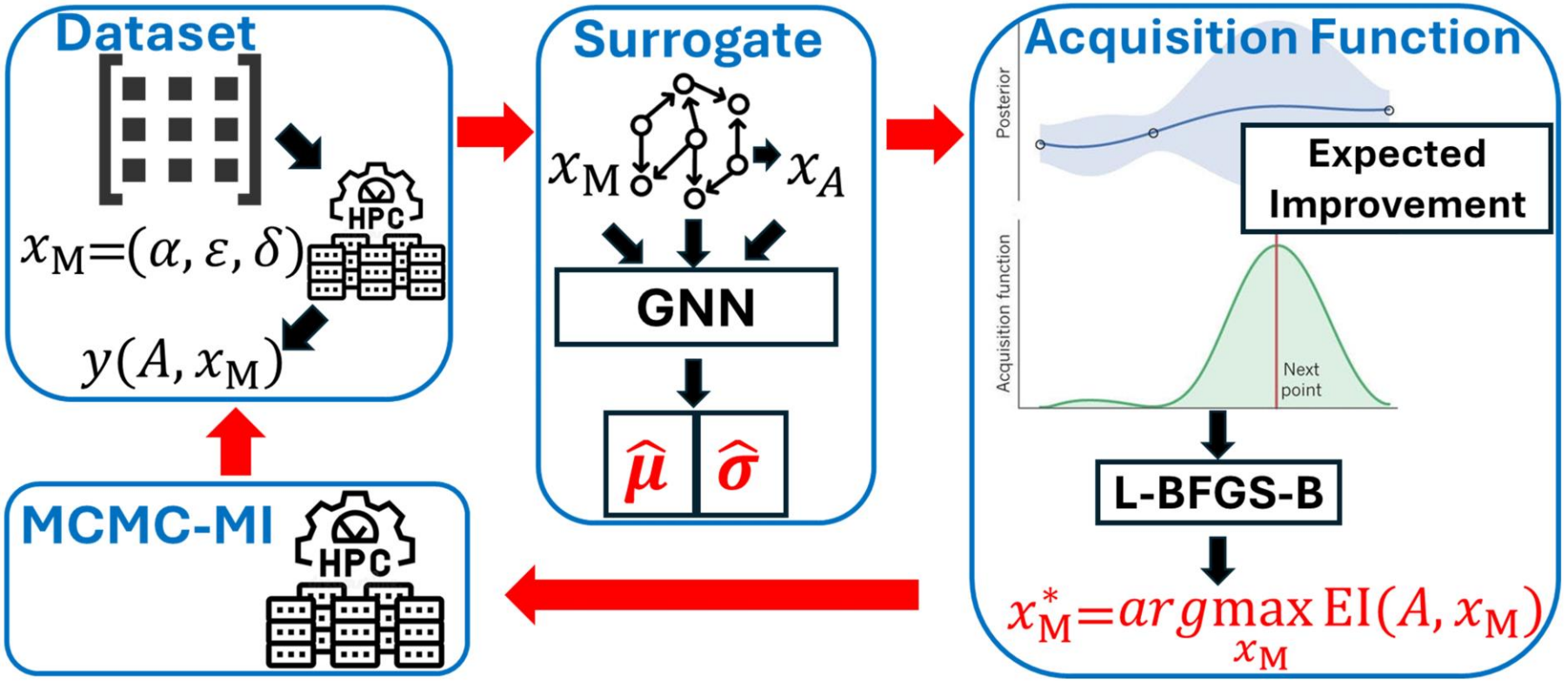
$\varepsilon \in (0,1]$: sampling tolerance (controls effective chain count)

$\delta \in (0,1]$: truncation tolerance (caps maximum walk length)

Alexandrov, V. N. (1998). Efficient parallel Monte Carlo methods for matrix computations. *Mathematics and computers in Simulation*, *47*(2-5), 113-122.

Sahin, E., Lebedev, A., Abaļenkovs, M., & Alexandrov, V. (2021, November). Usability of Markov chain Monte Carlo preconditioners in practical problems. In *ScalAH'21*.

AI-Assisted MCMC-MI

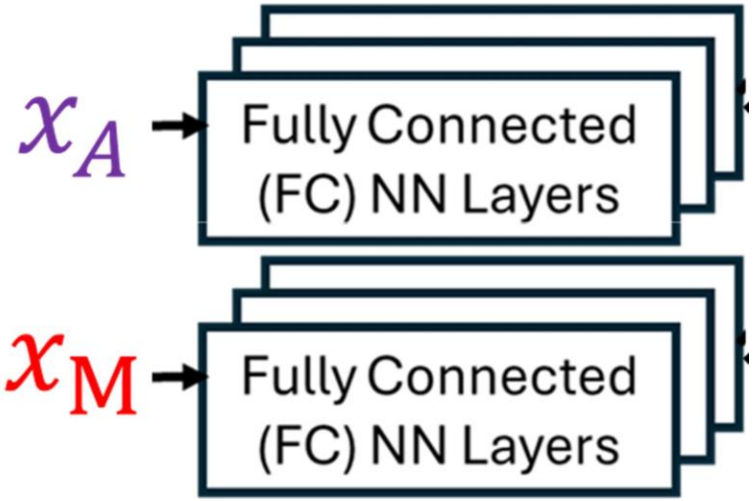


$$y(A, x_M) = \frac{\#\text{ of steps with preconditioner}}{\#\text{ of steps without preconditioner}}$$

AI-Assisted MCMC-MI

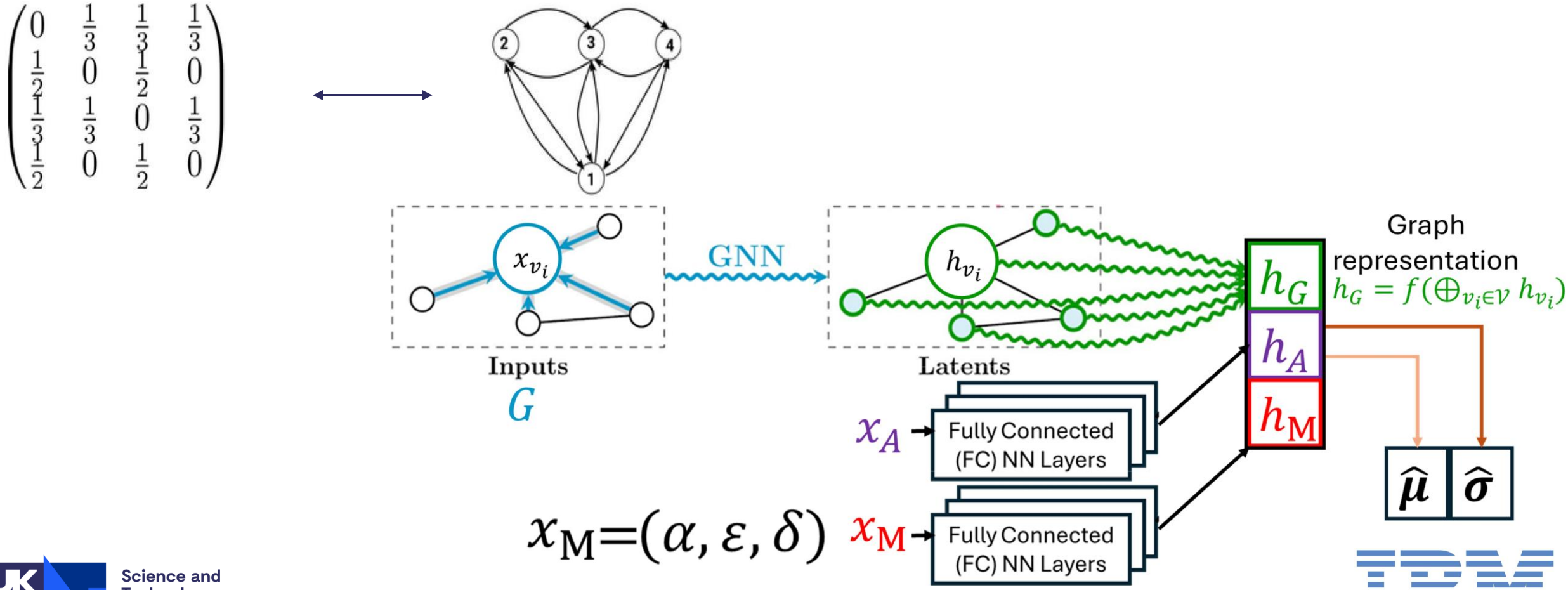
- Matrix features
- 1-norm
- Frobenius-norm
- Infinity-norm
- Sparsity
- Trace
- Symmetricity
- Largest element magnitude
- Smallest non-zero element magnitude

- $x_{\mathrm{M}}=(\alpha, \varepsilon, \delta)$



Representation of Matrix A and Surrogate Model

- Matrix -> Graph -> Graph Neural Networks -> Vector in latent space

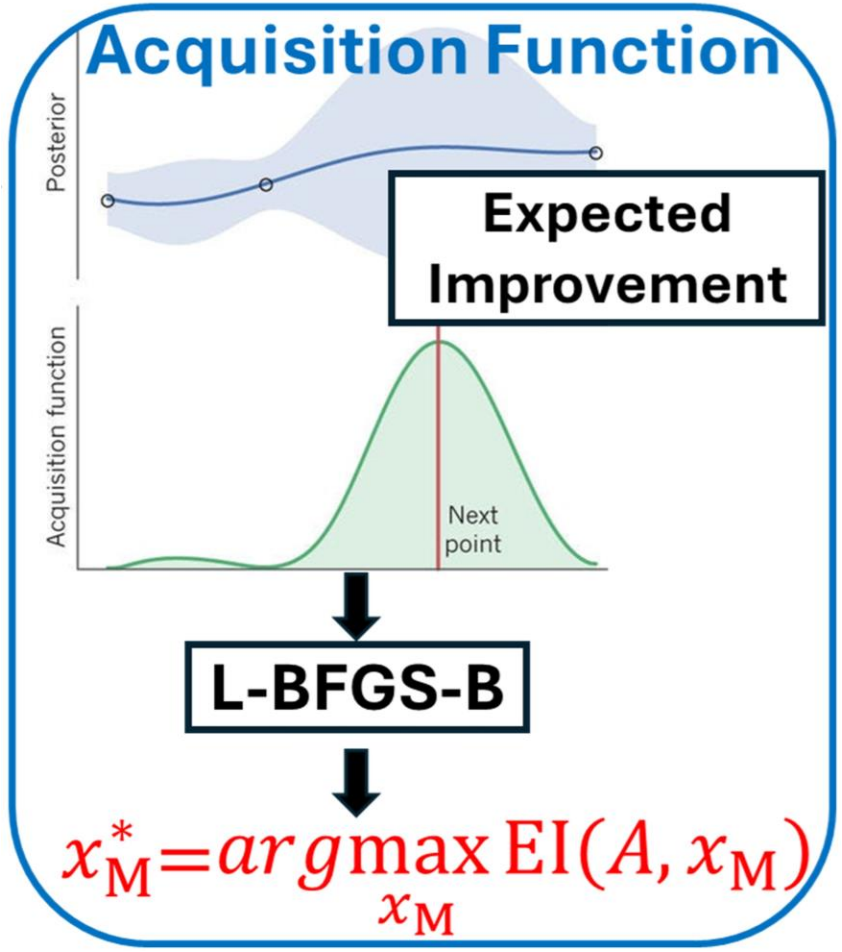


Acquisition Function

- Expected Improvement
- Weighting uncertain regions (Exploration)

- Weighting points to be predicted to perform well (Exploitation)

- Finding balance between them



Datasets

- From classical PDE discretisations
- Finite-difference Laplacians
- Finite-element operators

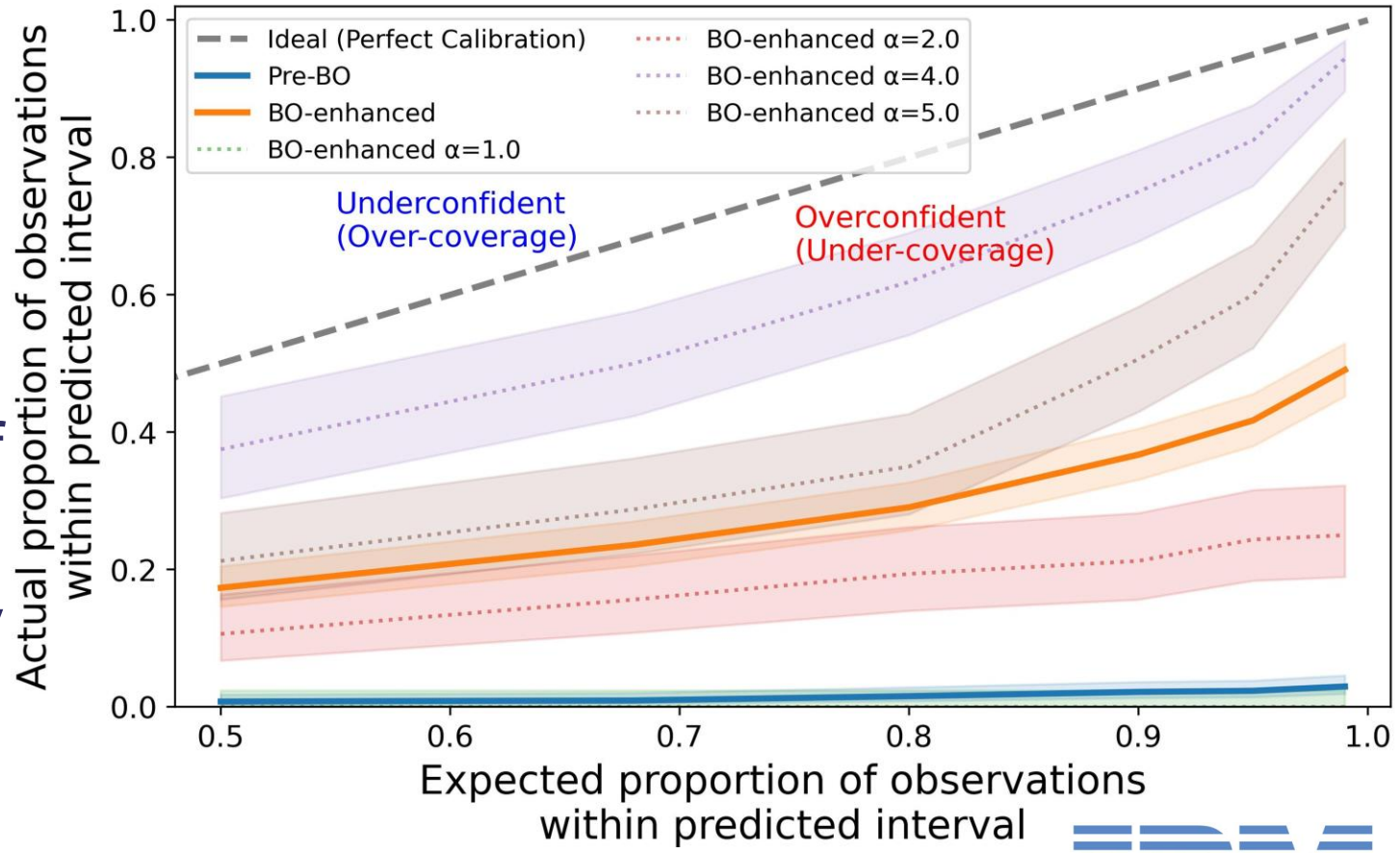
- From plasma physics
- From Advection-diffusion system
- From climate modelling

Matrix	Dimension	Symmet-ricity	$\kappa(A)$	$\phi(A)$
2DFDLaplace_16	225	Yes	1.0×10^2	0.042
2DFDLaplace_32	961	Yes	4.1×10^2	0.001
2DFDLaplace_64	3,969	Yes	1.7×10^3	0.0024
2DFDLaplace_128	16,129	Yes	6.6×10^3	0.0006
nonsym_r3_a11	20,930	No	1.9×10^4	0.0044
a00512	512	No	1.9×10^3	0.059
a08192	8,192	No	3.2×10^5	0.0007
unsteady_adv_diff_order1_0001	225	No	4.1×10^6	0.646
unsteady_adv_diff_order2_0001	225	No	6.6×10^6	0.646
PDD_RealSparse_N64	64	No	1.3×10^1	0.1
PDD_RealSparse_N128	128	No	5.0	0.1
PDD_RealSparse_N256	256	No	7.0	0.1

Validation

- Testing ill-conditioned advection-diffusion matrix
- Unseen during training

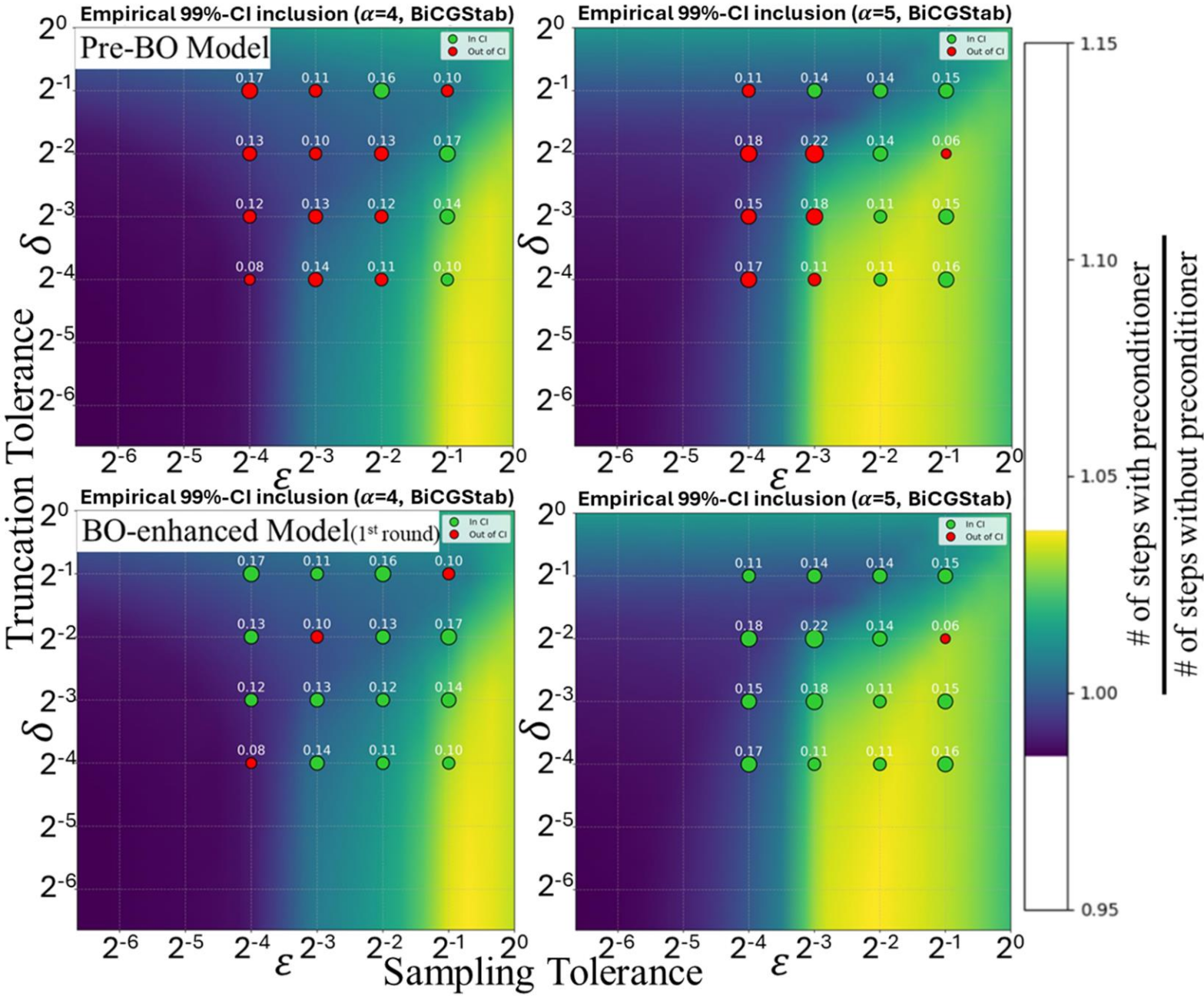
- Pre-BO: surrogate model trained on training set
- BO-enhanced: updated through 1st round of BO from Pre-BO
- Prob. of observed values lay within predictive C.I.



Validation

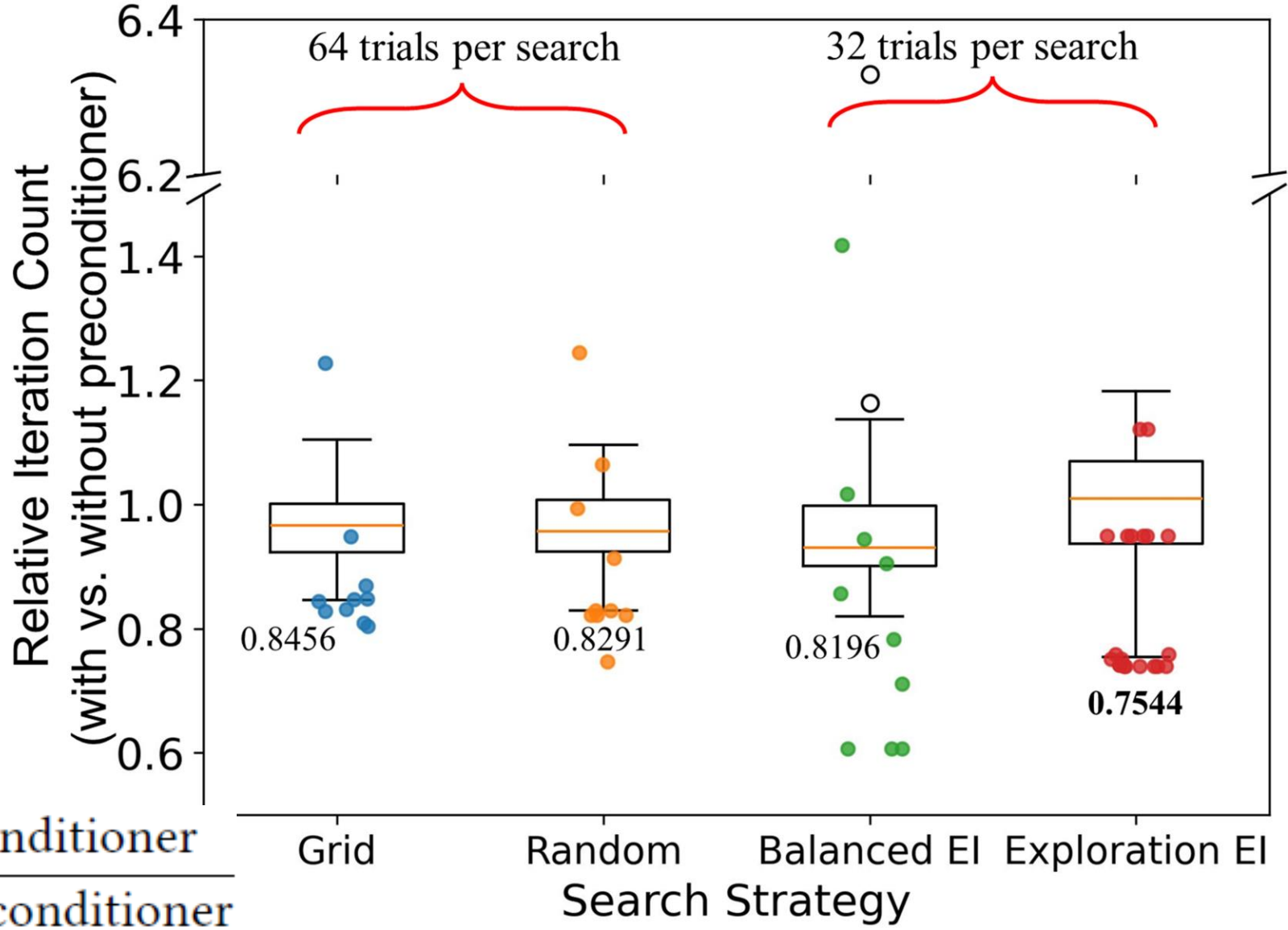
- Testing ill-conditioned advection-diffusion matrix
- Unseen during training

- Prob. of predicted values lay within empirical C.I.



Validation

- With 50% of search budgets, BO finds better MCMCMI parameters delivering ~10% fewer Krylov iterations



$$y(A, x_M) = \frac{\#\text{ of steps with preconditioner}}{\#\text{ of steps without preconditioner}}$$

Science and Technology Facilities Council
Hartree Centre

IBM

Next Steps

- Optimise time-to-solution in realistic HPC environments
  - GPU-accelerated and multi-node systems
  - Considering latency, communication & memory overheads

- Strengthen surrogate model with deep kernels or scalable GPs

- Upgrade acquisition scheme
  - Cost-aware
  - Batch
  - Constrained
- Active learning loop with GenAI (Exploring informative A)

Science and Technology Facilities Council

Hartree Centre

IBM

Paper

Poster

- Reception 1715-1900 18 Nov

Thank you

Wonkyung.lee@stfc.ac.uk