# RINS: RESIDUAL-IMAGE NEURAL SUBSPACE SOLVERS FOR LARGE SPARSE LINEAR SYSTEMS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Neural subspace solvers offer a promising way to accelerate the large sparse linear systems arising from PDE discretizations, but their value depends on generating effective correction directions at low computational overheads. We study this problem from a residual-image viewpoint: a correction subspace is useful when its image under the system operator captures the current residual. Guided by this view, we propose Gate-RINS, which generates multiple correction directions from shared polynomial residual probes and modulates them with a lightweight residual- and coordinate-dependent pointwise gate. We also introduce a hybrid schedule that uses GRANS-style graph controllers for the first correction block and Gate-RINS thereafter, combining an expressive graph based initial step with a lower-cost corrector as the solve progress. Theoretical analysis establishes a unified residual-image and dynamic approximate-inverse perspective for Gate-RINS and G-RANS, showing that a single neural-generated direction can emulate the residual quality of fixed-step GMRES. Numerical experiments on six PDE benchmarks demonstrate that Gate-RINS reaches a relative residual of $10^{-5}$ with 1.6-2.3$\times$ and 2.2-3.0$\times$ less wall-clock time than G-RANS and GMRES respectively. For Helmholtz system with meshsize up to $100,000$, Gate-RINS also achieves the lowest residual and outperforms a GPU-accelerated parallel GMRES implementation, while GRANS fails to reach relative residual of $10^{-4}$ in the same time budget.

## 1 INTRODUCTION

Solving large sparse linear systems is a fundamental computational task in PDE simulation (Knoll & Keyes, 2004), inverse modeling (Haber, 2014), and a broad range of scientific computing (Davis, 2006). Krylov methods are widely used for these systems, with prominent examples including conjugate-gradient methods (Hestenes & Stiefel, 1952), Lanczos-type iterations (Lanczos, 1950), Arnoldi/GMRES methods (Arnoldi, 1951; Saad & Schultz, 1986) and nonsymmetric solvers (van der Vorst, 1992; Freund & Nachtigal, 1991). These methods construct search space through matrix-vector products and compute approximate solutions through low-dimensional projection procedures (Greenbaum, 1997; Golub & Van Loan, 2013; Saad, 2003; Simoncini & Szyld, 2007), making them particularly attractive for large-scale sparse problems (Saad, 2003; Greenbaum, 1997; Simoncini & Szyld, 2007). Their efficiency depends on both how rapidly the search space captures effective correction directions and the computational cost of constructing and maintaining that space. Classical techniques such as restarting and deflation (Morgan, 2002), recycling (Parks et al., 2006), multigrid (Hackbusch, 1985; Xu, 1992) and preconditioning (Benzi, 2002; Saad, 1994; Elman et al., 2014) seek to improve efficiency by controlling search-space growth or exploiting problem structure to accelerate convergence. However, their effectiveness often relies on problem-specific design and parameter choices, motivating data-driven approaches that learn to adapt solver components to the problems being solved.

Recent advances in artificial intelligence for science (AI4Sci) (Wang et al., 2023; Karniadakis et al., 2021) offer opportunities to exploit structure in PDEs through learning. Physics-informed (Raissi et al., 2019; Karniadakis et al., 2021) and variational neural methods (E & Yu, 2018; Kharazmi et al., 2021) approximate PDE solutions by optimizing differential-equation residuals or variational objectives, while neural operators (Li et al., 2021; Lu et al., 2021; Kovachki et al., 2023) learn solution maps across families of PDEs. Despite their promise, directly optimizing neural solution



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

representations can be challenging for stiff or multiscale problems (Wang et al., 2021; Krishnapriyan et al., 2021; Grossmann et al., 2024). Moreover, a learned solution prediction alone does not ensure that a discretized system satisfies a prescribed residual tolerance (Grossmann et al., 2024; Zhang et al., 2024). These considerations motivate integrating learning into numerical iterations (Greenfeld et al., 2019; Zhang et al., 2024; Liao et al., 2026), where learned components can exploit problem structure while explicit residual evaluation guides solution refinement.

Within this setting, neural subspace methods learn to generate search directions while retaining a numerical projection to determine the solution update. This division assigns complementary roles to learning and numerical computation: the neural model proposes a correction space, and the projected solve selects a correction within that space. G-RANS (Liao et al., 2026) instantiates this approach using graph attention Veličković et al. (2018) to generate residual-dependent basis vectors from matrix, coordinate, and residual information. It thereby makes subspace construction adaptive to the current solver state. However, the resulting acceleration depends on both the quality of the generated directions and their computational cost. Graph-based basis generation introduces message-passing overhead that must be offset by improved convergence. In addition, a controller trained predominantly on early residual states may become less effective as the iteration progresses and the residual is increasingly dominated by components that are difficult to eliminate.

To understand what makes a learned correction space effective, consider a minimum-residual update with current residual $\mathbf{r}_t = \mathbf{b} - \mathbf{A}\mathbf{x}_t$ and correction basis $\mathbf{Q}_t$. The update takes the form $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{Q}_t\mathbf{y}_t$, where $\mathbf{y}_t \in \arg\min_{\mathbf{y}} \|\mathbf{r}_t - \mathbf{A}\mathbf{Q}_t\mathbf{y}\|_2$. The attainable residual reduction is therefore determined by the distance from $\mathbf{r}_t$ to the operator-image subspace $\mathrm{range}(\mathbf{A}\mathbf{Q}_t)$. This observation provides a criterion for evaluating learned directions: their value lies in how effectively their images under $\mathbf{A}$ represent the current residual. Fig. 1 illustrates how different correction spaces can lead to different projected residuals through their operator images. The resulting design challenge is to construct expressive, residual-dependent correction spaces at a sufficiently low cost to improve overall solution time.

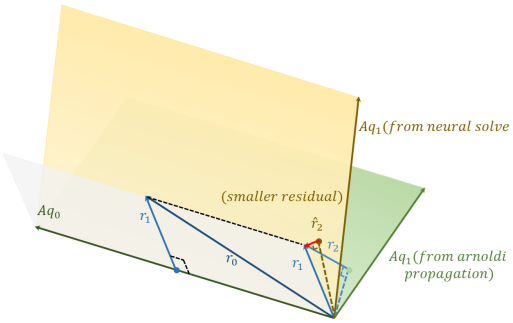



Figure 1: A neural solver can generate a correction space whose operator image leads to a smaller projected residual than an Arnoldi-generated image space.

Guided by this residual-image perspective, we propose Gate-RINS, a neural subspace solver that combines polynomial residual probes with lightweight pointwise modulation. The solver constructs a shared set of probes using sparse matrix-vector products and generates multiple correction directions through learnable combinations and a residual- and coordinate-dependent gate. The polynomial probes provide operator-dependent information, while the gate introduces nonlinear adaptation to the current residual. Sharing the probes across directions keeps the number of sparse operator applications tied to the probe order, rather than requiring a separate sequence of operator applications for each direction. The resulting basis is evaluated through the same projected minimum-residual update, allowing the learned component to focus on direction generation while the numerical solve determines the correction coefficients.

The changing residual structure over an iteration also motivates combining complementary basis generators. A graph-based controller can exploit matrix connectivity, coordinates, and residual information to produce effective initial corrections, whereas Gate-RINS provides a less expensive mechanism for subsequent basis generation. We therefore introduce a hybrid inference schedule that uses G-RANS for the first neural subspace block and Gate-RINS thereafter. Both controllers operate within the same projected solver, so the switch changes the basis-generation mechanism while preserving the update rule. Across six PDE-derived benchmark tasks, Gate-RINS reaches prescribed relative-residual thresholds faster than GMRES and the graph-based neural baseline in most evaluated settings, as measured by synchronized wall-clock time. Hybrid schedules provide further improvements in residual trajectories, while difficult-mode diagnostics and subspace visualizations support the residual-image interpretation of the observed gains.

Our contributions are summarized as follows:

Table 1: Comparison of learning-based sparse linear system solvers. "-" denotes that the neural model is not repeatedly invoked during iteration.

| Method | SPD | Learned Object | Model | Adaptive | Extra targets | Inference cost |
|---|---|---|---|---|---|---|
| **RINS (ours)** | ✗ | **Search-space basis** | **Gated polynomial** | ✓ | **None** | **Light-recurrent** |
| G-RANS | ✗ | Search-space basis | Graph attention | ✓ | None | High-recurrent |
| NeurKItt (Luo et al., 2024) | ✗ | Inv. subspace | FNO | ✗ | Eigenvectors | - |
| DCDM (Kaneda et al., 2023) | ✓ | CG direction | CNN | ✓ | Rayleigh–Ritz vectors | Medium-recurrent |
| GNP (Chen, 2025) | ✗ | Preconditioner | Graph convolution | ✗ | Solution pairs $(\mathbf{b}, \mathbf{x})$ | High-recurrent |
| NPO (Li et al., 2025) | ✓ | Preconditioner | Self-attention | ✗ | Solution pairs $(\mathbf{r}, \mathbf{x})$ | - |
| NeuralIF (Häusner et al., 2024) | ✓ | Preconditioner | Coates GNN | ✗ | None | - |

**1) Gate-RINS.** We propose a residual-image neural subspace solver that combines shared polynomial residual probes with residual- and coordinate-dependent pointwise gating. It generates multiple correction directions while keeping polynomial operator-probe cost.

**2) Hybrid controller.** We develop a schedule that uses G-RANS for the initial correction block and Gate-RINS thereafter under the same projected minimum-residual update. This combines an expressive graph-based initial correction with lower-cost polynomial basis generation.

**3) Theoretical analysis.** We interpret the directions generated by Gate-RINS and G-RANS as residual-adaptive inverse-like actions. Separately, we show that, under the stated assumptions, a sufficiently expressive neural generator can approximate the residual quality of fixed-step GMRES with a single correction direction.

**4) Empirical results.** Across six PDE-derived tasks, Gate-RINS reaches a relative residual of $10^{-5}$ with 1.6–2.3× and 2.2–3.0× speedups over G-RANS and GMRES, respectively, in mean wall-clock time. It also attains the lowest residual within the evaluated time budget on a separate larger Helmholtz system, while hybrid inference improves the residual trajectory on Poisson systems.

## 2 Relate Work

### 2.1 Scientific Machine Learning for PDEs

Scientific machine learning learns PDE solution fields or solution operators (Karniadakis et al., 2021). Physics-informed neural networks penalize differential-equation and boundary residuals (Raissi et al., 2019), while variational methods minimize energy functionals (E & Yu, 2018) and deep Galerkin methods use meshfree residual sampling (Sirignano & Spiliopoulos, 2018). Training difficulties include gradient imbalance (Wang et al., 2021) and failures on multiscale problems (Krishnapriyan et al., 2021), motivating pseudo-time stepping in TSONN (Cao & Zhang, 2023) and sequential correction in Scale-PINN (Chiu et al., 2026). Operator learning instead maps PDE inputs to solution fields through DeepONet (Lu et al., 2021), Fourier neural operators (Li et al., 2021), or reduced-basis-inspired representations (Bhattacharya et al., 2021); their approximation properties are studied within a function-space framework (Kovachki et al., 2023). Geometry-aware extensions include graph neural operators (Li et al., 2020a), multipole architectures (Li et al., 2020b), mesh-based simulators (Pfaff et al., 2021), and transformer-based models such as HAMLET (Bryutkin et al., 2024). Benchmarks such as PDEBench support evaluation across multiple PDE tasks (Takamoto et al., 2022), while comparisons with classical discretizations highlight remaining accuracy–cost trade-offs (Grossmann et al., 2024). These approaches complement RINS, which learns residual-conditioned correction spaces for an assembled linear system rather than directly representing its solution field or solution operator.

### 2.2 Neural-Iterative Solvers

Hybrid neural-numerical methods retain numerical structure while learning components such as discretizations (Bar-Sinai et al., 2019) or multigrid corrections (Greenfeld et al., 2019). As summarized in Table 1, learned linear solvers differ in what they predict and how often neural inference is used. Some learn approximate inverse operators or factorizations: NeuralIF predicts an incomplete factorization reused throughout preconditioned conjugate gradients (Häusner et al., 2024); NPO learns a matrix-dependent preconditioner incorporating multigrid structure (Li et al., 2025); and GNP applies a nonlinear graph-neural preconditioner recurrently within FGMRES (Chen, 2025). Other methods modify the search space itself. DCDM predicts residual-dependent conjugate directions at each iteration (Kaneda et al., 2023), whereas NeurKItt predicts an invariant subspace once to augment subsequent Krylov iterations (Luo et al., 2024). G-RANS generates residual-adaptive blocks

of search directions (Liao et al., 2026) using graph attention (Veličković et al., 2018), requiring recurrent graph inference followed by orthogonalization and projection. Polynomial approximation (Trefethen, 2019) and localized spectral filters (Defferrard et al., 2016) provide a less expensive operator-aware alternative for basis generation. RINS retains residual-adaptive block generation and the projected minimum-residual closure, but replaces graph propagation with cached polynomial probes and lightweight pointwise modulation, amortizing a small number of SpMVs over multiple generated directions.

# 3 METHODOLOGY

In this section, we first describe the GMRES iteration and its residual behavior, then extend fixed Krylov expansion to neural-controlled expansion of the search space and residual-image space. We use $\mathbf{Q}_t$ for an orthonormal basis matrix, $\mathbb{Q}_t = \mathrm{range}(\mathbf{Q}_t)$ for the corresponding search subspace, $\mathbf{P}_t = \mathbf{Q}_t\mathbf{Q}_t^\top$ for the orthogonal projector onto $\mathbb{Q}_t$, and $\mathbf{P}_{\mathbf{AQ}_t}$ for the orthogonal projector onto $\mathrm{range}(\mathbf{AQ}_t)$. RINS instantiates this view with a practical trade-off between SpMVs and approximation capacity: it builds polynomial probes from the current residual $\mathbf{r}_t$, forms candidate correction directions, and, in Gate-RINS, modulates them by a pointwise MLP using local probe values and coordinates. Finally, we describe a hybrid inference framework that composes a GRANS prefix with a shared RINS-family controller for later steps under the same projected numerical closure.

## 3.1 EXTENDING THE SUBSPACE EXPANDING DYNAMICS

**Discrete Optimal Control perspective of krylov iteration.** Let $\mathbf{Ax} = \mathbf{b}$, $\mathbf{A} \in \mathbb{R}^{n\times n}$. At iteration $t$, the residual is $\mathbf{r}_t = \mathbf{b} - \mathbf{Ax}_t$. GMRES searches the affine Krylov space $\mathbf{x}_0 + \mathbb{K}_t(\mathbf{A}, \mathbf{r}_0)$, where $\mathbb{K}_t(\mathbf{A}, \mathbf{r}_0) = \mathrm{span}\{\mathbf{r}_0, \mathbf{Ar}_0, \ldots, \mathbf{A}^{t-1}\mathbf{r}_0\}$. The residual bound of GMRES at iteration step $t$ is $\|\mathbf{r}_t^{\mathrm{GMRES}}\|_2 = \min_{p\in\mathbb{P}_t, p(0)=1} \|p(\mathbf{A})\mathbf{r}_0\|_2$. Geometrically equivalently, $\|\mathbf{r}_t^{\mathrm{GMRES}}\|_2 = \|(\mathbf{I} - \mathbf{P}_{\mathbf{A}\mathbb{K}_t})\mathbf{r}_0\|_2$, where $\mathbf{P}_{\mathbf{A}\mathbb{K}_t}$ is the orthogonal projector onto the residual-image space $\mathbf{A}\mathbb{K}_t = \mathrm{range}(\mathbf{AK}_t)$ for any basis matrix $\mathbf{K}_t$ of $\mathbb{K}_t$; explicitly, $\mathbf{P}_{\mathbf{A}\mathbb{K}_t} = (\mathbf{AK}_t)((\mathbf{AK}_t)^\top\mathbf{AK}_t)^+(\mathbf{AK}_t)^\top$, and superscript $+$ denotes the Moore–Penrose pseudoinverse. Neural solvers extend this characterization to learned search subspaces (from $\mathbb{K}_t$ to $\mathbb{Q}_t$): $\|\mathbf{r}_t^{\mathrm{NS}}\|_2 = \|(\mathbf{I} - \mathbf{P}_{\mathbf{AQ}_t})\mathbf{r}_0\|_2$. We view the search-space projector $\mathbf{P}_t$ as the state of the iteration and write its dynamics as $\mathbf{P}_{t+1} = \mathbf{P}_t + \mathbf{C}_t(\mathbf{C}_t^\top\mathbf{C}_t)^+\mathbf{C}_t^\top$, where $\mathbf{C}_t = (\mathbf{I} - \mathbf{P}_t)\mathbf{B}_t$ is the component of the newly generated block $\mathbf{B}_t$ orthogonal to the current search subspace $\mathbb{Q}_t$.

**Instantiate as GMRES.** $\mathbf{B}_t \in \mathbb{R}^{n\times 1}$ is fixed for GMRES before orthogonalization by applying $\mathbf{A}$ to the latest Arnoldi basis vector, once $\mathbf{A}$ and $\mathbf{r}_0$ are given. For neural solvers, $\mathbf{B}_t$ is generated from the current residual $\mathbf{r}_t$, the matrix $\mathbf{A}$, optional coordinates $\mathbf{X}_{\mathrm{coord}}$, and trained parameters $\boldsymbol{\theta}$. It can contain $m$ columns: $\mathbf{B}_t \in \mathbb{R}^{n\times m}$, while requiring fewer than $m$ expensive SpMV-equivalent propagation modules for basis generation. The parameters $\boldsymbol{\theta}$ encode problem-family information that helps the solver generate a residual-image subspace closer to the current residual.

**Instantiate as GRANS.** Specifically, GRANS uses a learned poly-GAT generator to construct a residual-dependent block $\mathbf{B}_t$ and orthogonalizes it with the accumulated basis to get $\mathbf{C}_t$: $\mathbf{B}_t = \mathrm{GAT}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{r}_t, \mathbf{X}_{\mathrm{coord}}), \mathbf{Q}_t = \mathrm{orth}\big([\mathbf{Q}_{t-1}, \mathbf{C}_t]\big) = [\mathbf{Q}_{t-1}, \mathrm{orth}(\mathbf{C}_t)]$. With $l$ message-passing layers and a $q$-direction head at each layer, GRANS emits $ql$ directions per neural iteration, namely $m = ql$. This amortizes graph propagation over many directions, but each attention layer still evaluates learned edge weights, feature projections, nonlinearities, and aggregation, so it is usually more expensive than $l$ times SpMV on the same sparsity graph.

## 3.2 RINS POLYNOMIAL BASIS GENERATOR

RINS follows the same subspace expanding dynamics above, but specifies the neural basis block $\mathbf{B}_t$ by a polynomial residual-image generator rather than by graph message passing. At iteration $t$, the goal is to construct a multi-column block $\mathbf{B}_t \in \mathbb{R}^{n\times m}$ from the current residual $\mathbf{r}_t$, so that the orthogonalized increment $\mathbf{C}_t = (\mathbf{I}-\mathbf{P}_t)\mathbf{B}_t$ expands $\mathbb{Q}_t$ toward a residual image space $\mathrm{range}(\mathbf{AQ}_{t+1})$ whose projector $\mathbf{P}_{\mathbf{AQ}_{t+1}}$ leaves a smaller projected residual. The key design constraint is that $\mathbf{B}_t$ should contain several useful directions while requiring fewer than $m$ expensive operator-probe modules for basis generation.

**Instantiate as Static-RINS.** In the default implementation, RINS uses Chebyshev residual probes. It first scales the matrix as $\widetilde{\mathbf{A}} = \mathbf{A}/s$, where $s$ is a cheap maximum absolute row-sum scale. Starting from $\mathbf{r}_t$, define $\mathbf{c}_t^{(0)} = \mathbf{r}_t, \mathbf{c}_t^{(1)} = \widetilde{\mathbf{A}}\mathbf{r}_t, \mathbf{c}_t^{(\ell)} = 2\widetilde{\mathbf{A}}\mathbf{c}_t^{(\ell-1)} - \mathbf{c}_t^{(\ell-2)}, \ell = 2, \ldots, l$. These vectors form a cached residual-probe dictionary $\mathbb{C}_t^{(l)}(\mathbf{A}, \mathbf{r}_t) = \mathrm{span}\{\mathbf{c}_t^{(0)}, \mathbf{c}_t^{(1)}, \ldots, \mathbf{c}_t^{(l)}\}$. It contains $l + 1$

probe vectors and requires $l$ SpMVs. Static-RINS chooses $\mathbf{B}_t$ from this dictionary by learnable polynomial filters. Its $j$-th emitted column is $\mathbf{b}_{t,j}^{\mathrm{static}} = \sum_{\ell=0}^{l} \alpha_{j\ell} \mathbf{c}_t^{(\ell)}, j = 1, \ldots, m$ with learnable coefficients $\alpha_{j\ell}$, and $\mathbf{B}_t^{\mathrm{static}} = [\mathbf{b}_{t,1}^{\mathrm{static}}, \ldots, \mathbf{b}_{t,m}^{\mathrm{static}}] \in \mathbb{R}^{n \times m}$. The resulting basis increment is $\mathbf{C}_t^{\mathrm{static}} = (\mathbf{I} - \mathbf{P}_t)\mathbf{B}_t^{\mathrm{static}}$, followed by QR orthogonalization and the same projected least-squares closure.

**Instantiate as Gate-RINS.** Gate-RINS enlarges this admissible set by adding a residual- and coordinate-dependent diagonal modulation before the block is appended to $\mathbb{Q}_t$ to avoid a degeneration when $m > l + 1$. For node $i$, let $\mathbf{z}_{t,i} = [c_{t,i}^{(0)}, c_{t,i}^{(1)}, \ldots, c_{t,i}^{(l)}, \mathbf{x}_i], i = 1, \cdots, n$. A small pointwise MLP computes $g_{t,ij} = 1 + s_g \tanh(\mathrm{MLP}_{\boldsymbol{\vartheta}}(\mathbf{z}_{t,i})_j), j = 1, \ldots, m$, and the gated block is $b_{t,ij}^{\mathrm{gate}} = g_{t,ij} \sum_{\ell=0}^{l} \alpha_{j\ell} c_{t,i}^{(\ell)}$. Equivalently, $\mathbf{b}_{t,j}^{\mathrm{gate}} = \mathbf{D}_{t,j} \mathbf{b}_{t,j}^{\mathrm{static}}$, where $\mathbf{D}_{t,j} = \mathrm{diag}(g_{t,1j}, \ldots, g_{t,nj})$. Thus Gate-RINS changes the admissible block $\mathbf{B}_t$ from a fixed polynomial span to a nodewise modulated polynomial span: $\mathbf{B}_t^{\mathrm{gate}} = [\mathbf{D}_{t,1}\mathbf{b}_{t,1}^{\mathrm{static}}, \ldots, \mathbf{D}_{t,m}\mathbf{b}_{t,m}^{\mathrm{static}}]$. This gives the controller nonlinear capacity to alter the search-space expansion path $\mathbb{Q}_t \mapsto \mathbb{Q}_{t+1}$, while the numerical update remains governed by the residual image projector $\mathbf{P}_{\mathbf{A}\mathbf{Q}_{t+1}}$.

Computing the order-$l$ dictionary requires $l$ SpMVs. Emitting $m$ columns of $\mathbf{B}_t$ then reuses the cached probes, so the dominant operator-probe cost of RINS basis generation is $l$. Gate-RINS adds only the pointwise MLP and diagonal modulation; it changes the admissible neural control family without increasing the dominant SpMV count for the probe dictionary.

**Sparse approximate inverse structure.** To better demonstrate the property discrepancy and iterative structure, we analyze these models from a unified approximate-inverse interpretation before orthogonalization. With reasonable simplification of ignoring input of coordinate, Gate-RINS equivalently emits basis preconditioned by learnable diagonal matrix: $\mathbf{b}_{t,j} = \mathbf{D}_{t,j} p_j(\widetilde{\mathbf{A}})\mathbf{r}_t$. Under the simplification that each node is represented by a scalar feature, a single GRANS propagation layer can be equivalently interpreted as a learned one-hop sparse matrix, with its entries determined by the attention mechanism, interleaved with feature transformations and nonlinearities. Multilayer propagation enables multi-hop interactions, allowing the model to go beyond the original sparsity pattern of $\mathbf{A}$. Block-GRANS restricts source-channel mixing to blocks, reducing source-projection parameters and restricting internal transport freedom. The minimum-residual closure selects an inverse-like correction from these generated directions, rather than requiring each direction to approximate $\mathbf{A}^{-1}\mathbf{r}_t$ individually. Appendix C formalizes this dynamic nonlinear preconditioning viewpoint and distinguishes internal channel commutativity from commutativity of the full graph operator.

### 3.3 Hybrid Models Bootstrap Training And Inference

**Towards non-parameter-sharing control.** The preceding derivation treats the new block $\mathbf{B}_t$ as the control that moves the search-space projector $\mathbf{P}_t = \mathbf{Q}_t\mathbf{Q}_t^\top$. A standard neural solver uses a shared controller for all steps, namely $\mathbf{B}_t = \mathcal{B}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{r}_t, \mathbf{X}_{\mathrm{coord}}), t = 0, 1, \ldots$, where $\mathcal{B}_{\boldsymbol{\theta}}$ denotes the controller and the columns of $\mathbf{B}_t$ are newly generated search directions. However, the controlled-expansion view does not require the same control law at every step. We therefore allow a step-wise controller sequence $\mathbf{B}_t = \mathcal{B}_{\boldsymbol{\theta}_t}^{(t)}(\mathbf{A}, \mathbf{r}_t, \mathbf{X}_{\mathrm{coord}}), \mathcal{B}_{\boldsymbol{\theta}_t}^{(t)} \in \{\mathrm{GRANS}, \mathrm{Gate\text{-}RINS}_{\boldsymbol{\theta}_1}, \mathrm{Gate\text{-}RINS}_{\boldsymbol{\theta}_2}, \ldots\}$. The standalone setting is the special case $\mathcal{B}_{\boldsymbol{\theta}_t}^{(t)} = \mathcal{B}_{\boldsymbol{\theta}}$ for all $t$. A hybrid model instead assigns controllers to solver stages, so that graph-attention and polynomial residual-image inductive biases can be used within the same projected solver.

**Multi-step bootstrap training.** Training is written in terms of frozen prefixes. Suppose controllers $0, \ldots, t-1$ have already been trained and fixed. Starting from $\mathbf{P}_0 = 0$, they generate prefix blocks $\mathbf{B}_0, \ldots, \mathbf{B}_{t-1}$, an accumulated basis matrix $\mathbf{Q}_t$, and the search subspace $\mathbb{Q}_t = \mathrm{range}(\mathbf{Q}_t)$. The residual exposed to the next controller is the prefix-projected residual $\mathbf{r}_t = (\mathbf{I} - \mathbf{P}_{\mathbf{A}\mathbf{Q}_t})\mathbf{r}_0, \mathbf{P}_{\mathbf{A}\mathbf{Q}_t} = \mathbf{A}\mathbf{Q}_t\big((\mathbf{A}\mathbf{Q}_t)^\top\mathbf{A}\mathbf{Q}_t\big)^+(\mathbf{A}\mathbf{Q}_t)^\top$. The new controller $\mathcal{B}_{\boldsymbol{\theta}_t}^{(t)}$ is trained to generate a block $\mathbf{B}_t$ whose orthogonalized increment $\mathbf{C}_t = (\mathbf{I} - \mathbf{P}_t)\mathbf{B}_t$ improves the next residual-image projector. Its bootstrap objective is $\ell_t(\boldsymbol{\theta}_t) = \min_{\mathbf{y}} \|\mathbf{A}\ \mathrm{orth}([\mathbf{Q}_t, \mathbf{B}_t])\mathbf{y} - \mathbf{r}_0\|_2^2, \mathbf{B}_t = \mathcal{B}_{\boldsymbol{\theta}_t}^{(t)}(\mathbf{A}, \mathbf{r}_t, \mathbf{X}_{\mathrm{coord}})$. For multi-step bootstrap training, this procedure is unrolled for a short horizon, but the accumulated basis $\mathbf{Q}_s$ and residuals are detached between steps for stability. Earlier controllers remain frozen, so the $t$-th model learns the residual distribution induced by the previously learned search-space trajectory rather than by the initial residual alone.

**Hybrid models inference.** This notation covers the hybrid schedules used in our experiments. In the first schedule, $\mathcal{B}^{(0)}_{\boldsymbol{\theta}_0}$ is a GRANS controller and $\mathcal{B}^{(t)}_{\boldsymbol{\theta}_G}$ for $t \geq 1$ is the same trained Gate-RINS controller reused at every later step; thus inference uses GRANS for the first block and shared Gate-RINS for all later blocks.

At inference time, standalone and hybrid models run the same cumulative projection loop. At step $t$, the active controller emits $\mathbf{B}_t$, the increment is orthogonalized as $\mathbf{C}_t = (\mathbf{I} - \mathbf{P}_t)\mathbf{B}_t$, $\mathbf{Q}_{t+1} = [\mathbf{Q}_t, \mathrm{orth}(\mathbf{C}_t)], \mathbf{P}_{t+1} = \mathbf{Q}_{t+1}\mathbf{Q}_{t+1}^\top$, optionally with truncation if a maximum basis budget is imposed. The numerical correction is then obtained from the residual-image least-squares problem $\mathbf{y}_t = \arg\min_{\mathbf{y}} \|\mathbf{A}\mathbf{Q}_{t+1}\mathbf{y} - \mathbf{r}_0\|_2, \mathbf{x}_{t+1} = \mathbf{x}_0 + \mathbf{Q}_{t+1}\mathbf{y}_t, \mathbf{r}_{t+1} = (\mathbf{I} - \mathbf{P}_{\mathbf{A}\mathbf{Q}_{t+1}})\mathbf{r}_0$. Thus hybridization changes the learned control sequence $\mathbf{B}_t$, but not the projected residual-image closure that evaluates every generated search space.

### 3.4 Theoretical Role of Fixed-Step GMRES Emulation

Why can a neural solver achieve a smaller residual using fewer basis vectors? We formalize this by asking whether $j$ neural-generated basis vectors can match the residual of $m$ GMRES steps with $j < m$. The most stringent case is $j = 1$, where a single generated direction is compared with an $m$-step Krylov subspace. Theorem A shows that, under compact and nondegenerate assumptions on a parameterized problem family, the $m$-step GMRES residual varies continuously with the parameter $\boldsymbol{\mu}$, such as PDE coefficients or geometry. A universal approximation argument then implies that a continuous neural generator emitting one direction can uniformly emulate this fixed-step GMRES residual quality. The detailed proof is given in Appendix A.

**Theorem 1** (Single-Direction Emulation of Fixed-Step GMRES)**.** *Let $\mathbb{D}$ be compact and let $\mathbf{A}(\boldsymbol{\mu}) \in \mathbb{R}^{n\times n}$ and $\mathbf{r}_0(\boldsymbol{\mu}) \in \mathbb{R}^n$ be continuous in $\boldsymbol{\mu} \in \mathbb{D}$. Assume that $\mathbf{A}(\boldsymbol{\mu})$ is uniformly nonsingular and that, for a fixed $m$, the matrix $\mathbf{Y}_m(\boldsymbol{\mu}) = \mathbf{A}(\boldsymbol{\mu})\mathbf{K}_m(\boldsymbol{\mu}), \mathbf{K}_m(\boldsymbol{\mu}) = [\mathbf{r}_0(\boldsymbol{\mu}), \mathbf{A}(\boldsymbol{\mu})\mathbf{r}_0(\boldsymbol{\mu}), \ldots, \mathbf{A}(\boldsymbol{\mu})^{m-1}\mathbf{r}_0(\boldsymbol{\mu})]$ has constant rank on $\mathbb{D}$. Let $\rho_m^{\mathrm{GMRES}}(\boldsymbol{\mu}) = \min_{\mathbf{y}} \|\mathbf{r}_0(\boldsymbol{\mu}) - \mathbf{Y}_m(\boldsymbol{\mu})\mathbf{y}\|_2 = \|\mathbf{r}_0(\boldsymbol{\mu}) - \mathbf{Y}_m(\boldsymbol{\mu})\mathbf{y}_m^*\|_2$, where $\mathbf{y}_m^* = \mathbf{Y}_m(\boldsymbol{\mu})^+\mathbf{r}_0$. For every $\varepsilon > 0$, there exists a continuous neural generator $\Phi_{\boldsymbol{\theta}}$ that maps the system data $(\mathrm{vec}\,\mathbf{A}(\boldsymbol{\mu}), \mathbf{r}_0(\boldsymbol{\mu}))$ to a correction vector $\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{\mu}) \in \mathbb{R}^n$ such that the one-dimensional projected update over $\mathrm{span}\{\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{\mu})\}$ satisfies $\min_{\alpha\in\mathbb{R}} \|\mathbf{r}_0(\boldsymbol{\mu}) - \alpha\mathbf{A}(\boldsymbol{\mu})\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{\mu})\|_2 \leq \rho_m^{\mathrm{GMRES}}(\boldsymbol{\mu}) + \varepsilon, \forall \boldsymbol{\mu} \in \mathbb{D}$.*

## 4 Experiments

We evaluate residual-reduction efficiency, and examine large-scale behavior, correction trajectories, spectral alignment and single-action approximate-inverse quality. Hybrid schedules and scale/geometry transfer tests further probe how the generator architecture affects solver behavior.

### 4.1 Main Benchmark

The main benchmark tests whether learned residual-adaptive basis generation improves residual-reduction efficiency over GMRES under a shared evaluation protocol.

**Tasks and solvers.** We evaluate RINS, GRANS, and GMRES on six PDE-induced sparse linear-system tasks: Poisson, Helmholtz, heat, advection-diffusion, Klein–Gordon, and wave; Appendix B gives the discretizations. Unless otherwise specified, RINS denotes Gate-RINS and Static-RINS denotes the ungated polynomial-filter ablation. A separate Helmholtz solve at $n \approx 100000$ is reported in Section 4.2 and Figure 3a. RINS and GRANS use the same projected minimum-residual update, with 200 neural iterations and 20 new basis vectors per iteration. The $n \approx 1000$ timing table uses CPU-optimized GMRES, which runs until the minimum neural-solver residual is reached; GPU GMRES was not faster at this scale in our measurements.

**Statistics and hardware.** Residual curves use a fixed training seed and report the median over 10 held-out samples, with shaded sample interquartile ranges (25th–75th percentiles). Timing tables pool all valid seed/sample time-to-threshold observations and report their pooled mean and standard deviation. No external preconditioner is used in this benchmark. Appendix D.1 reports separate integration experiments in which both training and inference use preconditioned data, demonstrating additional acceleration from combining RINS with preconditioning. All GPU neural-solver experiments use one NVIDIA H100 GPU.

Figure 2 and Table 2 show the advantage of RINS over GRANS and GMRES in rapidly reducing the residual across the different linear systems at $n \approx 1000$. Because RINS and GRANS add 20

Table 2: Time in seconds to reach relative residual $10^{-5}$ at $n \approx 1000$. Solver-time entries are the pooled mean and standard deviation over all valid seed/sample observations. The last two columns report the speedup of RINS over GRANS and GMRES, computed from pooled mean times.

| PDE | RINS | GRANS | GMRES | GRANS | GMRES |
|---|---|---|---|---|---|
| Poisson | $0.105 \pm 0.013$ | $0.219 \pm 0.041$ | $0.292 \pm 0.035$ | 2.09x | 2.79x |
| Helmholtz | $0.121 \pm 0.010$ | $0.194 \pm 0.036$ | $0.335 \pm 0.037$ | 1.61x | 2.77x |
| Heat | $0.058 \pm 0.004$ | $0.136 \pm 0.025$ | $0.128 \pm 0.019$ | 2.34x | 2.20x |
| Advection-Diffusion | $0.105 \pm 0.007$ | $0.232 \pm 0.024$ | $0.320 \pm 0.032$ | 2.20x | 3.04x |
| Klein-Gordon | $0.061 \pm 0.009$ | $0.120 \pm 0.014$ | $0.138 \pm 0.045$ | 1.97x | 2.27x |
| Wave | $0.061 \pm 0.008$ | $0.130 \pm 0.031$ | $0.146 \pm 0.045$ | 2.15x | 2.41x |

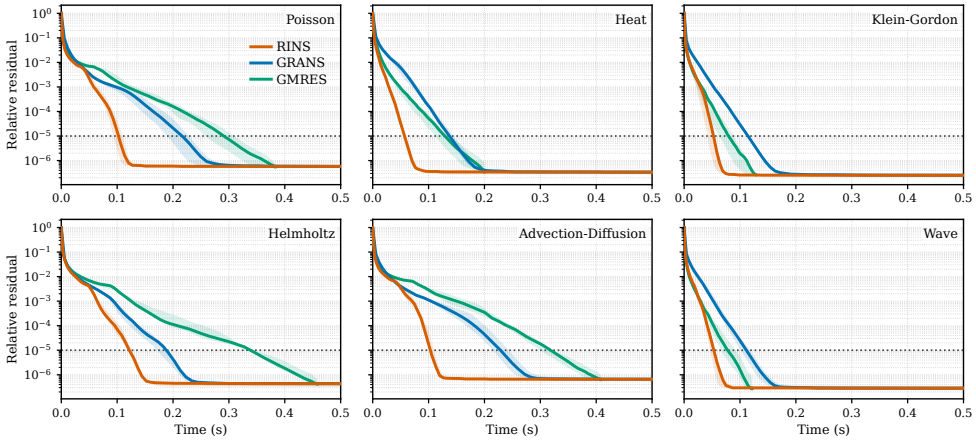



Figure 2: Main benchmark relative-residual curves at $n \approx 1000$. GMRES is run until it reaches $\min(\mathrm{RelRes}_{\mathrm{RINS}}, \mathrm{RelRes}_{\mathrm{GRANS}})$. Curves use one fixed training seed and report median residual over 10 held-out samples, with shaded bands showing the sample interquartile range.

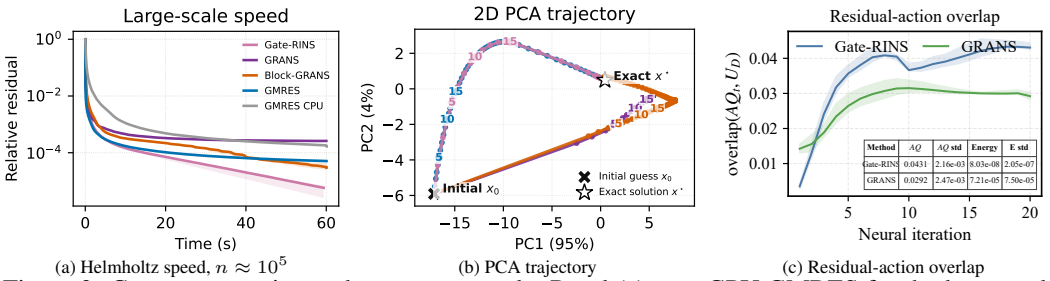
directions at each neural iteration, their subspace dimension grows faster per neural step than the GMRES Krylov dimension grows per Arnoldi step. For RINS this larger basis block does not imply a proportional increase in sparse matrix-vector products: an order-$L$ RINS call constructs the shared polynomial dictionary with $L$ SpMVs and then emits the 20 basis vectors through inexpensive linear combinations. This amortization helps explain why RINS can reach the residual threshold earlier in wall-clock time even though a single neural block iteration contains more work than one Arnoldi step. GRANS also amortizes $L$ message-passing layers over a larger $K \times L$ emitted basis block, but each graph-attention propagation layer is usually more expensive than one SpMV because it computes learned edge weights and feature transforms before aggregation. Under the measured synchronized wall-clock protocol, the neural solvers generally reach the target relative-residual level sooner. Appendix D.2 further shows that the polynomial residual-image generator is not solely benefiting from multi-vector batching. Restricted to generating one basis direction per neural iteration is faster as well.

### 4.2 Large Scale Comparison

Figure 3a focuses on a large-scale Helmholtz instance with $n \approx 10^5$ unknowns. It reports relative residual versus cumulative solve time for Gate-RINS, GRANS, Block-GRANS, and GPU GMRES, using the same synchronized H100 timing convention as the main benchmark. Block-GRANS is the structured GRANS variant described in Appendix C.2. The blue GMRES curve in this large-scale panel is the GPU implementation. This panel tests whether the residual-image polynomial generator remains useful when the sparse operator and accumulated projection basis are substantially larger than in the main $n \approx 1000$ table. The resulting curve summarizes the practical large-scale trade-off: a lower residual trajectory is meaningful only when it survives the full wall-clock cost of basis generation, orthogonalization, and projected least squares. In large-scale tasks, RINS achieves the fastest residual reduction and the lowest final residual within the same computational budget, followed by Block-GRANS, demonstrating favorable scalability.

### 4.3 Trajectory For Different Models

Figure 3b visualizes solver iterates by projecting the trajectories of different methods onto a shared two-dimensional PCA plane. The panel shows how the iterates move from the initial guess toward the exact solution and reveals that different neural solvers exhibit different paths at a different speed.




(a) Helmholtz speed, $n \approx 10^5$ (b) PCA trajectory (c) Residual-action overlap

Figure 3: Compact experimental summary panels. Panel (a) uses GPU GMRES for the large-scale Helmholtz comparison.

Gate-RINS stays close to the GMRES trajectory, consistent with its polynomial residual-probe construction, but each neural step can add a richer block of directions and therefore achieves larger residual reduction after the projected update. By contrast, GRANS and Block-GRANS follow visibly different paths, reflecting the additional freedom introduced by learned graph attention and, for Block-GRANS, its structured feature-block message projection. The colored dotted trajectories use the same method-color assignment as Figure 3a, and the numeric markers indicate the projected locations of the solution vectors at the corresponding neural or Krylov iteration steps. The legend labels only the initial guess and exact solution.

### 4.4 Difficult-Mode Spectral Diagnostics

Figure 3c asks whether the residual-image space $\mathrm{range}(\mathbf{AQ}_t)$ covers spectral components that are hard for the current residual to remove. We run this diagnostic on Poisson systems, where the matrix is symmetric positive definite after discretization. For each test matrix, we form $\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$, sort eigenvectors by increasing eigenvalue magnitude, and define $\mathbf{U}_\mathrm{D} = [\mathbf{v}_{i_1}, \ldots, \mathbf{v}_{i_s}]$ with $s = 8$. At diagnostic step $t$, after the tested solver emits $\mathbf{B}_t$ and updates $\mathbf{Q}_t$, we measure $o_{\mathbf{AQ}}(t) = \|\widehat{\mathbf{AQ}_t}^\top \mathbf{U}_\mathrm{D}\|_\mathrm{F} / \sqrt{\dim(\mathbf{AQ}_t)\dim(\mathbf{U}_\mathrm{D})}$, where $\widehat{\mathbf{AQ}_t} = \mathrm{orth}(\mathbf{AQ}_t)$. Larger values indicate stronger alignment between the residual space used by the projected least-squares update and the difficult eigenspace. Except for the first few iterations, Gate-RINS maintains stronger overlap with these difficult modes, suggesting that its gated polynomial probes become especially useful after the early residual components have already been removed. This alignment-based observation also motivates the later hybrid schedule: an expressive graph controller can provide the initial correction, while Gate-RINS can take over when the solve enters the difficult-mode-dominated stage. The shaded bands are the 25th–75th percentile ranges over the 10 held-out Poisson samples for one fixed training seed, and the inset table reports the final $\mathbf{AQ}$-overlap statistics.

### 4.5 Hybrid Inference Acceleration

From the perspective of discrete optimal control, the hybrid controller enables non-parameter-sharing, or further, non-model-sharing control at different iteration steps. Figure 4 evaluates the hybrid controller described in Section 3: the first neural subspace block is generated by a GRANS prefix trained specifically for the first bootstrap step of this hybrid experiment, and all subsequent blocks are generated by a shared trained RINS-family controller under the same projected least-squares closure. This GRANS prefix is frozen after its first-step training and is not the standalone GRANS checkpoint used elsewhere. We compare two schedules with the same frozen GRANS prefix: Hybrid(GRANS→G-RINS), which uses Gate-RINS after the first step, and Hybrid(GRANS→S-RINS), which uses Static-RINS (no pointwise gate) with the same emitted basis count $m$ and polynomial order $L$. On the Poisson test systems, the mixed schedules test whether the graph controller's initial correction can be composed with either gated or ungated polynomial residual-image generators. The wall-clock panel shows that this improvement is not only an iteration-count effect: because only the first step uses graph message passing, most inference steps keep the RINS cost profile while benefiting from a better initialized subspace trajectory.

### 4.6 Out-of-Distribution Generalization

We evaluate Gate-RINS, GRANS, and Block-GRANS under scale and geometry shifts with a fixed training seed and $\pm 30\%$ parameter variation. Figure 5a tests controllers trained at approximately 1000 nodes on larger meshes (1500–3500 nodes), while panel (b) tests controllers trained at approximately 5000 nodes on smaller meshes (2500–4500 nodes). Both panels report the change in mean

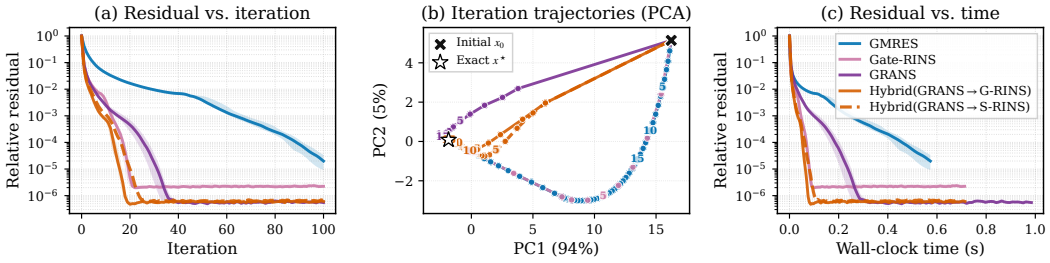
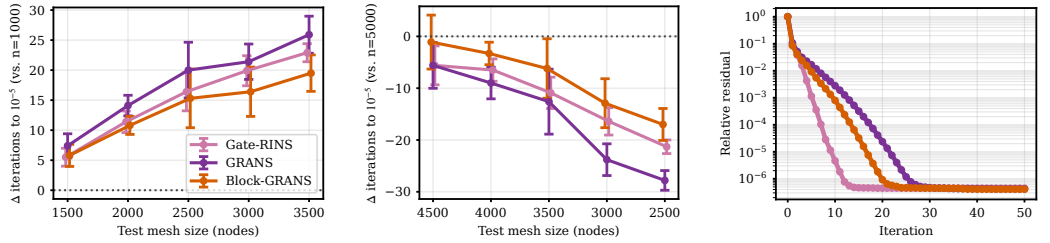



Figure 4: Hybrid inference acceleration on the Poisson problem.

neural iterations to relative residual $10^{-5}$ from the training-scale baseline, with sample-standard-deviation error bars over 10 test systems. Across the tested scale ranges, GRANS shows the largest overall iteration increase as meshes grow and the largest decrease as meshes shrink. By contrast, Block-GRANS shows the smallest overall increase and decrease, consistent with a more restricted transport family, though not establishing a causal role for commutativity (Appendix C). Gate-RINS nevertheless requires fewer iterations than GRANS at every tested size (Appendix B.2). Panel (c) tests transfer from rectangular Poisson domains to an I-shaped domain: all three methods converge, with Gate-RINS showing the fastest residual reduction in this test. Appendix B.3 details the geometry protocol.




(a) Scale-up, trained ≈ 1000 (b) Scale-down, trained ≈ 5000 (c) Geometry, I-shaped domain

Figure 5: Out-of-distribution generalization on Poisson systems under ±30% parameter variation, with a shared method-color legend across the three panels (Gate-RINS, GRANS, Block-GRANS).

### 4.7 Single-Action Approximate-Inverse Quality

We test the approximate inverse viewpoint in Appendix C on 10 Poisson systems at $n \approx 1000$, with $\mathbf{x}_0 = 0$ and $\mathbf{r} = \mathbf{b}$. For each system, we report the relative residual $e_s = \|\mathbf{A}_s\mathbf{z}_s - \mathbf{b}_s\|_2/\|\mathbf{b}_s\|_2$ and summarize its mean and sample standard deviation. Each neural solver uses a one-basis checkpoint, producing one vector $\mathbf{q}$ with correction $\mathbf{z} = \mathbf{q}(\mathbf{A}\mathbf{q})^+\mathbf{b}$. $z = \mathbf{M}\mathbf{b}$ for precondition methods, and $\mathbf{M}$ is the approximate inverse. NeuralIF and neural solvers are trained on 200 systems, GNP separately on each test matrix, and ILU uses fixed drop-tolerance/fill-factor pairs. Table 3 shows that Block-GRANS, GRANS, and Gate-RINS achieve lower mean residuals than NeuralIF, GNP, and the coarser ILU. Among neural methods, Block-GRANS performs best, followed by GRANS and Gate-RINS. Gate-RINS prioritizes lightweight polynomial probing. This also validates the hybrid experiment in Figure 4.

Table 3: Single-action relative residual: mean (sample standard deviation). ILU labels specify drop tolerance and fill factor.

| Method | Mean (Std) |
|---|---|
| NeuralIF | 0.20848 (0.02067) |
| GNP | 0.14405 (0.04038) |
| ILU ($10^{-2}$, 2) | 0.28199 (0.02387) |
| ILU ($10^{-4}$, 10) | 0.00183 (0.00019) |
| Gate-RINS | 0.12817 (0.01473) |
| GRANS | 0.10485 (0.03650) |
| Block-GRANS | 0.08147 (0.01419) |

## 5 Conclusion and Outlook

We presented RINS, a residual-image neural subspace solver that preserves projected minimum-residual updates while learning only the basis generator. Gate-RINS combines residual-image polynomial probes with a lightweight pointwise gate, improving subspace flexibility without increasing the dominant SpMV cost. Experiments show that faster convergence correlates with better alignment between $\mathrm{range}(\mathbf{A}\mathbf{Q}_t)$ and difficult residual components. Despite higher per-block cost than a GMRES Arnoldi step, RINS achieves larger residual reduction per iteration and lower end-to-end runtime on the evaluated PDE systems. Future work will explore new neural-solver variants and matrix-free deployment for large-scale scientific computing.

### AI USE STATEMENT

In this work, we used generative AI tools for drafting assistance and editing support during manuscript preparation. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

This work studies numerical solvers for sparse linear systems arising from PDE discretizations. We do not use human-subject data or personally identifiable information. Potential risks are indirect and relate to downstream scientific-computing applications using the proposed solver.

### REPRODUCIBILITY STATEMENT

We describe the model architecture, training schedule, benchmark construction, evaluation protocol, timing convention, and ablation settings in the main text and appendix. The experiments use fixed train/test splits and report median trajectories with uncertainty bands over held-out systems.

## REFERENCES

Walter Edwin Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9(1):17–29, 1951. URL https://www.ams.org/qam/1951-09-01/.

Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019. doi: 10.1073/pnas.1814058116.

Michele Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, 182(2):418–477, 2002. URL https://www.sciencedirect.com/science/article/pii/S0021999102971767.

Kaushik Bhattacharya, Bamdad Hosseini, Nikola B Kovachki, and Andrew M Stuart. Model reduction and neural networks for parametric PDEs. *The SMAI Journal of Computational Mathematics*, 7:121–157, 2021. doi: 10.5802/smai-jcm.74.

Andrey Bryutkin, Jiahao Huang, Zhongying Deng, Guang Yang, Carola-Bibiane Schönlieb, and Angelica I Aviles-Rivero. HAMLET: Graph transformer neural operator for partial differential equations. In *International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4624–4641. PMLR, 2024. URL https://proceedings.mlr.press/v235/bryutkin24a.html.

Wenbo Cao and Weiwei Zhang. TSONN: Time-stepping-oriented neural network for solving partial differential equations. *arXiv preprint arXiv:2310.16491*, 2023. URL https://arxiv.org/abs/2310.16491.

Jie Chen. Graph neural preconditioners for iterative solutions of sparse linear systems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2406.00809.

Pao-Hsiung Chiu, Jian Cheng Wong, Chin Chun Ooi, Chang Wei, Yuchen Fan, and Yew-Soon Ong. Scale-PINN: Learning efficient physics-informed neural networks through sequential correction. *arXiv preprint arXiv:2602.19475*, 2026. URL https://arxiv.org/abs/2602.19475.

Timothy A. Davis. *Direct Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2006. doi: 10.1137/1.9780898718881.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, volume 29, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/04df4d434d481c5bb723be1b6df1ee65-Abstract.html.

Weinan E and Bing Yu. The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018. doi: 10.1007/s40304-018-0127-z.

Howard C Elman, David J Silvester, and Andrew J Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Oxford University Press, 2 edition, 2014. doi: 10.1093/acprof:oso/9780199678792.001.0001.

Roland W Freund and Noël M Nachtigal. QMR: A quasi-minimal residual method for non-Hermitian linear systems. *Numerische Mathematik*, 60(1):315–339, 1991. doi: 10.1007/BF01385726.

Gene H Golub and Charles F Van Loan. *Matrix computations*. Johns Hopkins University Press, 4 edition, 2013. URL https://www.press.jhu.edu/books/title/10678/matrix-computations.

Anne Greenbaum. *Iterative methods for solving linear systems*. SIAM, 1997. doi: 10.1137/1.9781611970937.

Daniel Greenfeld, Meirav Galun, Ron Kimmel, Irad Yavneh, and Ronen Basri. Learning to optimize multigrid PDE solvers. In *International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2415–2423. PMLR, 2019. URL https://proceedings.mlr.press/v97/greenfeld19a/greenfeld19a.pdf.

Tamara G Grossmann, Urszula Julia Komorowska, Jonas Latz, and Carola-Bibiane Schönlieb. Can physics-informed neural networks beat the finite element method? *IMA Journal of Applied Mathematics*, 89(1):143–174, 2024. doi: 10.1093/imamat/hxae011.

Eldad Haber. *Computational Methods in Geophysical Electromagnetics*. Society for Industrial and Applied Mathematics, 2014. doi: 10.1137/1.9781611973808.

Wolfgang Hackbusch. *Multi-Grid Methods and Applications*, volume 4 of *Springer Series in Computational Mathematics*. Springer-Verlag, 1985. doi: 10.1007/978-3-662-02427-0.

Paul Häusner, Ozan Öktem, and Jens Sjölund. Neural incomplete factorization: learning preconditioners for the conjugate gradient method. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=FozLrZ3CI5.

Magnus R Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952. doi: 10.6028/jres.049.044.

Ayano Kaneda, Osman Akar, Jingyu Chen, Victoria Alicia Trevino Kala, David Hyde, and Joseph Teran. A deep conjugate direction method for iteratively solving linear systems. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 15720–15736, 2023. URL https://proceedings.mlr.press/v202/kaneda23a.html.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021. doi: 10.1038/s42254-021-00314-5.

Ehsan Kharazmi, Zhongqiang Zhang, and George Em Karniadakis. hp-VPINNs: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547, 2021. doi: 10.1016/j.cma.2020.113547.

D. A. Knoll and D. E. Keyes. Jacobian-free Newton–Krylov methods: A survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004. doi: 10.1016/j.jcp.2003.08.010.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023. URL https://www.jmlr.org/papers/v24/21-1524.html.

Aditi S Krishnapriyan, Amir Gholami, Shandian Zhe, Robert M Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 26548–26560, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/df438e5206f31600e6ae4af72f2725f1-Abstract.html.

Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4): 255–282, 1950. doi: 10.6028/jres.045.026.

Zhihao Li, Di Xiao, Zhilu Lai, and Wei Wang. Neural preconditioning operator for efficient PDE solves. *arXiv preprint arXiv:2502.01337*, 2025. URL https://arxiv.org/abs/2502.01337.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020a. URL https://arxiv.org/abs/2003.03485.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6755–6766, 2020b. URL https://proceedings.neurips.cc/paper/2020/hash/4b21cf96d4cf612f239a6c322b10c8fe-Abstract.html.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Weixin Liao, Mingquan Feng, Zhizhou Zhang, Youjia Wu, Yifan Fu, and Junchi Yan. G-RANS: Generalizable residual-aware neural solvers for sparse systems. In *International Conference on Machine Learning (ICML)*, 2026. URL https://icml.cc/virtual/2026/poster/60991.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021. doi: 10.1038/s42256-021-00302-5.

Jian Luo, Jie Wang, Hong Wang, Huanshuo Dong, Zijie Geng, Hanzhu Chen, and Yufei Kuang. Neural Krylov iteration for accelerating linear system solving. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/e88870ec82f2469b0ddf32c817920c68-Abstract-Conference.html.

Ronald B Morgan. GMRES with deflated restarting. *SIAM Journal on Scientific Computing*, 24(1): 20–37, 2002. doi: 10.1137/S1064827599364659.

Michael L Parks, Eric de Sturler, Greg Mackey, Duane D Johnson, and Spandan Maiti. Recycling Krylov subspaces for sequences of linear systems. *SIAM Journal on Scientific Computing*, 28(5): 1651–1674, 2006. doi: 10.1137/040607277.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=roNqYL0_XP.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

Youcef Saad and Martin H Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986. doi: 10.1137/0907058.

Yousef Saad. ILUT: A dual threshold incomplete LU factorization. *Numerical Linear Algebra with Applications*, 1(4):387–402, 1994. doi: 10.1002/nla.1680010405.

Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2 edition, 2003. doi: 10.1137/1.9780898718003.

Valeria Simoncini and Daniel B Szyld. Recent computational developments in Krylov subspace methods for linear systems. *Numerical Linear Algebra with Applications*, 14(1):1–59, 2007. doi: 10.1002/nla.499.

Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018. doi: 10.1016/j.jcp.2018.08.029.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: An extensive benchmark for scientific machine learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 1596–1611, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/0a9747136d411fb83f0cf81820d44afb-Abstract-Datasets_and_Benchmarks.html.

Lloyd N Trefethen. *Approximation theory and approximation practice, extended edition*. SIAM, 2019. doi: 10.1137/1.9781611975949.

Henk A van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631–644, 1992. doi: 10.1137/0913035.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, Anima Anandkumar, Karianne Bergen, Carla P. Gomes, Shirley Ho, Pushmeet Kohli, Joan Lasenby, Jure Leskovec, Tie-Yan Liu, Arjun Manrai, Debora Marks, Bharath Ramsundar, Le Song, Jimeng Sun, Jian Tang, Petar Veličković, Max Welling, Linfeng Zhang, Connor W. Coley, Yoshua Bengio, and Marinka Zitnik. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023. doi: 10.1038/s41586-023-06221-2.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021. doi: 10.1137/20M1318043.

Jinchao Xu. Iterative methods by space decomposition and subspace correction. *SIAM Review*, 34 (4):581–613, 1992. doi: 10.1137/1034116.

Enrui Zhang, Adar Kahana, Alena Kopaničáková, Eli Turkel, Rishikesh Ranade, Jay Pathak, and George Em Karniadakis. Blending neural operators and relaxation methods in PDE numerical solvers. *Nature Machine Intelligence*, 6(11):1303–1313, 2024. doi: 10.1038/s42256-024-00910-x.

## APPENDIX CONTENTS

A Uniform Emulation of Fixed-Step GMRES . . . . . 14
B Experimental Details . . . . . 15
B.1 Benchmark Configuration . . . . . 15
B.2 Grid-Size Generation . . . . . 17
B.3 Geometry Generalization . . . . . 18
C Dynamic Nonlinear Approximate-Inverse Framework . . . . . 19
C.1 Approximate-Inverse Interpretation and Transport Structure . . . . . 19
C.2 From GRANS to Block-GRANS . . . . . 20
D Supplementary Experimental Results . . . . . 21
D.1 Preconditioning Integration Experiments . . . . . 21
D.2 One-Basis Limit Test . . . . . 21

## A UNIFORM EMULATION OF FIXED-STEP GMRES

This appendix records the theoretical statement: an expressive continuous neural generator can uniformly emulate the correction produced by a fixed number of GMRES steps on a compact, nondegenerate problem family. Namely, a neural basis generator with at least one emitted direction can approximate the residual quality of fixed-step GMRES uniformly over the family.

**Theorem 1 (uniform emulation of $m$-step GMRES).** Let $\mathbb{D}$ be compact and let $\mathbf{A}(\boldsymbol{\mu}) \in \mathbb{R}^{n\times n}$ and $\mathbf{r}_0(\boldsymbol{\mu}) \in \mathbb{R}^n$ be continuous in $\boldsymbol{\mu} \in \mathbb{D}$. Assume that $\mathbf{A}(\boldsymbol{\mu})$ is uniformly nonsingular and that, for a fixed $m$, the matrix $\mathbf{Y}_m(\boldsymbol{\mu}) = \mathbf{A}(\boldsymbol{\mu})\mathbf{K}_m(\boldsymbol{\mu}), \mathbf{K}_m(\boldsymbol{\mu}) = [\mathbf{r}_0(\boldsymbol{\mu}), \mathbf{A}(\boldsymbol{\mu})\mathbf{r}_0(\boldsymbol{\mu}), \ldots, \mathbf{A}(\boldsymbol{\mu})^{m-1}\mathbf{r}_0(\boldsymbol{\mu})]$ has constant rank on $\mathbb{D}$. Let $\rho_m^{\mathrm{GMRES}}(\boldsymbol{\mu}) = \min_{\mathbf{y}} \|\mathbf{r}_0(\boldsymbol{\mu}) - \mathbf{Y}_m(\boldsymbol{\mu})\mathbf{y}\|_2 = \|\mathbf{r}_0(\boldsymbol{\mu}) - \mathbf{Y}_m(\boldsymbol{\mu})\mathbf{y}_m^*\|_2$, where $\mathbf{y}_m^* = \mathbf{Y}_m(\boldsymbol{\mu})^+\mathbf{r}_0$. Then, for every $\varepsilon > 0$, there exists a continuous neural generator $\Phi_{\boldsymbol{\theta}}$ that maps the system data $(\mathrm{vec}\,\mathbf{A}(\boldsymbol{\mu}), \mathbf{r}_0(\boldsymbol{\mu}))$ to a correction vector $\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{\mu}) \in \mathbb{R}^n$ such that the one-dimensional projected update over $\mathrm{span}\{\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{\mu})\}$ satisfies $\min_{\alpha\in\mathbb{R}} \|\mathbf{r}_0(\boldsymbol{\mu}) - \alpha\mathbf{A}(\boldsymbol{\mu})\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{\mu})\|_2 \le \rho_m^{\mathrm{GMRES}}(\boldsymbol{\mu}) + \varepsilon, \forall \boldsymbol{\mu} \in \mathbb{D}$.

**Proof.** Define the compact data set

$$\mathbb{S} := \{(\mathrm{vec}\,\mathbf{A}(\boldsymbol{\mu}), \mathbf{r}_0(\boldsymbol{\mu})) : \boldsymbol{\mu} \in \mathbb{D}\} \subset \mathbb{R}^{n^2+n}.$$

The set $\mathbb{S}$ is compact because it is the continuous image of the compact set $\mathbb{D}$.

First, we show that the GMRES correction is a continuous function of the system data on $\mathbb{S}$. For any system data $\mathbf{s} = (\mathrm{vec}\,\mathbf{A}, \mathbf{r}) \in \mathbb{S}$, define

$$\mathbf{K}_m(\mathbf{A}, \mathbf{r}) := [\mathbf{r}, \mathbf{A}\mathbf{r}, \ldots, \mathbf{A}^{m-1}\mathbf{r}], \qquad \mathbf{Y}_m(\mathbf{A}, \mathbf{r}) := \mathbf{A}\mathbf{K}_m(\mathbf{A}, \mathbf{r}).$$

The entries of $\mathbf{K}_m(\mathbf{A}, \mathbf{r})$ and $\mathbf{Y}_m(\mathbf{A}, \mathbf{r})$ are polynomial functions of the entries of $\mathbf{A}$ and $\mathbf{r}$. The rank of $\mathbf{Y}_m(\mathbf{A}, \mathbf{r})$ is constant for $\mathbf{s} \in \mathbb{S}$, so the Moore–Penrose pseudoinverse $\mathbf{Y}_m(\mathbf{A}, \mathbf{r})^+$ is continuous on $\mathbb{S}$. Therefore the map

$$\mathcal{F} : \mathbb{S} \to \mathbb{R}^n, \qquad \mathcal{F}(\mathrm{vec}\,\mathbf{A}, \mathbf{r}) := \mathbf{K}_m(\mathbf{A}, \mathbf{r})\mathbf{Y}_m(\mathbf{A}, \mathbf{r})^+\mathbf{r}$$

is continuous. For $\mathbf{s}(\boldsymbol{\mu}) = (\mathrm{vec}\,\mathbf{A}(\boldsymbol{\mu}), \mathbf{r}_0(\boldsymbol{\mu}))$, this definition gives

$$\mathcal{F}(\mathbf{s}(\boldsymbol{\mu})) = \mathbf{K}_m(\boldsymbol{\mu})\mathbf{Y}_m(\boldsymbol{\mu})^+\mathbf{r}_0(\boldsymbol{\mu}) := \mathbf{z}_m^\star(\boldsymbol{\mu}).$$

Second, we approximate this continuous correction map uniformly. By the universal approximation theorem on the compact set $\mathbb{S}$, for every $\delta > 0$ there exists a continuous neural network $\Phi_{\boldsymbol{\theta}} : \mathbb{R}^{n^2+n} \to \mathbb{R}^n$ such that

$$\sup_{\mathbf{s}\in\mathbb{S}} \|\Phi_{\boldsymbol{\theta}}(\mathbf{s}) - \mathcal{F}(\mathbf{s})\|_2 < \delta.$$

For $\mathbf{s}(\boldsymbol{\mu}) = (\operatorname{vec} \mathbf{A}(\boldsymbol{\mu}), \mathbf{r}_0(\boldsymbol{\mu}))$, write

$$\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{\mu}) := \Phi_{\boldsymbol{\theta}}(\mathbf{s}(\boldsymbol{\mu})).$$

Then

$$\sup_{\boldsymbol{\mu} \in \mathbb{D}} \|\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{\mu}) - \mathbf{z}_m^\star(\boldsymbol{\mu})\|_2 < \delta.$$

Third, we translate correction-vector approximation into residual approximation. Since $\mathbf{A}$ is continuous and $\mathbb{D}$ is compact,

$$b_{\max} := \sup_{\boldsymbol{\mu} \in \mathbb{D}} \|\mathbf{A}(\boldsymbol{\mu})\|_2 < \infty.$$

Uniform nonsingularity implies $\mathbf{A}(\boldsymbol{\mu}) \neq 0$ for all $\boldsymbol{\mu}$, so $b_{\max} > 0$. For each fixed $\boldsymbol{\mu}$, the projected residual over $\operatorname{span}\{\boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{\mu})\}$ is no larger than the residual obtained from the admissible scalar choice $\alpha = 1$. Therefore

$$\begin{aligned}
\min_{\alpha \in \mathbb{R}} \|\mathbf{r}_0(\boldsymbol{\mu}) - \alpha \mathbf{A}(\boldsymbol{\mu}) \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{\mu})\|_2 &\leq \|\mathbf{r}_0(\boldsymbol{\mu}) - \mathbf{A}(\boldsymbol{\mu}) \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{\mu})\|_2 \\
&\leq \|\mathbf{r}_0(\boldsymbol{\mu}) - \mathbf{A}(\boldsymbol{\mu}) \mathbf{z}_m^\star(\boldsymbol{\mu})\|_2 + \|\mathbf{A}(\boldsymbol{\mu})(\mathbf{z}_m^\star(\boldsymbol{\mu}) - \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{\mu}))\|_2 \\
&\leq \rho_m^{\mathrm{GMRES}}(\boldsymbol{\mu}) + \|\mathbf{A}(\boldsymbol{\mu})\|_2 \, \|\mathbf{z}_m^\star(\boldsymbol{\mu}) - \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{\mu})\|_2 \\
&< \rho_m^{\mathrm{GMRES}}(\boldsymbol{\mu}) + b_{\max} \delta.
\end{aligned}$$

Choose $\delta = \varepsilon / b_{\max}$. Then the last display gives

$$\min_{\alpha \in \mathbb{R}} \|\mathbf{r}_0(\boldsymbol{\mu}) - \alpha \mathbf{A}(\boldsymbol{\mu}) \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{\mu})\|_2 < \rho_m^{\mathrm{GMRES}}(\boldsymbol{\mu}) + \varepsilon$$

for every $\boldsymbol{\mu} \in \mathbb{D}$. Replacing the strict inequality by the weaker non-strict inequality gives the stated claim.

The intuition is that fixed-step GMRES defines a continuous correction rule on any compact region where the Krylov action matrix $\mathbf{Y}_m$ does not change rank. A sufficiently expressive continuous neural generator can uniformly approximate that rule, and the projected one-dimensional update can always choose the scalar $\alpha = 1$, so its residual cannot be worse than the residual of the approximated GMRES correction plus the operator-norm approximation error. □

# B Experimental Details

## B.1 Benchmark Configuration

All benchmark matrices are generated by a finite-element data-generation pipeline. Each sample is a sparse linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$. The benchmark labels refer to the PDE model used to generate the assembled linear system. All systems are assembled with continuous piecewise-linear triangular finite elements on a quadrilateral domain. The nominal domain has bottom corners $(-1, -1)$ and $(1, -1)$ and top corners $(-1, 1)$ and $(1, 1)$; for each randomized sample, the two top corners are perturbed independently by a uniform displacement in $[-0.3, 0.3]^2$. The mesh also includes a fixed circular interior facet of radius $0.25$ centered at the origin, while the coefficient inclusion used by some PDE branches may be perturbed separately. Nonzero bottom Dirichlet data are handled by a lifting vector $\mathbf{u}_0$: the stored linear system solved by all iterative methods is the homogeneous-boundary correction system for $\hat{\mathbf{u}}$, obtained after subtracting $\mathbf{A}\mathbf{u}_0$ from the assembled right-hand side and enforcing identity rows on boundary degrees of freedom.

For all six tasks, randomized scalar parameters are sampled from their baseline values with relative variation $\rho = 0.3$: each parameter $\theta_0$ is replaced by $\theta = \theta_0(1 + \xi)$, with $\xi \sim \mathrm{Unif}[-\rho, \rho]$, unless a smaller geometric perturbation range is stated below. The common diffusion and source pattern is based on a circular inclusion:

$$\kappa(\mathbf{x}) = \begin{cases} \kappa_{\mathrm{in}}, & \|\mathbf{x} - \mathbf{c}\| \leq r, \\ \kappa_{\mathrm{out}}, & \|\mathbf{x} - \mathbf{c}\| > r, \end{cases} \qquad f(\mathbf{x}) = \begin{cases} s_f, & \|\mathbf{x} - \mathbf{c}\| \leq r, \\ 0, & \|\mathbf{x} - \mathbf{c}\| > r. \end{cases}$$

Here $g(x, y) = c_g(1 - x^2)$ on the prescribed bottom boundary. For all tasks using this common pattern, $\kappa_{\mathrm{in}} \in [0.77, 1.43]$, $\kappa_{\mathrm{out}} \in [0.70, 1.30]$, $s_f \in [0.07, 0.13]$, and $c_g \in [0.70, 1.30]$.

Table 4: PDE-derived linear systems used in the main benchmark. Klein–Gordon and wave are implicit one-step systems from second-order time-evolution equations.

| Task | Continuous/discrete form |
|---|---|
| Poisson | $-\nabla\cdot(\kappa\nabla u)=f.$ |
| Helmholtz | $-\nabla\cdot(\kappa\nabla u)+\alpha(\mathbf{x})u=f.$ |
| Heat | Backward Euler step $(\mathbf{M}+\Delta t\mathbf{K})\mathbf{u}^{n+1}=\mathbf{M}\mathbf{u}^{n}+\Delta t\,\mathbf{b}.$ |
| Advection–diffusion | $-\nabla\cdot(\kappa\nabla u)+\mathbf{v}\cdot\nabla u=f.$ |
| Klein–Gordon | $(\Delta t^{-2}\mathbf{M}+c^2\mathbf{K}+\mathbf{M}_m)\mathbf{u}^{n+1}=\mathbf{f}^{n+1}+\Delta t^{-2}\mathbf{M}(2\mathbf{u}^{n}-\mathbf{u}^{n-1}).$ |
| Wave | $(\Delta t^{-2}\mathbf{M}+c^2\mathbf{K})\mathbf{u}^{n+1}=\mathbf{f}^{n+1}+\Delta t^{-2}\mathbf{M}(2\mathbf{u}^{n}-\mathbf{u}^{n-1}).$ |

Table 5: Actual mesh-size distributions for the main benchmark at $n\approx 1000$. Entries are mean±standard deviation with the observed min–max range in parentheses.

| Task | Train | Test |
|---|---|---|
| Poisson | $929.35\pm67.81$ (785–1117) | $913.70\pm93.47$ (786–1091) |
| Helmholtz | $918.72\pm66.86$ (741–1113) | $926.10\pm70.60$ (781–1035) |
| Heat | $909.88\pm60.63$ (764–1051) | $922.80\pm55.06$ (829–1011) |
| Advection-Diffusion | $913.74\pm55.55$ (761–1042) | $918.00\pm67.28$ (818–1026) |
| Klein-Gordon | $919.78\pm64.89$ (766–1092) | $935.50\pm50.59$ (892–1076) |
| Wave | $919.57\pm70.94$ (741–1133) | $941.90\pm60.14$ (842–1020) |

**Poisson.** The Poisson task uses the common diffusion/source inclusion and boundary lifting described above. Its only task-specific geometric perturbations are the coefficient-inclusion radius $r=0.25(1+\xi_r)$, where $\xi_r\sim\mathrm{Unif}[-0.15,0.15]$, and the two coordinates of the inclusion center, each sampled in $[-0.03,0.03]$. The assembled correction system is the P1 finite-element stiffness system $\mathbf{K}\hat{\mathbf{u}}=\tilde{\mathbf{b}}$, where local stiffness entries are obtained from $\int_{\mathbb{T}}\kappa\nabla\phi_i\cdot\nabla\phi_j$ and the load uses the centroid value of the source on each triangle.

**Helmholtz.** The Helmholtz task uses the same diffusion/source inclusion pattern as Poisson but adds a positive reaction term. The reaction parameters are sampled as $\alpha_{\mathrm{in}}\in[1.40,2.60]$ and $\alpha_{\mathrm{out}}\in[0.70,1.30]$. The coefficient-inclusion radius again uses $r=0.25(1+\xi_r)$, $\xi_r\sim\mathrm{Unif}[-0.15,0.15]$, while each coordinate of the inclusion center is sampled in $[-0.06,0.06]$. The discrete correction matrix is $\mathbf{K}+\mathbf{M}_\alpha$, where $\mathbf{M}_\alpha$ is the consistent P1 weighted mass matrix assembled with centroid-valued $\alpha(\mathbf{x})$. This benchmark task is therefore a positive-reaction Helmholtz-form sparse linear system, not the indefinite high-frequency Helmholtz regime.

**Heat.** The heat task represents one backward-Euler implicit step, not a long time-dependent simulation. It uses the common diffusion/source inclusion with the nominal center $\mathbf{c}=(0,0)$ and radius $0.25$. The time step satisfies $\Delta t\in[0.07,0.13]$. The previous state is parameterized as

$$u^n(x,y)=a_1\sin(\pi x)\sin(\pi y)+a_2\cos(\pi x)\cos(\pi y),$$

with $a_1\in[0.70,1.30]$ and $a_2\in[0.35,0.65]$, and is made consistent with the prescribed bottom boundary before forming the step. The resulting correction system uses $\mathbf{A}=\mathbf{M}+\Delta t\mathbf{K}$ and right-hand side $\mathbf{M}\mathbf{u}^n+\Delta t\mathbf{b}$, followed by the same lifting and boundary-row enforcement as the other finite-element tasks.

**Advection–Diffusion.** The advection–diffusion task uses the common diffusion/source inclusion with nominal center $\mathbf{c}=(0,0)$ and radius $0.25$, and adds a first-order transport field $\mathbf{v}(x,y)=\beta(-y,x)$, where $\beta\in[0.35,0.65]$. The discrete correction matrix is $\mathbf{K}+\mathbf{C}$. The diffusion block is the same stiffness matrix used in the Poisson branch, while the nonsymmetric transport block is assembled from the centroid quadrature approximation to $\int_{\mathbb{T}}\phi_i\,\mathbf{v}\cdot\nabla\phi_j$. This is the only main benchmark branch whose nonsymmetry comes from a first-order differential operator.

**Klein–Gordon.** The Klein–Gordon task represents one implicit step of the second-order equation $u_{tt}-c^2\nabla\cdot(\kappa\nabla u)+m^2(\mathbf{x})u=f$. It uses the common diffusion/source inclusion with nominal center $\mathbf{c}=(0,0)$ and radius $0.25$, wave speed $c\in[0.70,1.30]$, time step $\Delta t\in[0.35,0.65]$, and mass coefficients $m^2_{\mathrm{in}}\in[2.10,3.90]$, $m^2_{\mathrm{out}}\in[1.05,1.95]$. Two smooth previous states $u^n$ and $u^{n-1}$ are sampled from sinusoidal modes. The assembled correction matrix is $\Delta t^{-2}\mathbf{M}+c^2\mathbf{K}+\mathbf{M}_m$, and the right-hand side contains $\mathbf{f}^{n+1}+\Delta t^{-2}\mathbf{M}(2\mathbf{u}^n-\mathbf{u}^{n-1})$.

**Wave.** The wave task represents one implicit step of $u_{tt} - c^2\nabla\cdot(\kappa\nabla u) = f$. It uses the common diffusion/source inclusion, wave speed $c \in [0.70, 1.30]$, time step $\Delta t \in [0.35, 0.65]$, and two smooth previous states sampled from the same sinusoidal family used in the Klein–Gordon task. The assembled correction matrix is $\Delta t^{-2}\mathbf{M} + c^2\mathbf{K}$, and the right-hand side is $\mathbf{f}^{n+1} + \Delta t^{-2}\mathbf{M}(2\mathbf{u}^n - \mathbf{u}^{n-1})$.

For each PDE and each scale, the training set contains 200 samples and the held-out inference set contains 10 samples. The nominal mesh scales are $n \approx 1000$, but the actual number of nodes varies because both the domain geometry and the triangular refinement are randomized. Tables 5 report the realized distributions recorded by the data-generation logs.

**Model and training parameters.** The training configuration in the benchmark script uses the same schedule for RINS and GRANS: 100 stage-1 epochs, 100 stage-2 epochs, and 150 stage-3 epochs, with learning rate $10^{-3}$, and weight decay $10^{-5}$. The main RINS model uses 20 generated filters per neural iteration, polynomial order $L = 4$, Chebyshev initialization, gate hidden width 16, and gate scale 0.5. Static-RINS uses the same polynomial generator without the pointwise gate. GRANS uses coordinate dimension 2, positional encoding dimension 16, 48 hidden channels, 4 graph layers, 5 basis directions per layer, 3 attention heads, and dropout 0.08. During inference, both neural solvers are run for 200 neural block iterations in the main benchmark.

**Inference and statistic setting.** For timing, the benchmark fixes CPU linear-algebra threading with `OMP_NUM_THREADS=1` and `MKL_NUM_THREADS=1`; GPU timings are recorded with explicit CUDA synchronization around the measured region and wall-clock intervals from `time.perf_counter()`. These synchronization barriers prevent unfinished CUDA kernels from being hidden behind asynchronous host execution. An optimized deployment that pipelines these operations asynchronously would typically reduce overhead or improve throughput, so the reported neural-solver times should be viewed as conservative synchronized measurements. The paper figures report median curves over the 10 held-out test samples for a fixed training seed, with IQR bands over those samples. The timing tables pool all valid seed/sample time-to-threshold observations: for each available seed and each held-out sample, we record one value and compute the reported mean and standard deviation over the pooled observations. Thus, the reported standard deviation includes variation across seeds and variation across samples under each seed. Timed GPU sections call `torch.cuda.synchronize()` before and after the measured solver step.

### B.2 Grid-Size Generation

Figure 6 complements the baseline-relative curves in Section 4.6 with absolute neural iteration counts to relative residual $10^{-5}$. The two experiments use separate controllers trained at approximately 1000 and 5000 nodes, respectively, with a fixed training seed. Each plotted mean and sample standard deviation is computed over 10 test systems at the indicated target mesh size.

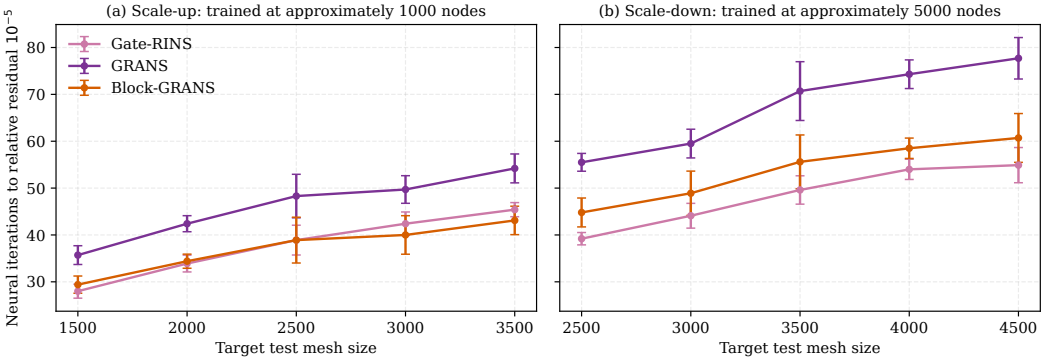



Figure 6: Absolute iteration counts for Poisson grid-size generalization: (a) training at approximately 1000 nodes and testing on larger meshes; (b) training at approximately 5000 nodes and testing on smaller meshes. Points and error bars denote means and sample standard deviations over 10 test systems.

**Transfer to larger systems.** The training-scale mean counts are 22.5, 28.3, and 23.6 for Gate-RINS, GRANS, and Block-GRANS, respectively. At target size 3500, these become $45.4 \pm 1.5$,

$54.2 \pm 3.1$, and $43.1 \pm 3.0$. Gate-RINS consistently uses fewer iterations than GRANS, but its ordering relative to Block-GRANS is not invariant: the two tie at 2500 nodes (38.9 iterations), and Block-GRANS has lower mean counts at 3000 and 3500 nodes.

**Transfer to smaller systems.** Training-scale means are 60.5, 83.3, and 61.8 in the same model order. At target size 2500, the corresponding counts decrease to $39.2 \pm 1.3$, $55.5 \pm 1.9$, and $44.8 \pm 3.1$. Gate-RINS has the lowest mean count at every tested size in this experiment.

### B.3 Geometry Generalization

This appendix records the detailed protocol behind Panel (c) of Figure 5 (and its summary in Section 4.6). The experiment isolates whether the learned controllers keep using spatial information, rather than memorizing a particular domain, by holding the PDE family fixed and changing the mesh geometry.

**Domain geometry.** Training systems are assembled on a rectangular domain with corners (-1,-1), (1,-1), (1,1), (-1,1). Test systems are assembled on an I-shaped domain whose boundary is the polygon (-1,-1), (1,-1), (1,-0.3), (0.2,-0.3), (0.2,0.3), (1,0.3), (1,1), (-1,1), (-1,0.3), (-0.2,0.3), (-0.2,-0.3), (-1,-0.3). Both domains are triangulated with meshpy.triangle to roughly $n \approx 1000$ nodes using an area-based refinement criterion proportional to $250/n_{\text{target}}$, so the two geometries yield comparable discretization resolution.

**Equation and randomized parameters.** All samples solve the heterogeneous Poisson equation

$$-\nabla \cdot \big(\kappa(\mathbf{x}) \nabla u\big) = f(\mathbf{x}) \quad \text{in } \Omega, \qquad u(\mathbf{x}) = g(\mathbf{x}) \quad \text{on } y = -1,$$

with the same circular inclusion of radius $0.25$ centered at the origin. Inside the inclusion $\kappa = \kappa_{\text{inner}}$ and the source $f = s$; outside, $\kappa = \kappa_{\text{outer}}$ and $f = 0$, with bottom Dirichlet data $g = \beta(1 - x^2)$. For each training and test sample, the two diffusion coefficients, the source strength, and the boundary scale are each drawn as $\theta = \theta_{\text{base}} \cdot \big(1 + \delta\, \mathcal{U}(-1, 1)\big), \delta = 0.3$ with baselines $\kappa_{\text{inner}} = 1.1$, $\kappa_{\text{outer}} = 1.0$, $s = 0.1$, and $\beta = 1.0$. Thus both the rectangular training set and the I-shaped test set carry the same $\pm 30\%$ parameter variation. Only the mesh geometry is shifted between them.

**Discretization and linear system.** Continuous piecewise-linear triangular finite elements assemble the stiffness matrix $\mathbf{A}$ and the body-force vector $b$. Nonzero bottom Dirichlet data are lifted by a solution $u_0$ supported on the boundary nodes (those with $y = -1$), and the iterative solvers act on the homogeneous correction system for $\widehat{u} = u - u_0$, after forcing identity rows on all boundary degrees of freedom. Each stored sample records the COO form of $\mathbf{A}$, the right-hand side $\mathbf{b}$, the node coordinates, and the direct solve $u$. The training set contains 200 samples on the rectangle, and the held-out inference set contains 20 samples on the I-shape; a single training seed 2026 is used throughout.

**Training.** Gate-RINS, GRANS, and Block-GRANS are trained on the 200 rectangular training systems with the same three-stage schedule as the main benchmark: 100 stage-1 epochs, 100 stage-2 epochs, and 150 stage-3 epochs, with AdamW, learning rate $10^{-3}$, weight decay $10^{-5}$, and gradient clipping at norm 1.0. Gate-RINS uses 20 filters, Chebyshev order $4$, coordinate dimension 2, gate hidden width 16, and gate scale 0.5. GRANS uses coordinate dimension 2, positional encoding dimension 16, 48 hidden channels, 4 graph layers, 5 basis directions per layer, 3 attention heads, and dropout 0.08; Block-GRANS adds a structured feature-block message projection with 4 projection blocks, matching the configuration in Section C.2.

**Inference and figure.** Each model is evaluated on the same 20 I-shaped test systems, generating 20 basis directions per neural iteration for a budget of 50 iterations with a maximum accumulated basis size of 1000. Figure 5c reports the per-iteration relative residual averaged over the 20 samples, with the shaded band spanning the 25th–75th percentile; panels (a) and (b) of the same figure use the shared method-color legend described in the caption. Per-model residual histories are cached as `npz` files and reused unless inference is explicitly re-run.

# C Dynamic Nonlinear Approximate-Inverse Framework

## C.1 Approximate-Inverse Interpretation and Transport Structure

**What is approximated.** Consider a nonsingular linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$, and let $\mathbf{r}_t$ denote the residual at iteration $t$. The ideal correction to the current iterate is $\mathbf{A}^{-1}\mathbf{r}_t$. Since $\mathbf{A}^{-1}$ is precisely the object the iteration aims to approximate—a target rather than an available input—the neural solver cannot invoke it directly. Instead, it generates a raw block $\mathbf{B}_t$ and selects the minimum-residual correction within $\mathrm{span}(\mathbf{B}_t)$, computed by least squares: $\mathbf{d}_t = \mathbf{B}_t(\mathbf{A}\mathbf{B}_t)^+\mathbf{r}_t, \|\mathbf{r}_t - \mathbf{A}\mathbf{d}_t\|_2 = \min_{\mathbf{y}} \|\mathbf{r}_t - \mathbf{A}\mathbf{B}_t\mathbf{y}\|_2$. The inverse approximation is thus selected by projection onto the generated block, rather than prescribed independently for each column.

**Gate-RINS: diagonal gating of local polynomial actions.** Here we follow the description in the Methodology while omitting the subscript $t$ that denotes the iteration step. For Gate-RINS generating $m$ basis vectors at each neural iteration, the block is $\mathbf{B} = [\mathbf{D}_1\mathbf{b}_1, \cdots, \mathbf{D}_m\mathbf{b}_m]$, where $\mathbf{D}_j$ is a diagonal matrix, $\mathbf{D}_j = \mathrm{diag}(g_{1j}, \cdots, g_{nj})$, and $\mathbf{b}_j = \sum_{\ell=0}^{l} \alpha_{j\ell}\mathbf{c}^{(\ell)}$. By absorbing the coefficients of the Chebyshev polynomials into $\alpha$, we obtain $\mathbf{b}_j = \sum_{\ell=0}^{l} \alpha_{j\ell}\mathbf{A}^\ell\mathbf{r}$. Each column of the block $\mathbf{B}$ can therefore be written as $\sum_{\ell=0}^{l} \alpha_{j\ell}\mathbf{D}_j\mathbf{A}^\ell\mathbf{r}$. If we restrict the model to generating one basis vector per iteration, omit the $\ell = 0$ term, and set $l = 1$, the resulting basis reduces to $\mathbf{D}\mathbf{A}\mathbf{r}$, which can be interpreted as an inverse-like residual-adapted action related to preconditioned Krylov iteration. When $\mathbf{D}$ commutes with $\mathbf{A}$, this can further be written as $\mathbf{D}\mathbf{A} = \mathbf{A}\mathbf{D} = \mathbf{A}\mathbf{M}^{-1}$ by identifying $\mathbf{M}^{-1} = \mathbf{D}$. The distinction is that $\mathbf{D}$ is residual-adaptive: $\mathbf{D} = \mathbf{D}(\mathbf{r}_t)$.

**GRANS: sparse propagation in a lifted feature space.** GRANS exhibits richer structural patterns. **Firstly**, we restrict $l = 1$, corresponding to one message-passing layer per neural iteration, and omit the subscript $t$ denoting the iteration step. $\mathbf{B} = \mathrm{MLP}(\mathbf{h}) \in \mathbb{R}^{n\times q}$, where $\mathbf{h}_i^{l+1} = \sum_{j\in\mathcal{N}_i} \gamma_{ij}\mathbf{W}\mathbf{h}_j^l \in \mathbb{R}^q$, with $i, j = 1, \cdots, n$. **Secondly**, let $q = 1$, namely, the embedding dimension of each node is reduced to one. Then $\mathbf{h}_i$ degenerates from a vector to a scalar, and the matrix $\mathbf{W}$ becomes a scalar as well. We then have

$$h_i^{l+1} = \sum_{j\in\mathcal{N}_i} \gamma_{ij} h_j^l, \tag{1}$$

where $W$ has been absorbed into $\gamma$. **Thirdly**, ignoring the coordinate input $\mathbf{X}_{\mathrm{coord}}$, we have $h_i = \beta r_i$, $i = 1, \cdots, n$, where $\beta$ is a scalar induced by the MLP and normalization. We then have $h_i^{l+1} = \sum_{j\in\mathcal{N}_i} \gamma_{ij} h_j^l$, namely, $\mathbf{h}^{l+1} = \mathbf{\Gamma}\mathbf{h}^l \in \mathbb{R}^n$, where $\mathbf{\Gamma}$ shares the same sparsity pattern as $\mathbf{A}$. Under the restriction $l = 1$, this gives $\mathbf{h} = \mathbf{\Gamma}\mathbf{r}$. Again, this can be interpreted as a residual-dependent preconditioning action, where $\mathbf{\Gamma} = \mathbf{\Gamma}_\theta(\mathbf{A}, \mathbf{r})$ is trained to approximate the action of $\mathbf{A}^{-1}$. The distinction is that $\mathbf{\Gamma}$ is residual-adaptive. For multilayer propagation, the output takes the form $\mathbf{h} = \mathbf{\Gamma}_1\mathbf{\Gamma}_2 \cdots \mathbf{\Gamma}_l\mathbf{r}$, whose effective sparsity pattern can become denser than that of $\mathbf{A}$.

**Lifting to the tensor-product space.** When $q = 1$, $\mathbf{\Gamma} \in \mathbb{R}^{n\times n}$ is a sparse matrix sharing the same sparsity pattern as $\mathbf{A}$. When $q > 1$, the corresponding propagation operator acts in $\mathbb{R}^{nq}$ and can be represented as a block-sparse matrix in $\mathbb{R}^{nq\times nq}$. Each node is associated with a $q$-dimensional embedding $\mathbf{h}_i \in \mathbb{R}^q$. The original node space $\mathbb{R}^n$ is lifted to the tensor-product space $\mathbb{R}^n \otimes \mathbb{R}^q$, where the first factor indexes the nodes and the second represents the latent feature space within each node. High-dimensional GRANS can therefore be interpreted as learning a block-sparse operator in a lifted state space, rather than directly learning an ordinary sparse matrix in the original space $\mathbb{R}^n$. Given an original residual $\mathbf{r} \in \mathbb{R}^n$, it is first lifted through an encoding map $E$ to $\mathbf{H} \in \mathbb{R}^{n\times q}$, then propagated in the lifted space by an operator $\mathcal{D}_\theta$, and finally projected back to the physical space through a projection head $\Pi$. The overall process can therefore be written as

$$\mathbf{r} \xrightarrow{E} \mathbf{H} \xrightarrow{\mathcal{D}_\theta} \mathbf{H}' \xrightarrow{\Pi} \mathbf{q},$$

so that the effective operator acting on the original physical space can be written as

$$D_{\mathrm{eff}}(\mathbf{r}) = \Pi\mathcal{D}_\theta(\mathbf{r})E.$$

From this perspective, the high-dimensional embedding can be viewed as a lifting mechanism: the original problem is first lifted to a higher-dimensional space endowed with auxiliary internal degrees

of freedom, block-sparse dynamics are then performed in the lifted space, and the result is finally projected back to the original solution space.

**The limits of commutativity.** For $q = 1$, the coefficients $\gamma_{ij}$ in (Eq. equation 1) are scalars and therefore naturally satisfy $\gamma_{ij}\gamma_{ji} = \gamma_{ji}\gamma_{ij}$. This property does not generally hold after lifting to the tensor-product space, where $\mathbf{\Gamma}_{ij}\mathbf{\Gamma}_{ji} - \mathbf{\Gamma}_{ji}\mathbf{\Gamma}_{ij} \triangleq [\mathbf{\Gamma}_{ij}, \mathbf{\Gamma}_{ji}] \neq 0$. We refer to this property as the *path-propagation noncommutativity* of GRANS. We acknowledge that its theoretical role is not yet fully understood. We leave a systematic theoretical investigation of this property to future work. Motivated by this observation, we design a variant of GRANS, termed Block-GRANS, to partially restrict this path-propagation noncommutativity. As shown in this paper, Block-GRANS exhibits distinct empirical properties and achieves better performance in most of the evaluated settings.

### C.2 From GRANS to Block-GRANS

At neural iteration $t$, the model receives the current residual $\mathbf{r}_t$, the sparse matrix graph induced by $\mathbf{A}$, edge attributes $a_{ij}$, and node coordinates, and returns a multi-column basis block that is inserted into the same projected minimum-residual closure. Block-GRANS modifies only the admissible graph basis generator while leaving the algebraic update unchanged.

**Restating GRANS.** In GRANS, node coordinates are first mapped to a learnable positional encoding $\mathbf{z}_i = \phi_{\mathrm{PE}}(\mathbf{x}_i)$, and the initial node feature is constructed as $\mathbf{h}_i^{(0)} = [\widetilde{r}_{t,i}, \mathbf{z}_i]$, where $\widetilde{\mathbf{r}}_t$ denotes the per-graph normalized residual. For each graph edge $(j, i)$ induced by a nonzero entry of $\mathbf{A}$, a GATv2 layer computes an attention weight from the target feature, source feature, and edge value $a_{ij}$. Abstractly, one can write

$$\gamma_{ij}^{(\ell)} = \mathrm{softmax}_{j\in\mathcal{N}(i)}\, a_{\boldsymbol{\theta}}^{(\ell)}\big(\mathbf{h}_i^{(\ell-1)}, \mathbf{h}_j^{(\ell-1)}, a_{ij}\big), \qquad \widehat{\mathbf{h}}_i^{(\ell)} = \sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(\ell)} \mathbf{W}_s^{(\ell)} \mathbf{h}_j^{(\ell-1)}. \tag{2}$$

After batch normalization and an ELU nonlinearity, each layer has its own projection head $\psi_\ell$, producing $\mathbf{V}^{(\ell)} = \psi_\ell(\mathbf{H}^{(\ell)}) \in \mathbb{R}^{n\times q}$. The raw GRANS block is then formed by concatenating the outputs from all message-passing layers, $\mathbf{V}_{\mathrm{raw}} = [\mathbf{V}^{(1)}, \mathbf{V}^{(2)}, \ldots, \mathbf{V}^{(l)}] \in \mathbb{R}^{n\times lq}$, followed by graph-wise QR orthogonalization before the least-squares projection step. In the default setting, $l = 4$ and $q = 5$, so one forward pass emits 20 basis directions.

**Characterizing Block-GRANS.** Block-GRANS preserves the same input construction, edge attention, per-layer basis heads, concatenation, and QR orthogonalization as GRANS. The only architectural modification is imposed on the source feature transformation inside each attention layer. Instead of using a dense source projection $\mathbf{W}_s^{(\ell)}$, Block-GRANS partitions the latent feature channels into $p_{\mathrm{blk}}$ contiguous blocks and replaces the dense projection by a block-diagonal map,

$$\mathbf{W}_{\mathrm{blk}}^{(\ell)} = \mathrm{diag}\big(\mathbf{W}_1^{(\ell)}, \mathbf{W}_2^{(\ell)}, \ldots, \mathbf{W}_{p_{\mathrm{blk}}}^{(\ell)}\big). \tag{3}$$

If the node embedding follows the same partition, $\mathbf{h}_j^{(\ell-1)} = [\mathbf{h}_{j,1}^{(\ell-1)}, \ldots, \mathbf{h}_{j,p_{\mathrm{blk}}}^{(\ell-1)}]$, then the propagated message becomes $\mathbf{m}_{ij}^{(\ell)} = \alpha_{ij}^{(\ell)} \mathbf{W}_{\mathrm{blk}}^{(\ell)} \mathbf{h}_j^{(\ell-1)} = \alpha_{ij}^{(\ell)} [\mathbf{W}_1^{(\ell)} \mathbf{h}_{j,1}^{(\ell-1)}, \ldots, \mathbf{W}_{p_{\mathrm{blk}}}^{(\ell)} \mathbf{h}_{j,p_{\mathrm{blk}}}^{(\ell-1)}]$. The layer then aggregates these messages over the same graph induced by $\mathbf{A}$, $\widehat{\mathbf{h}}_i^{(\ell)} = \sum_{j\in\mathcal{N}(i)} \mathbf{m}_{ij}^{(\ell)}$, while the remaining forward computation is identical to GRANS: $\mathbf{B}_t^{\text{Block-GRANS}} = \mathrm{qr}\left([\psi_1(\mathbf{H}^{(1)}), \ldots, \psi_l(\mathbf{H}^{(l)})]\right)$.

From the lifted-space perspective introduced above, standard GRANS applies dense transformations within the latent feature space $\mathbb{R}^q$ at each graph propagation step, allowing arbitrary cross-channel mixing before information is transported to neighboring nodes. Block-GRANS restricts this internal transformation to a direct sum of lower-dimensional subspaces,

$$\mathbb{R}^q = \mathcal{H}_1 \oplus \cdots \oplus \mathcal{H}_{p_{\mathrm{blk}}},$$

so that each source transformation acts independently within the corresponding latent block. Equivalently, the block-sparse operator in the lifted space retains the same spatial graph connectivity as GRANS, while its internal channel transport is restricted by the block-diagonal structure.

**The tunable path-propagation commutativity of Block-GRANS.** With $p_{\mathrm{blk}} = 1$, $\mathbf{W}_{\mathrm{blk}}^{(\ell)}$ reduces to the original dense source projection and Block-GRANS recovers the standard GRANS source

transformation. As $p_{\mathrm{blk}}$ increases, cross-block channel mixing is progressively removed, reducing the degrees of freedom of the latent transport operator. In the limiting case where each block contains a single latent channel, the internal source transformation becomes strictly diagonal. Absorbing the scalar attention coefficient $\gamma_{ij}$ into the diagonal map yields a local transport operator $\mathbf{\Gamma}_{ij} = \gamma_{ij}\mathbf{W}_{ij}$, which is also diagonal. Therefore, for any two local transport operators,

$$[\mathbf{\Gamma}_{ij}, \mathbf{\Gamma}_{kl}] = 0,$$

and the local path-propagation commutativity in the latent feature space is recovered.

# D Supplementary Experimental Results

## D.1 Preconditioning Integration Experiments

To test whether RINS composes with an existing numerical acceleration technique, we repeat the six-PDE $n \approx 1000$ benchmark after applying the same left Jacobi preconditioner to every training and inference linear system. The preconditioned datasets are deterministic transforms of the main benchmark splits, so all three methods see the same matrices and right-hand sides after preconditioning. Gate-RINS and GRANS are trained on the 200 preconditioned training systems for each PDE with the same $50/50/100$ schedule and the same 20-direction-per-iteration budget. At inference time, preconditioned RINS, preconditioned GRANS, and preconditioned GMRES are evaluated on the same 10 held-out systems. As in the main benchmark, GMRES is stopped when its relative residual reaches the better final residual obtained by RINS and GRANS on that PDE, or when the maximum iteration budget is exhausted.

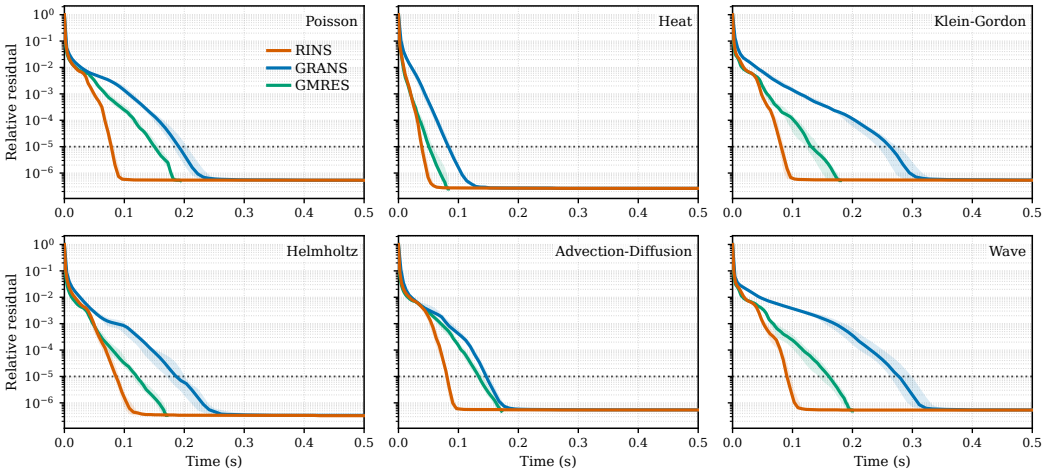



Figure 7: Preconditioned six-PDE inner-solve workflow at $n \approx 1000$. RINS, GRANS, and GMRES are all evaluated on the same left-Jacobi-preconditioned systems; curves report relative residual versus cumulative inner linear-solve time.

## D.2 One-Basis Limit Test

We additionally run a one-basis limit test to separate the quality of each generated direction from the advantage of producing many directions per neural iteration. In the archived checkpoints used for Figure 8, the RINS one-basis models are the gate-RINS variant with `num_filters=1` (namely, $m$=1) and `cheb_order=4` (namely, $L = 4$). The experimental protocol is otherwise identical to the main six-PDE benchmark: each task uses the same 200 training samples and the same 10 held-out test samples, and both Static-RINS and GRANS are trained with the same three-stage schedule of 100 epochs in the first stage, 100 epochs in the second stage, and 150 epochs in the third stage. The learning rate, weight decay, checkpoint interval, loss settings, and inference protocol are also unchanged. The only architectural change is the per-iteration basis budget. In the main benchmark, both neural solvers generate 20 new basis directions at each neural iteration. In this limit test, Static-RINS and GRANS generate only one new basis direction per neural iteration. Consequently, the accumulated neural subspace dimension grows at the same rate as unrestarted GMRES.

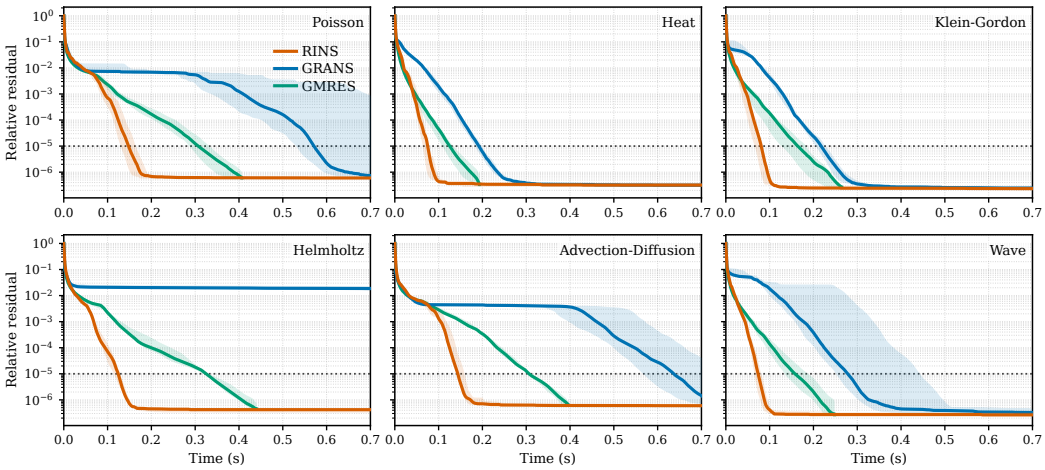



Figure 8: One-basis limit test at $n \approx 1000$. Solid curves show median relative residual versus cumulative solve time, shaded bands show interquartile ranges.

Figure 8 shows the six $n \approx 1000$ PDE tasks. Curves report the median over the held-out test samples and shaded regions show the interquartile range; numeric markers indicate 50 and 100 generated directions or GMRES iterations. Across these tasks, Gate-RINS still decreases the residual faster than the graph-only GRANS baseline. More importantly, Gate-RINS often reaches the target residual before the accumulated subspace reaches 100 dimensions, indicating that the learned low-dimensional residual-image subspace already contains an effective correction for the solution. The same H-100 GPU timing protocol used in the main benchmark shows that, even under this one-basis-per-iteration restriction, Gate-RINS reaches the reported target residual with less iterations than GMRES on these tasks. By contrast, GRANS with only one generated direction per neural iteration fails to converge in these runs. This behavior supports the interpretation that the polynomial residual-image generator is not only benefiting from a larger per-iteration basis budget in the main benchmark; its individual generated directions are also substantially useful for the projected minimum-residual update.